А.С.Подколзин

Введение в логические процессы

2025 г.

Введение

Эта книга посвящена исследованию по искусственному интеллекту. Однако, прежде, чем говорить об искусственном интеллекте, стоит повнимательнее присмотреться к естественному интеллекту, его формированию и его возможностям. Созданная природой нейросеть обладает способностью обучаться на примерах, вырабатывая, большей частью неосознанно, системы признаков, позволяющие распознавать различные ситуации. Однако, самой по себе этой способности недостаточно, и нейросеть с самых первых шагов погружается в среду обучения, программирующую ее интеллект. Такая среда формировалась человечеством на протяжении длительного времени и представляет собой целую пирамиду знаний и технологий, зафиксированную в книгах и статьях. Впитывая эту информацию, нейросеть становится в значительной степени лишь вычислительным устройством, в которое закладывается программа. Учебники передают ей знания и приемы решения задач, а способность обучаться на примерах помогает распознавать ситуации, в которых та или иная информация должна применяться. Таким образом, нейросеть алгоритмизирует пассивные знания, формируя некий Алгоритм Рассуждений, лишь частично описанный в учебниках, а в остальном уточняемый при обучении. Она им пользуется, но не способна в полной мере осознавать, как именно он устроен. Интуиция может подсказывать то или иное действие в конкретной ситуации, но почему она это делает - зачастую остается загадкой.

Первое, что приходит в голову в связи с искусственным интеллектом - создать искусственную нейросеть и заняться ее обучением. По этому пути и двигается большинство современных исследователей. Пока архитектура искусственных нейросетей сильно отличается от архитектуры нейросетей естественных, даже нейрон в них не такой. Для повышения уровня интеллекта этих сетей приходится привлекать весьма дорогостоящие суперкомпьютеры. Однако, даже наиболее продвинутые версии, такие, как ChatGPT, характеризуются крайне противоречивыми отзывами - от восторженных заявлений о том, что через 2 - 3 года они во всем превзойдут интеллект человеческий до заявлений типа "он тупит и рассуждать не умеет". Оставим в стороне критику современных искусственных нейросистем. Все это неважно. Вряд ли стоит сомневаться в том, что рано или поздно будет создана искусственная нейросеть, превосходящая по своему уровню интеллекта естественную. НО. Как и естественная нейросеть, она будет аппроксимировать Алгоритм Рассуждений, довольно смутно представляя себе, как он работает. На этом пути ни нам, ни искусственному интеллекту не удастся расшифровать данный Алгоритм. Его расшифровка потребует совсем иных усилий.

Вообще-то говоря, рассуждения - это мир логики, логических процессов. Можно усомниться в том, что нейросеть представляет собой наиболее эффективное вычислительное устройство для выполнения логических вычислений. Природа создавала ее, чтобы догонять, убегать и ловить, а не для того, чтобы теоремы доказывать. Конечно, она приспособилась и к этому, но об огромной избыточности такой реализации

логических процессов можно только догадываться. Подобно тому, как для арифметических вычислений обычный процессор гораздо эффективнее любой нейросети, для экономичной реализации логических вычислений понадобится свой, логический процессор. Нейросети здесь ни при чем.

Однако, логический процессор нужно еще запрограммировать, а для этого необходимо расшифровать упомянутый выше Алгоритм Рассуждений - со всеми его элементами, включая приемы решения задач, приемы проведения исследований, приемы создания новых приемов и т.д. Попытка такой расшифровки и предпринимается в данной работе. Она привела к созданию компьютерной системы, обученной на 12000 примерах решению задач в различных областях математики, а также элементарных физике и химии. Система уверенно решает стандартные задачи по известным задачникам, моделируя рассуждения человека и демонстрируя в этих областях интеллект, несопоставимо более высокий, чем у любой современной искусственной нейросети. В процессе обучения системы был накоплен материал, позволивший проследить всю цепочку алгоритмизации знаний и создать работающий прототип процедуры автоматического синтеза приемов. Никакого суперкомпьютера для системы не требуется - она прекрасно работает на обычных системных блоках и ноутбуках, причем работает быстро. Хотя уже сейчас система вобрала в себя огромное количество знаний, ее конструкция позволяет продолжать обучение, практически неограниченно увеличивая объем этих знаний. Впрочем, система развивалась в первую очередь как инструмент для расшифровки процесса алгоритмизации знаний, а ее способность решать задачи использовалась лишь для контроля адекватности такой расшифровки.

С точки зрения математики, проблема алгоритмизации знаний сводится к вопросу: как из теорем извлечь приемы решения задач? Прежде всего, пришлось накопить достаточно большое количество уже "алгоритмизированных" знаний и понять, как они должны быть организованы в компьютерной системе для того, чтобы она могла эффективно решать задачи. Прорабатывались примеры из множества различных предметных областей. Процесс решения разбивался на элементарные шаги, и для каждого из них предлагалось объяснение в виде небольшой программы - "приема", который в аналогичной ситуации выполнял аналогичные действия. Для приема определялось то ключевое понятие, появление которого в текущем контексте должно было инициировать попытку применения приема. За каждым понятием закреплялась ветвь программы решателя, к которой относились соответствующие приемы, и база приемов оказалась организована как энциклопедия приемов. Решение задачи происходило в процессе сканирования ее описания и обращения для текущего понятия к соответствующей ветви данной энциклопедии. Таким образом, система получила что-то вроде внутреннего "логического зрения" и могла принимать решение об очередном действии с учетом всей текущей картины.

Так как каждый прием действовал автономно и независимо от других приемов, а общее количество приемов на текущий момент достигло более чем 50000, возникла проблема организации разумного их взаимодействия, которое закладывалось в решающие правила приемов в процессе обучения на примерах. Всего было рассмотрено более 13000 задач из различных предметных областей: дискретная математика, алгебра множеств, элементарная алгебра, элементарная геометрия, аналитическая геометрия, линейная алгебра, математический анализ, дифференциальные уравнения, интегральные уравнения, комплексный анализ, теория вероятностей, общая алгебра, элементарная физика, элементарная химия, распознавание рукописных букв, текстовый анализ, шахматы. Система отображает процесс решения задачи "по ша-

гам" и оказалась способна решать многие задачи уровня конкурсных экзаменов по математике. Например, пошаговый показ решения задач по элементарной алгебре, который в последнее время демонстрирует программа С.Вольфрама, данная система умела делать еще 25 лет назад. Неплохо справляется она со стандартными задачами по элементарной геометрии, а также задачами из других перечисленных выше разделов. Таким образом, можно считать, что понимание того, как выглядят алгоритмизированные знания, было до некоторой степени достигнуто.

Чтобы упростить создание приемов, был создан специальный язык программирования ЛОС (Логический Описатель Ситуаций), максимально приближенный к логическому языку. Главной его задачей была формулировка сложных условий на целесообразность применения приема в текущем контексте. В этих условиях разрешалось использовать кванторы и описатели, причем операторы языка работали в режиме перечисления значений выходных переменных, позволившем обходиться без операторов цикла. Хотя язык ЛОС и оказался близок к известному языку логического программирования ПРОЛОГ, в отличие от ПРОЛОГа он ориентирован не на формулировку теоремы предметной области, а на формулировку условий управления этой теоремой. Для решателей это оказалось более важным, так как часто запись управляющей компоненты приема была во много раз сложнее записи теоретической компоненты. ЛОС существенно ускорил процесс программирования и упростил чтение программ. Для выполнения его программ создан интерпретатор, и вся работа системы, включая интерфейсы, происходит через ЛОС.

Дальше начался процесс постепенного движения вспять - от алгоритмизированных в виде приемов знаний к их источникам. Практика обучения решателя показала, что обычно прием основан на какой-то единственной теореме. В программе ЛОСа фрагменты этой теоремы и управления теоремой перемешаны достаточно хаотичным образом. Естественным первым шагом на пути к истокам программ стал переход к раздельной записи теоремы и управления. Для такого разделения был создан язык логического программирования ГЕНОЛО́Г, в котором прием задается теоремой, сопровожденной некоторой алгоритмизирующей разметкой ("генотипом" приема), понятной компилятору. Этот язык, в отличие от ЛОСа, не является непосредственно исполняемым. Компилятор преобразует описание приема на ГЕНОЛО́Ге в программу ЛОСа.

Чтобы сформулировать условия целесообразности применения теоремы в текущем контексте, ГЕНОЛО́Г использует полномасштабный логический язык. Таким образом, описание приема имеет два логических уровня - уровень предметной области, на котором задается теорема, и уровень структур данных, на котором задается управление теоремой. В этом заключается принципиальное отличие ГЕНОЛОГа от других языков логического программирования. Создание ГЕНОЛОГа происходило постепенно, по мере проработки задач из различных разделов. Фактически, он представляет собой огромную коллекцию способов алгоритмизации теорем. ГЕНОЛОГ настолько упростил и ускорил создание приемов, что позволил в сравнительно короткие сроки накопить их запас, достаточный, например, для решения задач по планиметрии. Проработка перечисленных выше разделов, в которых было создано более 50000 приемов решателя, была осуществлена на ГЕНОЛОГе. Предшествующие шесть томов данной монографии посвящены изложению этих приемов и описанию общей организации компьютерной логической системы.

Однако, ГЕНОЛО́Г оказался лишь промежуточным пунктом на пути от приемов к породившим их теоремам. Алгоритмизирующая разметка теоремы была нацелена

лишь на то, чтобы подробно объяснить компилятору, как по теореме создавать ЛОС-программу приема. В ней ничего не говорилось о целях применения приема. Чтобы автоматизировать создание таких разметок, была предпринята классификация приемов ГЕНОЛО́Га по целевому признаку. Согласно этой классификации, целевая установка приема описывалась типом приема и небольшим набором сопровождающих его данных. Например, направлением тождественной либо эквивалентной замены, выделением каких-то переменных, подтермов, и т.п. Такая целевая установка, получившая название спецификации приема, оказалась фактически альтернативным способом задания приема. Язык задания приемов с помощью сопровождающих теорему спецификаций был назван логическим ассемблером. Аналогия с обычным ассемблером, хотя и весьма отдаленная, заключается в том, что тип приема уподобляется коду операции, а дополнения к нему - операндам.

Число типов приемов приближается к 1500. Наиболее часто встречаются порядка 300 из них. Для перехода от задания приема на логическом ассемблере к заданию его на ГЕНОЛО́Ге был создан компилятор. Однако, этот процесс компиляции потребовал привлечь принципиально новый элемент - доводку создаваемого приема на задачах с целью оптимизации его параметров и бесконфликтного "вживления" в базу приемов.

Логический ассемблер, хотя и оказался языком пограничного слоя между теоремами и программами, примыкает к этому слою со стороны программ. В первую очередь, из-за того, что теоремы приемов оказалось целесообразно, в целях упрощения компиляции, несколько "деформировать" по отношению к обычным теоремам, отбрасывая избыточные проверки и добавляя некоторые элементы технического характера. Все-таки, теоремы приемов представляют собой лишь фрагмент языка программирования. Чтобы перейти через "пограничный слой" между приемами и теоремами и далее продолжить работу со стороны базы теорем, нужно было прежде всего создать эту самую базу теорем.

Заполнение базы теорем непосредственно из учебников привело бы к существенному разрыву между ними и теоремами приемов. Теоремы приемов обычно содержали множество обобщающих параметров или представляли собой какие-то комбинации теорем "из учебников", ориентированные на решение задач и в учебниках обычно отсутствующие. Чтобы проследить источники приемов, нужно было избежать указанного разрыва. Поэтому первоначально база теорем заполнялась теоремами, представляющими собой аккуратные с точки зрения логики переформулировки теорем приемов. Она представляла собой как бы "проекцию" базы приемов. В большинстве случаев теорема из базы теорем попросту совпадала с теоремой приема.

Следующим вопросом было: как по теореме создавать спецификации приемов? Имеющаяся база теорем, привязанная к базе приемов и к уже готовым их спецификациям, позволила провести определенную классификацию теорем и выработать некоторый список стандартных характеристик теорем, подсказывающих возможные типы приемов для них. Большинство этих характеристик легко вычислялись непосредственно по теореме, и для сопровождения ими теоремы была создана специальная процедура, названная характеризатором. Создание других характеристик требовало понимания предыстории возникновения теоремы. Они должны были появляться лишь в процессе вывода теорем. Так или иначе, в базе теорем каждая теорема сопровождалась списком своих характеристик. Спецификации приемов создавались процедурой, просматривающей характеристики теоремы и предлагающей для текущей характеристики список возможных спецификаций. Эта процедура получила название спецификатора.

Собственно говоря, уже с этого момента появилась возможность автоматического создания приемов по теореме: сначала характеризатор сопровождает теорему списком характеристик, затем спецификатор предлагает по каждой из них возможные спецификации, далее компилятор спецификаций преобразует их в описания приемов ГЕНОЛОГа, и, наконец, компилятор ГЕНОЛОГа получает ЛОС-программы приемов.

Однако, такие приемы, созданные без учета того, какие приемы уже имеются в решателе, обычно оказываются бесполезными или даже вредными. Либо они дублируют то, что делалось другими приемами, либо бесплодные попытки их применения сильно замедляют работу, либо они вообще направляют ход решения по ошибочному руслу. Чтобы преодолеть это явление, понадобились еще два этапа обработки приема.

Прежде всего, предпринимается попытка создать для приема простую тестовую задачу, которая решалась бы данным приемом, но не решалась в его отсутствие. Так как тип приема известен и известна его целевая ориентация, достаточно, чтобы задача была лишь одноходовкой, проверяющей, что решатель способен сделать шаг в направлении нужной цели. Данный этап обеспечивает настолько хорошую фильтрацию, что ее проходят только те приемы, которые действительно расширяют возможности решателя. Фактически, тестовый пример служит как бы доказательством необходимости приема.

Однако, даже необходимый для одной задачи прием бывает способен "поломать" ход решения других задач обучающего материала, сохраняемого в задачнике решателя. Поэтому, после примерки на тестовых задачах, предпринимается прокрутка решателя по одному или нескольким разделам задачника для выявления тех задач, решение которых сильно замедлилось или на которые стал возникать отказ. На этих задачах предпринимается доводка приема: варьируется уровень срабатывания приема, предпринимается переход к подтипу приема, обеспечивающему более высокую степень мотивированности срабатывания, и т.п. При доводке учитывается, что прием по-прежнему должен решать свою тестовую задачу.

Лишь после примерки и доводки автоматически созданные приемы регистрируются в накопителе результатов. Так как система находится лишь на стадии обучения, окончательный отбор приемов из накопителя и перенесение их в основную базу приемов пока выполняется вручную. Обычно отклоняется лишь меньшая их часть, причем причиной служит крайне маловероятное возникновение ситуации, на которую рассчитан прием.

Но вернемся к рассмотрению теорем - до того момента, как для них генерировались спецификации. Как уже говорилось, те теоремы, по которым создаются приемы, редко совпадают с "базисными" теоремами из учебников. Обычно они представляют собой результат определенной переработки базисных теорем, необходимой для решения задач. Такая переработка может заключаться в том, что теорема снабжается множеством обобщающих параметров, ориентированных на применение ее в "неявных" ситуациях, либо в комбинировании нескольких теорем для вывода стандартной "заготовки" для часто встречающейся в задачах ситуации, и т.п. Поэтому, для завершения рассмотрения цикла алгоритмизации теорем, осталось обеспечить указанный переход от базисных теорем к теоремам, по которым будут создаваться приемы. Этот переход будем называть программирующим логическим выводом.

Извлеченные из базы приемов решателя теоремы оказались превосходным обучающим материалом для создания приемов программирующего вывода. Они были рас-

пределены по специальным подразделам оглавления базы теорем - своего рода задачам на программирующий вывод. В первом пункте подраздела располагались одна или несколько базисных теорем, в остальных пунктах размещались те теоремы - источники приемов, которые должны были получаться программирующим выводом из базисных теорем. Эти подразделы получили название ячеек логического вывода.

Разумеется, доказательства теорем изложены в учебниках и хорошо известны. Не составляет особого труда и доказательство их следствий, используемых для создания приемов. Но умение доказывать теоремы ничего не дает, если сами теоремы еще отсутствуют. Поэтому проработка программирующего логического вывода означала ни много ни мало анализ процессов "открытия" теорем, начиная хотя бы с их простых следствий.

Чтобы объяснять, как та или иная теорема ячейки могла бы быть открыта при анализе базисных теорем, приходилось находить цепочку достаточно естественных переходов, быть может с привлечением дополнительных теорем, которые тоже заносились в общую базу теорем - для дальнейшего объяснения их "происхождения". Для устанавления того, какие переходы являются естественными, использовались характеристики теорем. Они позволили придать переходам в цепочке вывода вполне определенную целевую направленность. Каждый переход оформлялся в виде небольшой программы, анализирующей теорему в контексте заданной ее характеристики. Такие программы, названные приемами программирующего логического вывода, аккумулировались в своеобразном "теоремном" решателе системы. На текущий момент он насчитывает более 1500 приемов.

Прием программирующего вывода - существенно более развитый объект, чем обычное правило вывода в математической логике. Во-первых, он должен самостоятельно находить в базе теорем дополнительные теоремы, которые в сочетании с текущей анализируемой теоремой будут давать полезные следствия. Здесь используются как оглавление базы теорем, так и специальные процедуры быстрого поиска теорем заданного типа. Во-вторых, прием программирующего вывода может обращаться к решателю для различных вспомогательных задач, подсказанных его целевой установокой. В результате срабатывание одного такого приема часто оказывается равносильным длинной цепочке применений обычных правил вывода, причем устройство ее непредсказуемо из-за подключения мощного аппарата всей базы приемов решателя. В-третьих, прием должен использовать определенные эвристические правила для блокировки вывода малополезных (например, чрезмерно громоздких) теорем. В частности, для этого используется блокировка определенных сочетаний последовательно применяемых приемов вывода. При обучении такая блокировка позволила устойчиво обеспечивать исчерпание возможностей дальнейшего вывода в ячейке и выдачу окончательного результата за приемлемое время. В особых случаях результаты вывода выносились в новые ячейки, и глубина вывода таким образом увеличивалась. В-четвертых, прием вывода должен сопровождать теорему характеристиками. Обычно для этого используется общая процедура характеризатора, но иногда прием сам указывает характеристики, объясняющие цель, ради которой он ее получил.

Грань между программирующим логическим выводом и исследовательским выводом является весьма условной. При проработке базы теорем оказалось, что получение многих "классических" теорем может быть объяснено приемами логического вывода того же уровня сложности, что и для их "технических" следствий. Несложные приемы, объясняющие, как одна теорема могла бы быть выведена из других,

были созданы, например, для формул корней квадратных и кубических уравнений в элементарной алгебре, теоремы Пифагора и теорем синусов и косинусов в планиметрии, свойств определителей в линейной алгебре, основных формул вычисления первообразных в математическом анализе и т.д. Фактически, теоремы прорабатываются почти подряд, без разделения на базисные и вторичные. Все это вывело процесс обучения решателей на качественно более высокий уровень. Если раньше анализировались задачи из задачников и нужно было предложить приемы, которые доводили аналогичные задачи до ответа, то теперь анализируются теоремы и предпринимаются попытки создать приемы для "открытия" новых теорем. При этом разрешается использовать весь ранее накопленный системой потенциал решения задач.

Создан прототип генератора приемов, функционирующий по следующей схеме. Выбирается ячейка логического вывода, и в ней запускается процесс вывода теорем. Обычно возникают десятки теорем. В процессе вывода каждая теорема снабжается характеристиками. По этим характеристикам генерируются спецификации приемов (обычно - тоже вплоть до десятка спецификаций на каждую теорему). Компилятор спецификаций преобразует те из спецификаций, для которых пока не созданы приемы, в описания приемов на ГЕНОЛОГе. Для текущего такого приема (пока не откомпилированного на ЛОС) создается тестовый пример. Проверяется, что этот пример не решается системой, после чего новый прием компилируется, и проверяется, что теперь тестовый пример решается. Для отобранных таким образом новых приемов предпринимается расчистка - удаляются приемы, тестовые задачи которых решаются другими новыми приемами. В результате остается лишь малая часть изначально созданных приемов. Для них предпринимается завершающая двухэтапная доводка - сначала происходит прогонка по тому разделу задачника системы, для которого создавались приемы, затем - по всему задачнику. После каждой прогонки отбираются "испортившиеся" задачи, и для восстановления их нормального решения приемы корректируются. В процессе доводки приемы сортируются на пригодные для перенесения в решатель и непригодные. Последние дают информацию для развития генератора приемов. Окончательное перенесение приема в основную базу приемов пока происходит вручную.

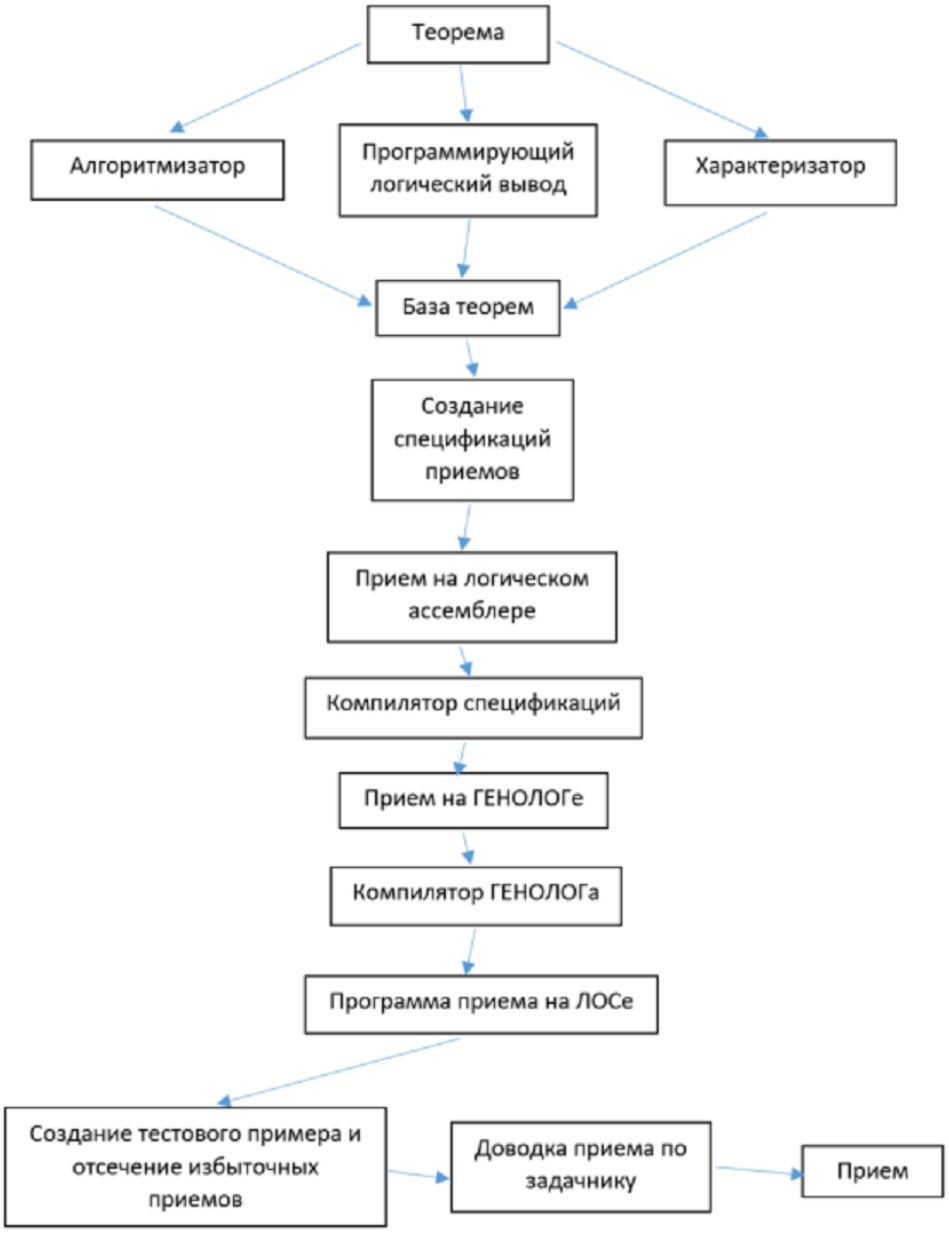
Указанный прототип был протестирован на различных разделах базы теорем и позволил создать более 2000 новых приемов, аналогичных ранее созданным вручную и не только им не уступающих, но иногда даже превосходящих, так как при ручном синтезе многие требующие учета особые случаи упускались из виду.

В действительности не все приемы решателя основаны на теоремах. Некоторые из них основаны на тех или иных общих особенностях конкретного раздела, сформулированных в виде так называемых протоколов базы теорем. Эти протоколы уточняют способы алгоритмизации теорем раздела; в частности, порождают приемы, обращающиеся для вывода следствий или преобразований к вспомогательным задачам безотносительно к каким-либо конкретным теоремам. Система, создающая протоколы при общем рассмотрении раздела базы теорем, названа алгоритмизатором. Технически алгоритмизатор является частью процедуры вывода теорем.

Наконец, упомянем об еще одном классе приемов - общелогических приемах, запрограммированных непосредственно на ЛОСе. Перевод их на ГЕНОЛОГ вряд ли целесообразен, так как ГЕНОЛОГ, по сути дела, является переходником между логическим языком предметной области и языком для описания структур данных, а в указанных приемах языком предметной области как раз и является язык структур

данных - ЛОС. Этих приемов немного, они имеют универсальный характер, и автоматизация их создания пока не актуальна. По-видимому, эту автоматизацию можно свести к решению задач, формулируемых в терминах предикатов и операций ЛОСа.

Приведем диаграмму, на которой представлены основные этапы и основные "действующие лица" изложенного процесса алгоритмизации теорем:



Подробное описание компьютерной системы содержится в многотомной монографии "Компьютерное моделирование логических процессов". Однако, она слишком велика и плохо подходит для первоначального знакомства. По существу, она представляет собой скорее технический отчет, нежели учебник. Настоящая книга призвана устранить этот недостаток.

Автор выражает искреннюю благодарность В.Б.Кудрявцеву, поддержка которого сделала возможным проведение данного исследования.

Оглавление

Глава 1

Логическая система "Искра"

1.1 Логические процессы

К какой бы области ни относился процесс рассуждений, текущее состояние этого процесса характеризуется множеством рассматриваемых объектов и анализируемых связей между ними. Такое состояние можно исчерпывающим образом описать совокупностью утверждений некоторого логического языка. Какие-то из этих утверждений будут истинными, какие-то - лишь предположениями, какие-то и вовсе будут сформулированы с помощью не очень четких понятий. Это несущественно, так как во всех этих случаях они, все-таки, будут записаны на логическом языке. Разумеется, такой логический контекст при необходимости должен сопровождаться нелогической уточняющей информацией, оформленной как комментарии к логическим объектам. Например, могут указываться степени уверенности, вероятности, ссылки на битмэпы изображений и на фрагменты этих битмэпов, и т.п. Логический контекст будет играть роль структуры, связующей все эти нелогические элементы в единое целое.

В процессе рассуждений логический контекст изменяется: при выводе следствий в него заносятся новые элементы, старые видоизменяются, ненужные элементы удаляются, поступает информация из внешнего мира, принимаются решения о внешних реакциях, фиксируемые как элементы контекста, и т.д. Возникает своего рода логический автомат, состояниями которого служат логические контексты, а функция переходов образована огромным множеством приемов, анализирующих эти контексты и изменяющих их. Поведение такого автомата и будем называть логическим процессом.

В книге описывается компьютерная система, предназначенная для моделирования логических процессов и получившая название "Логическая система Искра".

1.2 Установка и запуск логической системы

Скачать программу логической системы можно на сайте кафедры МАТИС механико-математического факультета МГУ (intsys.msu.ru). Раздел сайта "Исследования - Автоматический решатель". Там же можно скачать монографию "Компьютерное моделирование логических процессов".

Программа написана под Windows. Чтобы установить ее, нужно разархивировать файл *logsyst.zip* и разместить полученную директорию в произвольном месте на компьютере. Размер директории невелик - около 500 МБ. Фактически, программа

хранится и исполняется в почти-архивированном виде, что несущественно сказывается на ее быстродействии. Малые размеры программы не должны вводить в заблуждение - объем хранящихся в ней знаний и приемов огромен.

Никаких дополнительных действий по установке не требуется. Запускается программа файлом *logsyst.exe*. Он представляет собой интерпретатор языка ЛОС. На всякий случай, подробно опишем действия по созданию ее ярлыка на рабочем столе. Правой кнопкой мыши вызывается контекстное меню, в котором выбирается пункт "Создать", далее - "Ярлык". Через "Обзор" находится директория, в которой размещена система, выбирается файл *logsyst.exe*, нажимается ОК, "Далее" и "Готово".

Собственно, для начала этого достаточно. Приведем ряд дополнительных сведений о директории системы. Основные файлы системы хранятся в поддиректориях GEN,INF,LOS,TCH,TER,TXT,BIT. Поддиректория ALT служит для размещения в ней альтернативной версии системы, если нужно перенести из нее какие-то данные в основную версию. Поддиректория COPY хранит последнюю сохраненную копию системы - на случай необратимой порчи основной ее версии. Директория FLS представляет собой буфер для хранения произвольных "чужих" файлов - в нем они будут доступны системе для чтения и изменения.

C++ - файлы ARIFM96, logsyst, polinom, RISUNOK, SHAHMATI, vichprog образуют проект logsyst.cbp для компиляции программы *logsyst.exe*. Для компиляции использована бесплатная платформа IDE Code::Blocks, компилятор GCC. Чтобы развивать интерпретатор ЛОСа, их нужно установить.

1.3 Вводная экскурсия по системе

Система предоставляет достаточно эффективный решатель задач из различных разделов математики и показывает процесс поиска решения "по шагам". Еще 25 лет назад она могла решать "по шагам" задачи стандартных задачников по элементарной алгебре, и лишь недавно это было повторено С.Вольфрамом в его системе "Математика". Сегодня система "Искра" успешно справляется со стандартными задачи средней сложности из самых разных разделов: элементарная алгебра, планиметрия, математический анализ, дифференциальные и интегральные уравнения, аналитическая геометрия, комплексный анализ и т.д. Иногда даже она набирала "проходные" баллы по вариантам вступительного письменного экзамена на механико-математический факультет МГУ. Более того, успешно решала задачи письменного выпускного экзамена факультета.

Однако, система развивалась лишь как инструмент для изучения логических процессов. Поэтому никаких попыток создать на ее основе программный продукт для массового пользователя не предпринималось. Этим объясняется и "бедный" на первый взгляд интерфейс системы. Несмотря на свою внешнюю непритязательность, он тем не менее прошел многолетнюю оптимизацию для обеспечения максимально быстрого выполнения различных действий. Переход к более привычным "стандартным" элементам интерфейса лишь замедлил бы работу.

Однако перейдем к экскурсии по системе. При запуске ее на экране появляется главное меню - белый прямоугольник, разделенный вертикальными и горизонтальными линиями на двенадцать клеток - четыре горизонтальные полосы по три клетки в полосе. Активация клетки происходит либо однократным нажатием левой кнопки

мыши, либо нажатием клавиши, указанной в клетке. В названиях клавиш используется только кириллица. Для вызова контекстного меню в любой ситуации, кроме главного меню, нажимается F1. Исключение единственное - при работе в текстовом редакторе контекстное меню вызывается нажатием Ctr-F1. Правая кнопка мыши обычно используется для других целей и контекстного меню не вызывает.

Рассмотрим несколько подробнее клетки главного меню:

1. Справочник по системе (F1). Нажатие F1 переводит в оглавление справочника по системе.

   Здесь уместны несколько слов, относящихся к любым оглавлениям системы. Пункты оглавления пронумерованы. Если после номера идет закрывающая скобка, то этот пункт соответствует входу в подменю, если же идет точка, то пункт концевой. Чтобы перемещаться по оглавлению, достаточно четырех клавиш курсора. Клавиши "вверх" - "вниз" либо колесико мыши изменяют выделенный пункт оглавления. Клавиша "вправо" либо нажатие левой кнопки мыши на выбранном пункте переводит в просмотр его содержимого. Для возвращения служит клавиша "влево". Если номер пункта оглавления - цифра, то нажатие на эту цифру тоже переведет в просмотр содержимого пункта. Если текущее меню оглавления не помещается на экране целиком, то для ускоренной его прокрутки можно использовать клавиши "PageUp" и "PageDown". При полной прокрутке пунктов меню в направлении "вверх" экран может оказаться пустым. Это не страшно - обратной прокруткой исчезнувшие пункты возвращаются.

   Содержание справочника по системе до некоторой степени дублирует содержание данной книги. К тому же, многие его разделы дублируются другими справочными средствами системы, о которых речь пойдет дальше. Поэтому мы не будем останавливаться на нем подробно. Пожалуй, наиболее востребованным пунктом оглавления справочника служит пункт "Формульный редактор". Он содержит сведения о клавишах, используемых при наборе формул в обычной математической записи. При чтении текстов, представленных в справочнике, прокрутка вверх-вниз выполняется соответствующими клавишами курсоров, колесиком мыши и клавишами "PageUp", "PageDown".

2. Оглавление задачника (з).

   Задачи, использованные при обучении системы, обычно сохраняются в ее задачнике. Это не значит, что запоминаются решения задач - задачник используется лишь для проверки того, что новые приемы не нарушают ее способности решать старые задачи. Конечно, для этого приходится сохранять ответы задач и данные о времени, затраченном на их решение (в специальных единицах, не зависящих от выбора компьютера).

   При переходе по клавише "з" попадаем в некоторый раздел оглавления задачника. Возможно, не корневой, так как оглавление запоминает раздел, в котором находилось в последний раз, и при повторном использовании сразу переходит в тот пункт указанного раздела, который был выделен. Чтобы перейти в корневой пункт, жмем на клавишу "курсор влево", пока картинка не стабилизируется.

В корневом меню оглавления задачника перечислены разделы - "Дискретная математика", "Теория множеств", "Элементарная алгебра", и т.д. Перемещаясь вглубь этих разделов, в конце концов приходим к "концевому" меню, в котором имеются только концевые пункты, причем после каждого номера идут три тире. Эти пункты суть отдельные задачи.

Если нажать на таком пункте "курсор вправо", то на экране появится условие соответствующей задачи, расположенное в верхней части экрана между двумя горизонтальными линиями. Если оно небольшое, то непосредственно под его нижней линией будет прорисовано условие, следующей задачи, и т.д. Если большое - нижней линии видно не будет. Фактически, здесь прорисован фрагмент "ленты задач" для задач концевого меню. Эту ленту можно прокручивать вверх-вниз соответствующими клавишами курсора и клавишами "PageUp", "PageDown". Важно понимать, что текущей задачей считается та, верхняя отделяющая линия которой - самая верхняя из прорисованных на экране. Если нажать клавишу "курсор влево", то возвращение произойдет к номеру меню, который соответствует текущей задаче. Чтобы вести прокрутку ленты задач таким образом, что каждый переход приводит к очередной задаче (верхняя линия задачи при этом пойдет по верхней границе экрана), нужно пользоваться клавишами "Ctr-курсор вверх" и "Ctr-курсор вниз".

Подробно о том, как создавать новые задачи, изменять старые и запускать их решение, будет рассказано далее. Пока заметим, что для получения лишь ответа на текущую задачу, без пошаговой трассировки ее решения, служит клавиша "о", а для пошаговой трассировки - клавиша "р", причем очередной шаг прорисовывается при нажатии клавиши Enter. Для получения более полной информации при трассировке решения имеются дополнительные возможности.

3. Решить задачу (Ctr-з)

   Этот пункт главного меню предлагает своего рода "логический калькулятор" для ускоренного ввода стандартных задач и получения ответа на них. Нажатие клавиши "Ctr-з" переводит в оглавление, содержащее краткую инструкцию и перечень разделов. После ввода задачи и нажатия Enter на экране прорисовывается ответ (если задача не была решена, символ "отказ"). Сама задача сохраняется в разделе корневого меню задачника "Буфер" - "Последние задачи". Ее можно либо удалить вместе с буфером, нажав клавишу "Shift-о", либо перенести в другой раздел задачника. При этом используются обычные средства работы с оглавлениями, о которых речь пойдет ниже.

4. Оглавление теорем (б)

   Через данный пункт осуществляется вход в оглавление теорем логической системы. Эти теоремы представляют собой исходный материал для создания приемов решателя задач. При переходе к приему теорема обычно преобразуется к виду, удобному для компиляции. При этом она может утратить логическую корректность, но основанный на ней прием в разумных контекстах будет работать безошибочно. Условно теоремы системы можно разделить на два класса: "базисные" теоремы из учебников и "производные" теоремы, по которым создаются приемы. В явном виде такое разделение нигде не зафиксировано.

Большинство теорем распределены по так называемым ячейкам логического вывода - концевым меню оглавления базы теорем, в которых первый пункт содержит именно базисную теорему, а остальные - следствия ее и некоторых других теорем. Среди таких следствий могут встречаться как базисные теоремы, так и производные. Распознать концевое меню, представляющее собой ячейку логического вывода, можно нажатием "Ctr-и" после выделения его первого пункта. В случае ячейки появляется некоторый непустой текст, иначе - пустой экран с курсором в левом верхнем углу. В обоих случаях выход из проверки - нажатием Esc.

После нажатия в главном меню клавиши "б" возникает некоторый раздел оглавления теорем. Чтобы попасть в корневое меню оглавления, несколько раз нажимается "курсор влево". Здесь приведен список разделов, в которых сформированы теоремы - вплоть до раздела "Словарь", начиная с которого идут уже данные технического характера. В основном, группировка теорем придерживается "предметного" принципа - по ключевому понятию, встречающемуся в теореме.

В каждом концевом пункте оглавления теорем могут размещаться несколько теорем, хотя обычно - единственная теорема. Если теорем несколько, то переходы между ними выполняются клавишами "курсор вверх" - "курсор вниз".

При входе в концевой пункт оглавления в верхней части экрана оказывается прорисована теорема. Под ней проведена горизонтальная черта, и далее - список характеристик теоремы. Под этим списком тоже проведена горизонтальная черта. Если она зеленая, то процедура логического вывода уже обучена "открывать" эту теорему как следствие первой теоремы ячейки вывода, если черная - пока не обучена.

5. Оглавление приемов (г)

Оглавление приемов относится лишь к тем приемам, которые заданы на языке ГЕНОЛО́Г, как теорема, снабженная алгоритмизирующей разметкой. Нажатие клавиши "г" переводит в некоторый раздел оглавления. Нажимая несколько раз клавишу "курсор влево", от него следует перейти к корневому меню. Последние разделы "Архив" и "Буфер" имеют технический характер. В них регистрируются автоматически сгенерированные, добавленные, измененные и удаленные приемы. Как и теоремы, приемы сгруппированы сначала по разделам, а затем, в основном, по "предметному" принципу.

При входе в концевой пункт оглавления на экране прорисовывается описание приема в виде теоремы, снабженной некоторой дополнительной разметкой, позволяющей компилятору создать исполняемую программу приема на языке ЛОС. Элементы этой разметки отделены от теоремы и друг от друга горизонтальными линиями. Подробнее о них будет рассказано в главе, посвященной языку ГЕНОЛО́Г.

В одном и том же концевом пункте оглавления может располагаться несколько приемов. Переходы между ними осуществляются клавишами "курсор вверх" - "курсор вниз".

Если с приемом связан его источник - теорема из базы теорем, то нажатие клавиши "Ctr-F9" переведет в просмотр этого источника. Для возвращения в базу приемов нужно нажать, находясь в просмотре теоремы, клавишу "п", затем при помощи "курсор вверх" - "курсор вниз" выбрать нукжный прием и нажать Enter.

Как уже отмечалось, теорема приема обычно отличается от теоремы - отбрасываются антецеденты, заведомо истинные в разумных контекстах, добавляются вспомогательные обозначения, создаются антецеденты для обращения к тем или иным процедурам, и т.п. Иными словами, теорема приема - это уже элемент языка программирования ГЕНОЛОГ, а не логически корректная теорема.

Чтобы перейти от просмотра описания приема на языке ГЕНОЛОГ к его испольняемой программе на языке ЛОС, нажимается Home. Возвращение к данному описанию, из любой точки ЛОС-программы, осуществляется нажатием End.

6. Оглавление программ (л)

   Оглавление программ позволяет быстро находить различные фрагменты ЛОС-программ решателя и переходить к просмотру этих фрагментов. Переход по клавише "курсор вправо" на выбранном концевом пункте оглавления переводит в просмотр фрагмента ЛОС-программы. При этом голубым цветом будет выделен оператор вида "прием(N)" - контрольная точка, созданная в программе для создания ссылки из оглавления программ. Чтобы вернуться в тот же пункт оглавления программ, достаточно нажать "End".

   Заметим, что оглавление программ не затрагивает приемы решателя, заданные на ГЕНОЛОГе. На его долю остается лишь малая часть приемов, реализованных непосредственно на ЛОСе, а также разнообразные процедуры технического характера - интефейсы, редакторы программ, компиляторы и т.п.

   В концевых пунктах оглавления программ содержатся краткие пояснения действий соответствующего участка ЛОС-программы. Участки, относящиеся к одной и той же программе или ее блоку, собраны в подраздел оглавления. Таким образом, фактически оглавление программ - это оглавление комментариев к программам, что-то вроде древовидной архитектуры этих программ, "нависающей" над ними как своеобразные строительные леса. Оно упрощает отладку и развитие программ ЛОСа.

   Перечислим основные разделы корневого меню оглавления программ:

   (a) Приемы решателя. Здесь собраны те приемы решателя задач, которые реализованы непосредственно на ЛОСе, без участия ГЕНОЛОГа. Их доля в общей массе приемов решателя невелика. Как правило, это приемы общелогического характера, не связанные с конкретными предметными областями.

   (b) Редактор ЛОСа. Программы интерфейса редактора приемов ЛОСа и его вспомогательных операторов.

   (c) Отладчик ЛОСа. Программы отладчика ЛОСа, позволяющие устанавливать прерывания при отладке и анализировать текущий кадр отладки.

(d) Редактор ГЕНОЛОГа. Программы просмотра приемов ГЕНОЛОГа и их редактирования.

(e) Компилятор ГЕНОЛОГа. Программы компилятора ГЕНОЛОГа, преобразующего описание приема на ГЕНОЛОГе (т.е. теорему, снабженную указателями ее алгоритмизации) в исполняемую программу на ЛОСе.

(f) База теорем. Программы, позволяющие просматривать теоремы, добавлять, удалять и изменять их. Программы, сопровождающие теоремы характеристиками, необходимыми для создания приемов. Программы, реализующие логический вывод в базе теорем, ориентированный на создание новых приемов. Программы спецификатора, определяющие по теореме и ее характеристике список возможных целей и способов применения теоремы (спецификаций приемов).

(g) Синтез приемов. В этом разделе содержатся:

i. Оглавление типов приемов. Фактически, это оглавление языка "Логический ассемблер". Каждый концевой пункт данной ветви оглавления программ прикреплен к программе, инициирующей создание описание приема на ГЕНОЛОГе по его типу и сопровождающим тип данным (т.е. по спецификации приема).

ii. Программа, позволяющая просмотреть все приемы заданного типа, имеющиеся в системе. Она запускается из просмотра концевогоо пункта оглавления типов приемов нажатием клавиши "л".

iii. Программа, обновляющая список ссылок на приемы заданных типов путем полного сканирования базы приемов.

iv. Программа компилятора спецификаций, создающая описание приема на ГЕНОЛОГе по его спецификации. Обращение к ней происходит из программ, связанных с концевыми пунктами оглавления типов приемов.

v. Программы интерфейсов, позволяющих протестировать создание спецификаций по текущей теоремы из базы теоремы, синтезировать приемы ГЕНОЛОГа по отдельным таким спецификациям и скомпилировать их на ЛОС. Результаты сохраняются в буфере базы приемов.

vi. Программы полного цикла генератора приемов. Этот генератор запускается из просмотра некоторой теоремы, хранящейся в базе теорем. Предпринимается обращение к процедуре вывода следствий данной теоремы, представляющих интерес для создания приемов. Для каждой полученной теоремы создаются спецификации приемов, они компилируются в описания приемов ГЕНОЛОГа, те - в программы на ЛОСе. Для каждого такого приема генерируется тестовый пример, проверяющий, что созданный прием не избыточен - не сводится к работе уже имевшихся приемов. Прошедшие отбор приемы подвергаются доводке по задачнику - происходит одна или несколько полных прокруток цикла решения его задач, в процессе которых выявляются и корректируются новые приемы, портящие решение "старых" задач. Наконец, все прошедшие тестирование приемы регистрируются в разделе "Архив" оглавления базы приемов, где они распределяются по результатам доводки.

Заметим, что прокрутки по задачнику и циклы коррекции приемов чрезвычайно трудоемки. Чтобы они выполнялись за разумное время (не свыше одного - двух десятков минут), приходится прибегать к распараллеливанию вычислений. Кратность распараллеливания можно выбирать произвольную. Впрочем, даже на 24-поточном процессоре вычисления ускоряются отнюдь не в 24 раза. Поэтому предусмотрена параллельная прокрутка на 2-х или 3-х компьютерах, соединенных в локальную сеть. При распараллеливании запускаются совершенно независимые полные копии программы логической системы, которым передаются различные задания. По завершении их работы результаты передаются в "основную" копию, где объединяются.

(h) Анализатор решений. Решатель задач показывает по шагам лишь процесс поиска решения. Он может содержать множество лишних действий. Чтобы отбросить их и показать лишь то, что нужно, создана программа анализатора решений. Она сохраняет протокол процесса поиска решения и обрабатывает его, удаляя шаги, не использованные в цепочке логических переходов от постановки задачи к ответу. Работа над анализатором решений находится в самом начале, и по существу он ориентирован пока только на планиметрические задачи. В данном разделе оглавления программ представлены прграммы, обеспечивающие составление протокола решения задачи, интерфейс его просмотра и простейшую оптимизацию.

(i) Интерфейс оглавлений. Для работы с приемами, теоремами, программами и многим другим используется одна и та же процедура "оглавление". Она позволяет не только просматривать и изменять указанные разделы, но и запускать различные процедуры, связанные с ними. По существу, оглавления - универсальный "пульт управления", используемый в логической системе.

В разделе содержатся ссылки на блоки программы, реализующие основные действия интерфейса оглавлений. Впрочем, часть таких действий вынесена в другие разделы оглавления программ, более для них подходящие.

(j) Интерфейс просмотра и редактирования задач. В связи с задачами созданы два почти независимых друг от друга интерфейса.

Первый из них (оператор "списокзадач") предназначен для просмотра "ленты задач" концевого раздела задачника. При переходе от любого концевого пункта данного раздела по клавише "курсор вправо" в верхней части экрана появляется прорисовка соответствующей задачи. При помощи вертикальных курсоров можно переходить к просмотру других задач раздела - над текущей задачей либо под ней. В разделе представлены пункты, обеспечивающие интерфейс просмотра и редактирования ленты задач, а также запуск решения выбранной задачи.

Второй интерфейс (оператор "цепьзадач") предназначен для просмотра аналогичным образом устроенной ленты задач, отражающей текущий момент решения выбранной задачи. Самая верхняя задача этой ленты - та, которая была выбрана в задачнике для решения. Ниже идет ее подзадача, далее - подзадача подзадачи, и т.д. вплоть до текущей решаемой задачи. Под текущей задачей обычно прорисовывается описание текущего

действия, выполяемого системой. Иногда под таким описанием размещаются кадры обращений к вспомогательным задачам, решенным для выполнения действия. При необходимости можно по шагам просматривать ход решения таких вспомогательных задач. В разделе представлены пункты, обеспечивающие интерфейс просмотра текущей ленты задач ("цепи задач"), выход в отладчик ЛОСа и установку прерываний.

(k) Интерфейс просмотра и редактирования информационных блоков. Программы "технического" интерфейса, позволяющего вручную редактировать непрограммные файлы логической системы на нижнем уровне. Обычно используется для исправления испорченных данных.

(l) Общий интерфейс. Программы, обслуживающие функционирование главного меню системы, устанавливающие различные общие параметры, редактирующие шрифты и словари системы, и т.п.

(m) Вспомогательные процедуры интерфейсов. Программы формульного, текстового, текстформульного и геометрического редакторов. Просмотр древовидных структур данных ЛОСа, программы блоков диалога, программы для работы с графиками и битмэпами. Программы для прорисовки шахматных позиций. И т.п. Все это - вспомогательные процедуры, обращение к которым возможно из самых различных точек системы.

(n) Текстовый анализатор. Программы основных блоков текстового анализатора: чтение фразы, морфологический разбор, синтаксический разбор, отождествление объектов и действий, упоминаемых в тексте, создание логического подстрочника фразы, обращение к задачам, анализирующим логический подстрочник.

(o) Интерфейс упрощенного ввода задачи. Программы, восстанавливающие полное описание задачи по ее типу и сокращенным входным данным.

7. Ресурсы и установки (p)

Вход в подменю главного меню, позволяющее выполнять следующие действия:

(a) Просматривать логические символы системы, упорядоченные по номерам.

(b) Просматривать логические символы системы, упорядоченные по алфавиту.

(c) Вводить названия новых символов, удалять либо изменять старые названия.

(d) Определять название логического символа по его номеру либо, наборот, номер символа (в том числе шестнадцатиричный) по его названию.

(e) Пополнять словарь текстого анализатора (корни слов, окончания, суффиксы и их кодирование логическими символами).

(f) Выбирать точку входа при запуске системы (главное меню либо задачник либо справочник по системе)

(g) Выбирать размеры окна системы.

(h) Просматривать и редактировать информационные блоки системы на нижнем уровне (обычно используется для исправления этих блоков).

(i) Редактировать контекстные меню, извлекаемые по F1.

(j) Изменять шрифты системы.

(k) Выбирать параметры распараллеливания при прокрутках.

8. Уплотнение измененных файлов (у). При изменениях в программных и информационных файлах системы старая версия фрагмента помечается как неиспользуемая, а новая версия этого фрагмента дописывается в конце файла, причем ссылка на нее происходит через старую версию. Данный пункт интерфейса запускает процедуру исключения удаленных старых версий путем полного переписывания содержимого всех измененных файлов системы и необходимой переадресации. Рекомендуется прибегать к нему почаще. Реальная практика - перед каждым выходом из системы.

9. Сохранение копий файлов (х). Чтобы можно было восстановить испорченную версию системы, необходимо регулярно сбрасывать ее текущее состояние в поддиректорию COPY. Это и происходит при выборе данного пункта главного меню. Предварительно проверяется целостность версии. Если она нарушена, копирование не происходит, а реализуется немедленный выход из системы.

   Рекомендуется почаще уплотнять измененные файлы и сразу же сохранять их копии. Так как система хранится в сжатом виде, ремонт ее испорченных файлов оказывается чрезвычайно сложной задачей. На практике здесь выручает только извлечение предыдущей версии из директории COPY. Кстати, это можно делать, не выходя из системы. Достаточно нажать из главного меню "Ctr-Backspace".

10. Просмотр программы логического символа (п).

    После нажатия клавиши "п" в обведенном синей рамкой окне появляется курсор текстового редактора. Вводится название логического символа, нажимается Enter, и происходит переход в просмотр корневого фрагмента ЛОС-программы данного символа. Для пробы можно ввести символ "равно". Далее клавишами курсоров можно перемещаться по всему дереву фрагментов данной программы: курсоры вправо-влево обеспечивают выбор нужного перехода к подфрагменту (выделен желтым цветом); курсор вниз - вход в подфрагмент, курсор вверх - возвращение в надфрагмент. Кроме просмотра программы, можно ее редактировать.

11. Директория FLS (д). Чтобы логическая система имела доступ к "чужому" файлу произвольного формата, этот файл следует поместить в поддиректорию FLS директории системы. После нажатия "д" появляется оглавление поддиректории FLS. При входе в концевой пункт оглавления файл прорисовывается в шестнадцатиричном либо текстовом форматах. При необходимости его можно редактировать. Пока директория FLS использовалась только для ручной коррекции испорченных двоичных файлов системы.

12. Операторы ЛОСа (о). Переход к справочному оглавлению, в котором представлены операторы ЛОСа - как реализуемые интерпретатором, так и запрограммированные на самом ЛОСе. От просмотра описания оператора нажатием Home можно перейти к его ЛОС-программе. Возвращение - по нажатии End.

В заключение раздела заметим, что ту часть логической системы, которая непосредственно отвечает за решение задач (т.е. базу приемов, реализованных на ЛОСе либо на ГЕНОЛОГе) будем иногда называть решателем.

1.4 Предварительное знакомство с элементами интерфейса

Приведем здесь краткое описание тех элементов интерфейса системы, которые понадобятся в первую очередь.

Оглавления

Одним из наиболее часто используемых элементов интерфейса логической системы являются ее оглавления. Простейшие действия с ними уже рассматривались в предыдущем разделе. Оглавления служат не только для поиска нужных сведений, но и для запуска различных процессов логической системы, являясь своего рода "пультом управления". Подробнее об этих их функциях будет говориться в различных разделах книги. Здесь же ограничимся описанием лишь самых общих действий.

Как и обычно, оглавление имеет древовидную структуру и состоит из отдельных меню. Пункты меню пронумерованы. Если после номера пункта идет круглая скобка, то данный пункт представляет собой подменю, если идет точка - концевой пункт.

Текущий пункт оглавления выделен синим цветом. Для его выбора моожно использовать клавиши курсора "вверх - вниз" либо левую кнопку мыши.

Если меню не помещается на экране целиком, для его ускоренной прокрутки можно использовать клавиши PageUp, PageDown. Кроме того, прокрутку можно выполнять и вертикальными клавишами курсора.

Для входа в текущий пункт оглавления (подменю либо концевой пункт) служит клавиша "курсор вправо" либо "Enter". Для первых девяти пунктов меню можно также использовать цифровые клавиши.

Чтобы вернуться в надменю либо выйти из просмотра содержимого концевого пункта оглавления обратно в меню, служит клавиша "курсор влево".

Чтобы изменить название текущего пункта оглавления, используется клавиша "р". Она переводит в текстовый редактор. По завершении ввода нового названия нажимается Enter.

Чтобы добавить концевой пункт оглавления в конце меню, нажимается клавиша "к". Если такой пункт нужно добавить перед текущим пунктом оглавления, нажимается "К".

Чтобы добавить подменю в конце меню, нажимается "м". Если подменю нужно добавить перед текущим пунктом оглавления, нажимается "М".

Чтобы удалить пустое текущее подменю либо пустой текущий концевой пункт, нажимается Ctr-Del. Если подменю либо концевой пункт непустые, их нужно сначала расчистить.

Для быстрого возвращения из оглавления в главное меню нажимается End. Обычно при этом запоминается текущее место в оглавлении, так что при следующем обращении к оглавлению оно сразу восстанавливается. Однако, если текущим местом было подменю, при возвращении произойдет переход к первому пункту данного подменю.

Текстовый редактор

Текстовый редактор, используемый системой, несколько отличается от стандартного. Это объясняется тем, что первые версии программы создавались еще в период появления первых персональных компьютеров, и все элементы интерфейсов делались "с нуля". Постепенно они оптимизировались для обучения логической системы, так что даже текстовый редактор получил множество дополнительных функций. При некотором привыкании к его командам он ничуть не уступает стандартному, а иногда и удобнее стандартного.

Разумеется, символы текста вводятся обычным образом. При входе в текстовый редактор автоматически устанавливается кириллица. Если в процессе набора нужно перейти к латинице, нажимается F12. Для возвращения в кириллицу служит F11.

По умолчанию, режим вставки отключен: новый символ заменяет тот, который был до этого на позиции курсора.

Ввиду чисто технического характера текстов, для ввода которых служит текстовый редактор, форматирование не используется. При достижении правого края рамки ввод текста автоматически продолжается с новой строки, без каких-либо указателей переноса. Форматирование введено в текст-формульном редакторе, позволяющем чередовать набор текстов и формул. Текстовый редактор служит его фрагментом.

Для завершения редактирования нажимается клавиша Enter. Для отмены - Esc.

Справочная информация о текстовом редакторе может быть получена через справочник по системе. Его корневое меню имеет раздел "Текстовый редактор".

Приведем специальные команды текстового редактора:

1. Чтобы ускорить перемещение курсора, предусмотрен режим "больших шагов". Для перехода в него нажимается клавиша Tab. Она же выводит из этого режима.

2. Для перевода курсора в начало строки нажимается Home, для перевода в конец строки - End.

3. Для удаления фрагмента текста курсор сначала переводится на начало этого фрагмента и нажимается Delete. На этой позиции остается красный маркер (красный фон символа). Затем курсор перемещается на конец фрагмента и снова нажимается Delete. Если после выделения начала фрагмента нужно отменить операцию, нажимается "Backcpace".

4. Для перехода в режим вставки текста начиная с позиции курсора (т.е. перед символом, на котором он расположен) нажимается Insert. В режиме вставки курсор приобретает голубой цвет. По окончании вставки снова нажимается Insert. Перемещение курсора на другие позиции, если он находится в режиме выставки, заблокировано.

На практике оказалось удобнее вставку выполнять по-другому. Для этого сначала нажимается F7 (один либо несколько раз), после чего текст, ранее располагавшийся с позиции курсора, смещается вниз на столько строк, сколько раз была нажата клавиша F7. Для вставки высвобождается пустое пространство, и она выполняется без перехода в режим "Insert". Если вставлялся кусок программы ЛОСа, то по завершении редактирования программы нажатием Enter все пробелы автоматически устраняются. Иначе их можно устранить вручную (см. предыдущий пункт).

5. Если нужно фрагмент текста перенести на другое место, то курсор переводится на начало этого фрагмента и нажимается F2. На этом месте остается зеленый маркер (цвет фона). Далее курсор переводится на конец фрагмента и снова нажимается F2 - появляется аналогичный маркер. Наконец, курсор переводится на ту позицию, начиная с которой должен быть расположен фрагмент. После нажатия F2 происходит удаление фрагмента на старом месте и перенесение его на новое место. Чтобы отменить начатую операцию, используется клавиша Backspace.

6. Если нужно вставить чистую строку начиная с позиции курсора, нажимается F7. При этом все, что идет после этой позиции, сдвигается вниз на одну строку.

7. Если нужно удалить строку, начиная с позиции курсора, аналогичным образом нажимается F6. На практике эта операция никогда не используется - удобнее применять режим Delete. Случайное нажатие F6 может испортить текст. Чтобы не потерять его, в этой ситуации следует выйти из режима редактирования по Esc. Предварительно можно сохранить только что набранную часть в буфере, а при возвращении в редактирование - восстановить ее.

8. Чтобы сохранить в буфере фрагмент текста, курсор подводится к началу фрагмента и нажимается PageDown. На этом месте остается темно-синий маркер. Затем курсор переводится на последнюю позицию фрагмента и снова нажимается PageDown.

9. Чтобы извлечь из буфера фрагмент текста и вставить его с нужной позиции, курсор переводится на эту позицию и нажимается PageUp. Вставку фрагмента из буфера можно проводить многократно.

10. Предусмотрена возможность сопоставления друг с другом открывающих и закрывающих скобок текста. Чтобы найти скобку, парную к той, на которой расположен курсор, нажимается F1 (увы, нестандартное, но ставшее очень привычным использование клавиши).

11. Наконец, для просмотра информации о клавиатуре текстового редактора (если уже идет набор текста) нажимается F3.

1.5 Распараллеливание работы системы

При первом чтении этот раздел можно пропустить.

Если добавляется один или несколько приемов решения задач, нужна проверка того, что эти приемы не нарушают решения задач, хранящихся в задачнике системы

(приводят к отказу, изменяют ответ или существенно увеличивают время решения). Аналогично, при добавлении нового приема вывода теорем нужна проверка того, что этот прием не нарушает ранее проработанных цепочек вывода в базе теорем и не приводит к переполнению ее разделов. Эти проверки выделяют особые точки, позволяющие скорректировать прием. Без них система очень быстро разрегулировалась бы и перестала делать многое из того, чему уже обучена.

Для указанной проверки предпринимается цикл решения задач из заданных разделов задачника либо вообще всего задачника. В случае базы теорем - цикл полного вывода для всех ее разделов либо для всей базы теорем. Такую прокрутку можно было бы проводить в последовательном режиме, но тогда она будет крайне длительной - занимать часы. Поэтому прокрутка распараллеливается: запускаются независимые копии логической системы, между которыми заблаговременно распределяются диапазоны прокрутки. По окончании работы копии завершают свою работу, а головная система собирает из их файлов те данные, которые они получили при прокрутке.

Число копий ограничивается возможностями машины. Например, оно может быть равно максимальному числу потоков, реализуемых ее процессором. Возможно распараллеливание на 2 или 3 машинах, соединенных в сеть. Сейчас для развития системы прокрутка выполняется на 3 машинах, причем число копий логической системы на головной машине равно 24, а на двух побочных - 20. Впрочем, прокрутка столь большим числом потоков на самом деле не критично отличается от 16 или даже 8 потоков - узким местом становится размер кэша процессора.

Если нужно использовать 2 или 3 машины, то головная машина должна получить имя "R", дополнительные - "L" и "M". Это делается через параметры операционной системы windows.

Чтобы пользоваться параллельной прокруткой, необходимо предварительно подготовить директорию, в которой расположены основные папки GEN, INF, LOS, TCH, TER, TXT логической системы и ее исполняемый файл *logsyst.exe*. В зависимости от числа m потоков, которые может выполнять процессор, нужно создать в этой директории папки $EX1$, $EX2$, ..., EXm. В любом случае, параметр m не должен превосходить 23. Если он больше 9, то после $EX9$ идет EXa, затем EXb, и т.д. В каждую из этих папок нужно скопировать указанные выше шесть папок логической системы и исполняемый файл, которой нужно переименовать. Вместо *logsyst.exe* в папку EXi помещается его копия *logsysi.exe*.

На каждой из дополнительных машин должна быть создана директория *logsyst*, копирующая ту директорию головной машины, в которой находится логическая система. Через сеть нужно открыть прямой доступ к этой директории и предоставить для нее права "ВСЕ" на чтение и запись. Такие же права не помешает установить и для головной машины.

Для облегчения работы с копиями системы нужно установить на компьютер (основной и побочные, если они есть) систему AUTOIT. Она бесплатно скачивается в интернете. Эта система позволяет создавать скрипты, взаимодействующие с операционной системой windows.

Чтобы автоматически пересылать исполняемый файл системы по указанным папкам EXi после каждого его изменения, создан AUTOIT - скрипт "copir". Если работа ведется одновременно на трех системных блоках - основном ("R") и дополнительных ("L", "M"), то этот скрипт выполняет пересылку и на дополнительные блоки.

Создан еще один полезный AUTOIT - скрипт "fenix". Он немедленно запускает любой из параллельных процессов, как только операционная система останавливает его и выдает сообщение об ошибке. При перезапуске система пропускает ту ячейку логического вывода, на которой произошел сбой, но запоминает эту точку и впоследствии укажет на нее в табло результатов. При отсутствии скрипта можно выполнять перезапуск вручную. Остались только сбои, которые трудно отследить в последовательном режиме: при повторном запуске система их не обнаруживает.

Заметим, что система сигнализирует о начале своей работы размещением файла $vkl.los$ в текущей ее директории. Пока он имеется, $fenix$ будет немедленно повторно запускать систему в случае любого ее сбоя. По окончании работы логическая система удаляет этот файл, заменяя его на файл $fin.los$. Головная версия системы собирает итоговые данные лишь после того, как на всех побочных версиях окажется $fin.los$.

После того, как указанная подготовка директории системы проведена, следует в главном меню системы зайти в пункт "Ресурсы и установки" и установить в окне "Распараллеливание" нужное число потоков m. Еще раз напоминаем, что оно не должно превосходить 24. Для входа в редактирование числа потоков нужно нажать на него левой кнопкой мыши, затем нажать нужную цифровую клавишу и Enter.

Кроме того, если используются несколько машин, нужно найти в оглавлении программ пункт "Общий интерфейс" - "указание числа потоков для побочных машин", выйти через него в ЛОС-программу, и в операторе "равно(x10 набор(вариант(равно(x6 1)десзапись(набор(2 0)) десзапись(набор(2 0))))0 0))" поменять первую "десзапись(...)" на "десзапись(l)", вторую - на "десзапись(m)", где l - число потоков машины L, m - число потоков машины M. Для цифры ставится она сама, для числа из двух цифр $c1, c2$ - выражение "набор($c1$ $c2$)".

При запуске прокрутки окно головной системы уменьшается до размеров небольшого прямоугольника, размещенного в правом верхем углу экрана и имеющего название $EX0$. Окна дополнительных потоков имеют такие же размеры.

В окне параллельного потока при прокрутке будут прорисованы: в левой части - трудоемкость вывода в текущей ячейке вывода, в правой части - название потока EXi. Трудоемкости меняются в окне по ходу работы, и таким образом можно проверять, что поток не завис.

По мере выполнения работы, дополнительные потоки исчезают. Когда работу завершает основной поток, он перекрашивается в зеленый цвет. Возвращение в обычный формат его экрана произойдет автоматически после завершения работы всех потоков. Если при этом система обнаружит, что какие-то выводы были прерваны, она прежде всего попытается их повторить, так что в итоговом табло будут отображены лишь те прерванные выводы, которые при повторной попытке снова были прерваны. Обычно это означает резко возросшую трудоемкость вывода в ячейке и требует специального анализа.

Если какой-либо из потоков явно завис, его нужно остановить обычным образом и сразу же заново запустить. Чтобы это было возможно, полезно вывести на рабочий стол ярлыки всех версий EXi, причем сделать это так, чтобы они не перекрывались окнами потоков. В левой части экрана места для этого хватает.

Если потоки остановлены через диспетчер задач, то для восстановления работоспособности системы нужно запустить главный поток - logsyst.exe и сразу же нажать "Break", "Ctr-з", "Esc", "з", "Ctr-ы". Затем закрыть программу и снова зайти в нее.

Глава 2

Логический язык

В системе используется логический язык открытого типа, пополняемый в процессе обучения. Для каждого нового понятия вводятся свои правила образования с его помощью корректных синтаксических конструкций и свои соглашения о их смысловой интерпретации, учитываемые при создании приемов, использующих данное понятие. При этом соблюдаются некоторые простейшие общие требования, которые будут изложены ниже.

Ниже речь идет лишь о внутреннем представлении утверждений выражений в системе; для диалога с пользователем применяется другая запись, приближенная к стандартной математической, которая автоматически транслируется во внутреннее представление. Специальная программа формульного редактора переводит внутреннюю запись утверждений и выражений в привычную математическую запись. Она же позволяет быстро вводить формулы в обычной записи и переводить их во внутренний логический формат системы.

2.1 Алфавит логического языка. Логические символы и символы переменных

Алфавит языка состоит из элементов двух типов - логических символов и символов переменных. Логические символы обозначают конкретные понятия (отношения между объектами, операции над ними, имена объектов, логические связки, кванторы и т.д.); переменные служат для обозначаения варьируемых объектов. Символы каждого типа пронумерованы последовательными натуральными числами. В компьютерной реализации для этих номеров зарезервирован диапазон от 1 до $2^{19} - 1$. Логические символы, по существу, отождествлены со своими номерами, и в этом смысле все они изначально имеются в алфавите логического языка. Однако, используемыми на текущий момент считаются только те из них, для которых введено название - слово либо словосочетание, имеющее не более 24 букв либо цифр. Такие названия хранятся в специальном файле и легко могут быть изменены. При решении задач эти названия не используются - они нужны только для отображения логических символов на экране в различных диалогах. В настоящее время используется порядка 8000 логических символов.

Примерами логических символов могут служить символы "равно", "и", "или", "длялюбого", "плюс", "меньше", "прямая", и т.п. Чтобы просмотреть список используемых логических символов, упорядоченных по номерам, достаточно из главного меню

нажать "с". Появляется первая страница списка - два столбца названий символов. Выделять нужный символ можно при помощи клавиш курсора. Чтобы просмотреть справочную информацию о выделенном (он прорисовывается в голубом цвете) символе, достаточно нажать "и". Для возвращения в список символов нажимается "пробел". Для перехода к следующей либо предыдущей странице списка служат клавиши "Page Up" и "Page Down". Возвращение в главное меню - по Esc.

Для просмотра списка логических символов, упорядоченных в алфавитном порядке, нужно из главного меню нажать "а" (кириллица).

Переменные, как и логические символы, пронумерованы. Переменная с номером i прорисовывается текстовым редактором как "xi". Первая буква здеь - кириллица, так как использование латинской версии в текстовом редакторе потребовало бы постоянной смены режимов "кириллица - латиница". При прорисовке в стандартной математической записи используются малые и большие латинские буквы, быть может, с индексом: a, b, A, B, D_1, и т.п. При этом нумерация фиксирована: a - x1, b - x2, c - x3, и т.д. Для удобства чтения в тех случаях, когда в одном контексте используется и текстовая, и формульная версии обозначения переменной, прорисовывается специальный переходник, указывающий соответствие обозначений. Хотя современная версия интерпретатора позволяет работать с $2^{19} - 1$ различными символами переменных, ряд операторов языка программирования ЛОС реализован так, что поддерживает лишь 512 переменных. Это связано с применением двоичных масок для задания множеств переменных - чтобы число ячеек, определяющих такие множества, было не слишком большим. Впрочем, на практике 512 переменных до сих пор было достаточно - редко случается в одном процессе встретить более 100 переменных.

Алфавитный список логических символов и список этих символов, упорядоченных по номерам, не дает представления о распределении символов по различным разделам. Чтобы найти список логических символов, относящихся к некоторому разделу, нужно зайти через главное меню в оглавление приемов ГЕНОЛОГа и выбрать нужный раздел. В нем выбрать подпункт "Справочники", а в этом подпункте - войти в подпункт "Содержание". В верхней части экрана будет прорисована запись вида "содержание(A $S_1 \ldots S_n$)", где A - название раздела; $S_1, \ldots, S_n$ - относящиеся к разделу логические символы либо названия его подразделов. Например, "содержание(элементарнаяалгебра 0 1 2 3 4 5 6 7 8 9 величина число Число минус плюс умножение дробь неравенства модуль сигнум Сигнум целыечисла степени логарифм e тригонометрия гипфункции комбинаторныефункции многочлены)". Здесь "неравенства", "тригонометрия", "целыечисла", "комбинаторныефункции", "многочлены" - названия подразделов. В каждом из таких подразделов имеется свой пункт "Содержание", перечисляющий его символы.

Чтобы ввести новый логический символ, нужно зайти в раздел "Ресурсы и установки" главного меню. В нем - выбрать пункт "Ввод названия нового символа (н)" и войти в данный пункт. Далее текстовым редактором набирается название символа - произвольная последовательность букв и цифр, длина которой не превосходит 24. Маленькие и большие буквы различаются, так что "плюс" и "Плюс" - различные символы. Система находит первый неиспользуемый номер логического символа и присваивает ему введенное название.

Если нужно изменить название уже имеющегося логического символа (не изменяя его номер), в разделе "Ресурсы и установки" выбирается пункт "Изменение назва-

ния символа (Ctr-н)". Сначала вводится название, которое требуется изменить. Его следует вводить осторожно, так как при вводе несуществующего названия система зацикливается, и придется из нее выходить. Впрочем, никаких последствий это иметь не будет. После ввода названия нажимается Enter, и вводится новое название. Если оно уже использовалось, то система об этом сообщит. Придется здесь же вводить другое. Заметим, что к моменту ввода нового названия старое уже удалено.

Чтобы удалить ненужный логический символ (фактически - удалить его название из файла названий), следует в разделе "Ресурсы и установки" выбрать пункт "Удаление названия символа (Ctr-у)". Затем ввести удаляемое название и нажать Enter. Заметим, что удалять логический символ следует только тогда, когда есть уверенность, что все ссылки на него в логической системе уже удалены.

Чтобы получить номер логического символа по его названию, следует в разделе "Ресурсы и установки" выбрать пункт "Определение номера символа по его названию (с)". Далее ввести символ и нажать Enter. Появится номер символа - как десятичный, так и шестнадцатеричный.

Чтобы получить название логического символа по его номеру, не обязательно просматривать список символов, упорядоченных по номерам. Вместо этого достаточно выбрать пункт "Определение названия символа по его номеру (Ctr-с)" раздела "Ресурсы и установки", ввести номер символа и нажать Enter.

Упражнения:

1. Ввести новый логический символ "ppp". Убедиться по алфавитному списку символов, что он появился.

2. Изменить название "ppp" на "ttt". Убедиться, что в списке символов это отражено.

3. Удалить символ "ttt". Убедиться, что в списке символов его больше нет.

4. Найти номер символа "умножение".

5. Найти символ с номером 523.

2.2 Термы, утверждения и выражения

Логический язык образован словами некоторого специального вида, построенными при помощи логических символов, символов переменных и скобок. В языке рассматриваются слова двух типов - утверждения и выражения. Выражения используются для обозначения объектов; утверждения - для формулировки свойств объектов и отношений между ними. Как выражения, так и утверждения представляют собой записи со скобками (термы), определяемые следующим индуктивным образом:

1. Каждое однобуквенное слово, состоящее из логического символа либо символа переменной, является термом.

2. Если $t_1, \ldots, t_n$ - термы; $n \geq 1$; f - логический символ, то слово $f(t_1 \ldots t_n)$ есть терм.

Никаких дополнительных ограничений на термы не накладывается, и в большинстве случаев термы совершенно бессмысленны. Помимо общего правила образования новых термов, с каждым используемым логическим символом f будут связываться свои дополнительные ограничения на элементы $t_1, \ldots, t_n$, при которых новый терм представляет собой осмысленное утверждение или выражение.

Эти ограничения вводятся в систему вручную, при добавлении нового логического символа. Чтобы посмотреть либо отредактировать их, нужно в пункте "просмотр программы логического символа" главного меню набрать название символа, нажать Enter, и затем нажать F3. Возникающий текст можно прокручивать вверх-вниз клавишами курсора либо перелистыванием страниц. Текст состоит из фрагментов, редактируемых независимо. Для выделения фрагмента служит левая кнопка мыши. Чтобы отредактировать фрагмент, нужно навести на него курсор мыши и нажать правую ее кнопку. Новые фрагменты добавляются нажатием Enter. Вставка происходит перед выделенным фрагментом, а если его не было, то в конце текста.

Другой способ выйти на те же самые сведения о правильном использовании логического символа - через указанный выше просмотр списка всех символов. Символ выделяется, и нажимается "и".

Из просмотра приема ГЕНОЛОГа можно получить информацию о символе, нажав "с" и введя этот символ. Наконец, из того же просмотра клавиша "Ctr-я" переводит в оглавление, где сведения о символах сгруппированы по разделам.

Если терм $f(t_1 \ldots t_n)$ представляет собой утверждение, то символ f называется предикатным символом; если этот терм - выражение, то символ f называется функциональным символом. Такой подход несколько отличается от принятого в математической логике, где логические связки и кванторы выделяются в отдельную категорию и предикатными символами не считаются. Нам будет удобно присоединить их к классу предикатных символов, чтобы упростить различение утверждений и выражений.

Заметим, что на практике часто используются суррогаты утверждений и выражений - некоторые их сокращения, однозначно понимаемые лишь в определенных контекстах. Например, утверждение $f(x) = O(g(x))$ имеет смысл лишь в контексте, где уточняется что-нибудь типа $x \to a$. Это явление мы будем называть контекстной семантикой и использовать его так же, как оно используется традиционным образом. Приемы решения задач будут понимать такого рода сокращения и работать с ними, не допуская ошибок.

Напомним, что предикатом в математической логике называется функция, принимающая значения "истина" и "ложь". Поэтому утверждения можно рассматривать как термы, задающие предикаты, а выражения - как термы, задающие "обычные" функции. Например, утверждением является терм "меньше(a, b)". Он определяет предикат, принимающий значение "истина", если значение переменной a меньше значения переменной b. С другой стороны, терм "плюс(a, b)" - выражение, определяющее функцию, значением которой служит сумма значений переменных a и b.

Каждый логический символ f характеризуется арностью - числом n операндов, для которого допускается строить выражение либо утверждение $f(t_1 \ldots t_n)$. В особых случаях число этих операндов может отличаться от арности символа - например, если символ обозначает двуместную ассоциативную операцию. Тогда число операндов допускается любым, большим единицы. Все такие ситуации явно оговариваются в справочной информации, регламентирующей использование символа.

Если внутри терма выделена некоторая позиция, отличная от позиции скобки, то говорим, что задано вхождение подтерма в этот терм. Позицию формально можно определять различными способами - например, указывая номер буквы в слове. При хранении терма в памяти компьютера вхождение будет задаваться адресом в памяти, начиная с которого расположен подтерм. Очевидным образом определяется результат замены вхождения подтерма в некоторый терм на другой подтерм.

Еще одно предварительное соглашение общего характера относится к понятиям свободной и связанной переменных терма. Среди логических символов выделены пять особых символов, называемых связывающими. Это символы кванторов "существует", "длялюбого", "Существует" (существует и единственен) и описатели "класс", "отображение", задающие множества и функции. Вхождение переменной x_i в терм θ называется связанным, если оно расположено внутри подтерма θ' терма θ, имеющего вид $f(x_1 \dots x_k\ t_1 \dots t_p)$, где f - связывающий символ; $x_1, \dots, x_k$ - переменные, $t_1, \dots, t_p$ - термы; $k \geq 0$; $p \geq 0$; $i \in \{1, \dots, k\}$. Список переменных $x_1 \dots x_k$ называется связывающей приставкой терма θ'. Вхождения переменных в терм, не являющиеся связанными, называются свободными. Переменная, имеющая хотя бы одно свободное вхождение в терм, называется его свободной переменной, или параметром.

Вообще говоря, переменная может иметь как свободные, так и связанные вхождения в терм. Впрочем, это нежелательно, и в таких случаях связанные переменные сразу же будут переобозначаться.

Часто бывает нужна операция подстановки термов $t_1, \dots, t_n$ вместо переменных x_1, $\dots, x_n$ в терм A. Она заключается в том, что все вхождения переменных x_i в терм A одновременно заменяются на соответствующие термы t_i. Результат применения такой операции обозначается $S^{x_1 \dots x_n}_{t_1 \dots t_n} A$. Впрочем, эту запись мы обычно не используем, а результат применения подстановки Q к терму T обозначаем попросту $Q(T)$.

Не всегда результатом применения подстановки к утверждению либо выражению служит осмысленный терм. Например, не стоит подставлять вместо переменной связывающей приставки квантора либо описателя какой-либо терм, отличный от переменной. Даже отождествлять подстановкой переменные связывающей приставки нельзя, так как получится терм, не имеющий "стандартного" вида квантора либо описателя. На практике мы будем использовать подстановки только вместо таких переменных, которые не имеют связанных вхождений в терм.

Естественно было бы ожидать, что при подстановке в терм A термов $t_1, \dots, t_n$ вместо свободных переменных $x_1, \dots, x_n$ будет происходить и подстановка функций, определяемых термами t_i, в функцию, определяемую термом A. Однако, это не всегда имеет место. Например, утверждение "$\exists_x(x < y)$" истинно для любого вещественного y. Однако, при подстановке x вместо y получается ложное утверждение "$\exists_x(x < x)$". Другой пример - выражение "$\mathrm{set}_x(x < y)$", определяющее открытый луч числовой прямой с концом в точке y, при подстановке x вместо y превращается в выражение "$\mathrm{set}_x(x < x)$", задающее пустое множество. В обоих случаях имела место подстановка терма t_i, для которого некоторое вхождение переменной x_i было расположено в области действия квантора либо описателя по переменной - параметру терма t_i. Оказывается, что если таких ситуаций избегать, то действительно подстановка термов будет приводить к соответствующей подстановке для их функций. Это приводит к понятию допустимой подстановки:

Подстановка выражений $t_1, \dots, t_n$ вместо переменных $x_1, \dots, x_n$ в терм A называется допустимой, если выполнены условия:

1. Никакая переменная x_i не имеет связанных вхождений в терм A; $i = 1, \ldots, n$.

2. Никакое вхождение переменной x_i в терм A не находится в области действия квантора либо описателя, некоторая связывающая переменная которого является параметром выражения t_i.

Иногда бывает необходимо найти такую подстановку S вместо переменных X, которая отождествляет термы $A_1, \ldots, A_n$ с термами $B_1, \ldots, B_n$: $S(A_1) = S(B_1), \ldots, S(A_n) = S(B_n)$. Например, эта ситуация складывается при последовательном применении двух теорем: заключение первой теоремы отождествляется с одной из посылок второй теоремы. Оказывается, что если отождествление вообще возможно, то существует "минимальная" отождествляющая подстановка T, т.е. такая, что любая отождествляющая подстановка S представима как последовательное применение подстановки T и некоторой другой подстановки. Эта минимальная подстановка называется подстановкой, унифицирующей термы $A_1, \ldots A_n$ с термами $B_1, \ldots, B_n$. Она единственная с точностью до переобозначения переменных в подставляемых выражениях.

Алгоритм нахождения унифицирующей подстановки несложен. Достаточно рассматривать систему равенств $A_1 = B_1, \ldots, A_n = B_n$ как систему уравнений относительно переменных X, причем значениями этих переменных должны служить термы. Достаточно всего двух приемов решения такой системы:

1. Равенство $f(t_1 \ldots t_m) = f(s_1 \ldots s_m)$ заменяется на группу равенств $t_1 = s_1, \ldots, t_m = s_m$.

2. Если в системе возникло равенство $x = t$, где x - переменная списка X, то проверяется, что x не встречается в t либо совпадает с t (иначе система несовместна). В первом случае равенство используется для исключения из системы неизвестной x, во втором - тавтологичное равенство $x = x$ отбрасывается.

Если в процессе преобразований возникает равенство двух термов с различными заголовками, отличными от переменных списка X, либо двух термов с различным числом корневых операндов, то система несовместна.

Для практических надобностей понятие унифицирующей подстановки пришлось существенно расширить. Если рассматриваются два терма, в которых встречаются коммутативные операции либо симметричные отношения, то в процессе их отождествления можно переставлять соответствующие операнды. В результате унифицирующая подстановка оказывается определенной уже неоднозначно. Например, при унификации $(\sin x)^2 + (\cos x)^2$ с $y + z$ можно либо вместо y подставить $(\sin x)^2$, а вместо z - $(\cos x)^2$, либо вместо y подставить $(\cos x)^2$, а вместо z - $(\sin x)^2$. Если разрешать в процессе унификации несложные тождественные преобразования, например, отождествлять a и $b \cdot c$, представляя переменную a как $1 \cdot a$, то количество способов унификации еще больше увеличится. Соответственно, процедура унификации в рассматриваемой логической системе превращается в достаточно сложный алгоритм, обращение к которому сопровождается множеством опций, в том числе, опцией обрыва перечисления унифицирующих подстановок, если их чрезмерно много.

Еще одно важное понятие - идентифицирующая подстановка. В этом случае имеются термы $A_1, \ldots, A_n$ и некоторое множество X их переменных. Если подстановка S

вместо переменных X переводит термы $A_1, \ldots, A_n$ в некоторое подмножество множества M, то она называется идентифицирующей эти термы с термами M. Подстановка, идентифицирующая посылки теоремы с некоторыми утверждениями, позволяет использовать теорему для вывода следствий данных утверждений. Поэтому работа решателя, в основном, сводится к поиску идентифицирующих подстановок.

Так называемая скобочная запись - представление термов в виде $f(t_1 \ldots t_n)$ - используется только для внутренних действий логической системы. Для более привычной прорисовки утверждений и выражений на экране используется формульная запись, приближенная к общепринятой математической записи. Имеется программа, преобразующая скобочную запись в структуру данных формульной записи и обратно. Она используется как при вводе термов в систему (например, при постановке задач решателю), так и при прорисовке текущих действий системы. Подробнее о способах формульной прорисовки некоторых конкретных типов утверждений и выражений, а также о способах ввода термов формульным редактором будет сказано далее.

Упражнения:

1. Найти все свободные и все связанные переменные утверждения

   $\forall_x(\exists_y(P(x,y,z)) \rightarrow Q(x,v))$

2. Найти подстановку вместо переменных x, y, z, u, v, унифицирующую термы $f(x, g(x,y))$ и $f(h(z,v),u)$.

3. Ввести новый символ "ттт", обозначающий операцию симметрической разности множеств "ттт(A,B)". Зарегистрировать сведения об этом символе в справочной информации решателя (войти в пустую программу символа "ттт", нажать F3 и далее создать поясняющий текст). Затем удалить этот текст (Ctr-Del) и удалить символ "ттт".

2.3 Примеры записи на логическом языке утверждений и выражений

Перечислим способы построения утверждений и выражений с помощью некоторых часто встречающихся логических символов. Чтобы получить более полную информацию о логическом языке системы, можно использовать следующие возможности:

1. Если нужно узнать, как используется конкретный логический символ S, в главном меню выбирается пункт "Просмотр программы логического символа" и символ S вводится в соответствующем окне. Затем нажимается Enter. На экране прорисовывается корневой фрагмент программы символа S. Сейчас он не нужен, так что сразу нажимается F3. На экране будет прорисовано множество различных сведений, связанных с символом S. Их можно прокручивать вверх-вниз клавишами курсора и PageUp-PageDown. Среди этих сведений выбирается текст вида "$S(x1 \ldots xn)$ - ". После тире идет объяснение того, что означает данное утверждение либо выражение. Похожий текст вида "$S(x1 \ldots xn)$.", где вместо тире стоит точка, используется для объяснения того, как работает оператор либо операторное выражение ЛОСа с заголовком S.

2. Чтобы получить список логических символов, рассматриваемых в некотором разделе R , нужно в оглавлении приемов выбрать этот раздел, перейти в его подпункт "Справочники", и далее - войти в концевой пункт "Содержание". Появится текст "содержание($R\, s_1 \dots s_n$)", где s_i - названия логических символов раздела либо его подразделов. Чтобы получить список логических символов подраздела, нужно снова выйти в оглавление приемов, найти этот подраздел и повторить описанную процедуру.

3. Имеется отдельное оглавление разделов с указанием их логических символов и их использованием. Чтобы войти в него, нужно из главного меню выбрать пункт "Оглавление приемов", войти в произвольный концевой пункт оглавления приемов для просмотра какого-либо приема, и далее нажать Ctr-я.

4. Из просмотра какого-либо приема ГЕНОЛОГа (как в предыдущем пункте) можно нажать клавишу "c", ввести название нужного логического символа, и после нажатия Enter перейти в просмотр справочной информации об этом символе (информация та же, что по нажатию F3 из просмотра ЛОС-программы).

5. Из просмотра приема ГЕНОЛОГа можно выделить левой кнопкой мыши вхождение какого-либо символа в теорему приема, после чего нажать правую кнопку. Синзу появится поясняющий текст.

6. Можно просмотреть полный список логических символов, нажав из главного меню "a" (упорядочение символов по алфавиту) либо "c" (упорядочение символов по номерам). Символы прорисованы в два столбца. Для смены выделенного символа в столбце использовать вертикальные клавиши курсора, для перехода между столбцами - горизонтальные. Для перелистывания списка использовать PageUp и PageDown. Чтобы получить информацию о выделенном символе, нажимается "и". Информация - та же, что по F3 из просмотра ЛОС-программы. Заметим, что данный способ - самый неудобный из перечисленных и используется крайне редко.

Ниже, приводя вид утверждений и выражений, иногда будем указывать способ их прорисовки в формульной записи. Если этого не сделано, формульная запись выглядит так же, как скобочная. Большинство материала данного раздела носит справочный характер, и при первом чтении его можно пропустить. Ограничимся лишь избранными понятиями пяти разделов: общелогические понятия, алгебра множеств, элементарная алгебра, элементарная геометрия, математический анализ. Для первого знакомства этого вполне достаточно.

Общелогические понятия

Начнем с перечисления наиболее общих логических символов языка, сохраняя для них те же названия, которые используются в системе. Все эти названия будем выделять в тексте кавычками.

Прежде всего, выделим логические константы "истина", "ложь", а также логические связки "и", "или", "не", "эквивалентно", позволяющие строить новые утверждения "и($A_1 \dots A_n$)"; "или($A_1 \dots A_n$)"; "не(A_1)"; "эквивалентно($A_1 A_2$)"из ранее построенных утверждений $A_1, \dots, A_n$. Формульная прорисовка этих утверждений обычная: "A_1 & ... & A_n", "$A_1 \vee \dots \vee A_n$", "$\neg(A)$", "$A_1 \leftrightarrow A_2$".

Логическая связка "если-то"используется только в сочетании с квантором общности: "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то A_0)", где $x_1, \dots, x_n$ - попарно различные переменные, $A_0, A_1, \dots, A_m$ - некоторые утверждения. Это обусловлено тем, что в бесквантерных ситуациях предпочтительнее использовать связки "или", "и", "не", к которым она легко сводится. В случае $m = 0$ запись приобретает вид "длялюбого($x_1 \dots x_n A_0$)". Как при $m > 0$, так и при $m = 0$ данная запись с квантором общности называется кванторной импликацией. Утверждения $A_1, \dots, A_m$ называются антецедентами импликации, утверждение A_0 - консеквентом.

Формульная запись кванторной импликации с непустым списком антецедентов имеет вид "$\forall_{x_1 \dots x_n}(A_1 \ \& \ \dots \ \& \ A_m \to A_0)$", с пустым - "$\forall_{x_1 \dots x_n} A_0$".

Аналогичным образом записывается утверждение существования значений $x_1, \dots, x_n$, при которых истинно A_0: "существует($x_1 \dots x_n A_0$)". Чтобы обеспечить возможность компактной записи утверждения о существовании и единственности этих значений, введена модификация квантора существования "Существует($x_1 \dots x_n \ A_0$)" (логический символ "Существует" - с большой буквы). Символы "длялюбого", "существует"и "Существует" - связывающие; переменные $x_1, \dots, x_n$ образуют в приведенных выше утверждениях связывающую приставку.

Формульная запись утверждения существования имеет вид "$\exists_{x_1 \dots x_n} A_0$" либо, если утверждается существование и единственность, вид "$\exists!_{x_1 \dots x_n} A_0$".

Простейшими выражениями языка являются однобуквенные слова, образованные переменными либо логическими символами. Во втором случае считается, что выражение имеет своим значением тот символ, из которого оно состоит. Такое соглашение упрощает использование логического языка для описания вида его собственных конструкций. Хотя из него и вытекает, что константы "истина"и "ложь"одновременно должны считаться и утверждениями, и выражениями, особых неудобств в дальнейшем это не создает.

Для задания значения путем разбора случаев используется условное выражение "вариант($A \ t_1 \ t_2$)". Если утверждение A является истинным, то значение этого выражения равно значению выражения t_1, иначе оно равно значению выражения t_2. Аналогичным образом, рассматриваются условные утверждения "альтернатива($A \ B_1 \ B_2$)". Если утверждение A истинно, то истинность условного утверждения совпадает с истинностью утверждения B_1, иначе - с истинностью утверждения B_2. Условные утверждения оказались практически неиспользуемыми в формулировках и решениях задач, однако они весьма часто применяются в логических описаниях структурного уровня - внутри программ приемов решателя.

В формульной записи условное выражение имеет вид "(t_1 при A, иначе t_2), условное утверждение - вид "альтернатива(A, t_1, t_2)".

Используются всего две конструкции, вводящие связанные переменные при построении новых выражений. Первая из них - "класс($x_1 \dots x_n \ \ A$)" - определяет класс всех наборов ($x_1 \dots x_n$), на которых утверждение A является истинным. Вторая - "отображение($x_1 \dots x_n \ \ A \ t$)" - задает функцию, определенную на множестве всех таких наборов ($x_1 \dots x_n$), для которых утверждение A является истинным, и принимающую на этом множестве значения, определяемые выражением t. Для задания интеграла, производной, предела, суммы по множеству значений параметра, и т.п., используются обычные операции над функциями, не вносящие сами по себе каких-либо связанных переменных.

В формульной записи описатель "класс" имеет вид "$\mathrm{set}_{x_1 \ldots x_n} A$", описатель "отображение" - вид "$\lambda_{x_1 \ldots x_n}(t, A)$". Например, "$\mathrm{set}_x(x - \text{число} \ \& \ 1 < x \ \& \ x < 2)$" обозначает численный интервал $(1, 2)$, "$\lambda_x(x^2, x - \text{число})$" - квадратичную функцию, определенную на всей числовой прямой.

К списку используемых в языке общелогических конструкций добавим также равенство "равно(t_1 t_2)" и выражение "значение(f t)", определяющее значение функции f в точке t (в формулной записи - $t_1 = t_2$ и $f(t)$).

Алгебра множеств

Утверждения "множество(A)" ; "принадлежит(t A)" ; "содержится(AB)"означают, соответственно, что A есть множество; t - элемент этого множества; множество A является подмножеством множества B. Для условия непересечения двух множеств A,B введено обозначение "непересек(A B)". Формульная прорисовка - "$A - \mathrm{set}$", "$t \in A$", "$A \subseteq B$", "непересек(A, B)".

Пустое множество обозначается посредством логического символа "пусто". Выражения "объединение($A_1 \ldots A_n$)", "пересечение($A_1 \ldots A_n$)", "разность(A B)", "прямое-произведение($A_1 \ldots A_n$)"используются для обозначения простейших операций над множествами. Формульная прорисовка обычная: $\emptyset$, "$A_1 \cup \ldots \cup A_n$", "$A_1 \cap \ldots \cap A_n$", "$A \setminus B$", "$A_1 \times \ldots \times A_n$".

Для задания конечной последовательности элементов $A_1, \ldots, A_n$ используется выражение "набор($A_1 \ldots A_n$)". Всюду далее, говоря о наборах, мы отождествляем это понятие с понятием конечной последовательности. В тех случаях, когда приходится вводить операции либо отношения с переменным числом операндов (за исключением двуместных ассоциативных операций), обычно они представляются как одноместные операция либо отношение над набором. В частности, конечное множество, состоящее из элементов $A_1, \ldots, A_n$, обозначается посредством выражения "перечень(набор(A_1 $\ldots A_n$))". Для пустого набора (последовательности длины 0) введено обозначение "пустоеслово". Формульная прорисовка набора - "$(A_1, \ldots, A_n)$", конечного множества - "$\{A_1, \ldots, A_n\}$".

Для результата последовательной записи наборов $A_1, \ldots, A_n$ используется выражение "конкатенация($A_1 \ldots A_n$)". Результаты добавления к началу либо концу набора A элемента t обозначаются, соответственно, "префикс(t A)"и "суффикс(A t)". Чтобы выделить из набора $(A_1 \ldots A_n)$ поднабор $(A_i \ldots A_j)$, используется обозначение "поднабор(A i j)". Для обозначения результата вставки в указанный набор элемента B перед i-м элементом служит выражение "Вставка(A i B)". Формульная прорисовка конкатенации - "$; A_1; \ldots; A_n$".

Особо выделяются случаи прорисовки термов "перечень(A)", задающих множество элементов конечного набора A. Если заголовок терма A - набор, то формульная прорисовка уже указана выше. Если A имеет вид "префикс(a, b)", то множество элементов прорисовывается как "$\{a; b\}$". В остальных случаях - как "$\{; A\}$".

Утверждение "семействомножеств(A)"означает, что A есть функция, принимающая в качестве своих значений множеств. Выражения "объединениевсех(A)", "пересечениевсех(A)" обозначают объединение и пересечение множеств для всевозможных значений аргумента функции A, принадлежащих ее области определения. Формуль-

ная прорисовка - стандартная. Например,

$$\bigcup_{i=1}^{n} A(i)$$

Обычным образом используются выражение "мощность(A)", утверждение "конечное(A)"и константы "счетное", "континуум". Формульная прорисовка мощности - "card(A)".

Перечислим обозначения, введенные для работы с числовыми множествами. Числовой промежуток с концами a, b и указателями c, d отнесения этих концов к промежутку (0 - конец не включается в промежуток, 1 - включается) обозначается "промежуток(a b c d)". В частности, таким образом могут задаваться и бесконечные промежутки; для обозначения их концов (и в других аналогичных случаях) используются логические символы "минусбеск"и "плюсбеск". В формульной прорисовке числовые промежутки изображаются традиционным образом, с использованием круглой скобки для указания невключения конца в промежуток и квадратной - если конец относится к промежутку. Символы бесконечности прорисовываются как ∞ и $-\infty$. Множество вещественных чисел во внутреннем представлении обозначается посредством "промежуток(минусбеск плюсбеск 0 0)", хотя в формульном виде оно прорисовывается как R. Утверждение "числовойотрезок(A)"означает, что A есть отрезок на числовой прямой (включая концы).

Множества целых, целых неотрицательных, натуральных и рациональных чисел обозначаются, соответственно, "целые" , "целыенеотрицательные", "натуральные"и "рациональные". В формульной прорисовке - $\mathbb{Z}, \mathbb{N}+, \mathbb{N}, \mathbb{Q}$.

Множество целых чисел, состоящее из элементов $i, i+1, \ldots, j$, обозначается "номера(i j)" . В формульной прорисовке - "$\{i, \ldots, j\}$". Набор тех же самых элементов обозначается "наборномеров(i j)" . Множество всех членов арифметической прогрессии с первым элементом a и знаменателем p обозначается "арифмпрогрессия(a p)".

Для работы с функциями используются обозначения "функция(f)"(утверждение о том, что f есть функция); "область(f)"(область определения функции f); "значения(f)"(множество значений функции f); "Отображение(f A B)"(утверждение о том, что f есть функция, определенная на множестве A и принимающая значения в множестве B); "взаимнооднозначно(f)"(утверждение о том, что функция в различных точках своей области определения принимает различные значения). В формульной прорисовке область определения выглядит как "Dom(f)", множество значений - "Val(f)". Условие "функция(f)" прорисовывается как "f – функция".

Образ множества M для функции f обозначается "образ(f M)" ; прообраз множества M - "прообраз(f M)" ; прообраз элемента t - "слой(f t)".

Для задания конкретных функций используются следующие выражения.
"конст(M b)"обозначает константную функцию, принимающую во всех точках своей области определения M значение b. "См(a b)"обозначает функцию, определенную на одноэлементном множестве $\{a\}$ и принимающую на нем значение b. Формульная прорисовка - "$a \rightarrow b$". "тождфунк(a)" обозначает тождественную функцию, определенную на множестве a. "таблица(A)"обозначает результат "объединения" функций множества A, принимающих на пересечении своих областей определения одинаковые значения. "доопределение(f M b)" обозначает результат доопре-

деления функции f в тех точках множества M, где она еще не определена, значением b. "сужение(f M)" обозначает сужение функции f на множество M. Утверждение "перестановка(f A)"означает, что функция f представляет собой взаимнооднозначное отображение начального отрезка натурального ряда на конечное множество A.

Число переменных многоместной функции f обозначается "числопеременных(f)" ; функция, получающаяся из f подстановкой констант набора b вместо переменных, номера которых (начиная с 1) перечислены в наборе a, обозначается "подфункция(f a b)" . Утверждение "существперем(i f)" означает, что i - я переменная функции f является существенной.

Элементарная алгебра

Все рассматриваемые в этом разделе операции и отношения являются вещественнозначными; для работы с комплексными числами предусмотрены альтернативные обозначения.

Для представления десятичного числа $a_1 \dots a_n$ ($a_1, \dots, a_n$ - цифры) в виде терма используется запись "величина($a_1 \dots a_n$)" . Если число дробное, то одно из a_i в этой записи является символом запятой (запятая введена в число логических символов только для данной цели; при изображении на экране термов в формульной записи вместо запятой используется точка). Заметим, что в случае $n = 1$ символ "величина"не используется, а число представляется цифрой. Для простейших операций применяются обозначения "плюс($a_1 \dots a_n$)" (здесь и далее $n \geq 2$); "минус(a)" ; "умножение($a_1 \dots a_n$)" ; "дробь(a_1 a_2)" ; "степень(a_1 a_2)" ; "максимум($a_1 \dots a_n$)" ; "минимум($a_1 \dots a_n$)". Формульные обозначения стандартные. Операция вычитания не введена, так как все равно при решении задач ее приходится выражать через "плюс"и "минус", а прорисовка вычитания в стандартной математической записи неотличима от выражения с плюсом и минусом.

Сумма и произведение конечного семейства значений обозначаются "суммавсех (F)"и "произведениевсех(F)". Здесь F - функция, определенная на некотором конечном множестве и принимающая в качестве своих значений суммируемые либо перемножаемые величины. В формульной записи эти выражения изображаются обычным образом, с помощью знаков $\sum$ и $\prod$.

Неравенства записываются в виде "меньше(a b)" ; "больше(a b)" ; "меньшеилиравно(a b)" ; "большеилиравно(a b)". Утверждение "число(a)"означает, что a есть вещественное число. Логические символы "е"и "пи"обозначают соответствующие числовые константы. Формульные прорисовки обычные.

Логарифм числа b по основанию a обозначается "логарифм(a b)" . Для прочих элементарных функций используются обозначения: "синус(a)" ; "косинус(a)" ; "тангенс(a)" ; "котангенс(a)" ; "секанс(a)" ; "косеканс(a)" ; "арксинус(a)" ; "арккосинус(a)" ; "арктангенс(a)" ; "арккотангенс(a)" ; "модуль(a)". Выражение "сигнум(a)" считается равным минус единице для отрицательных чисел, не определенным в нуле, и равным единице для положительных чисел. "Сигнум(a)" доопределяется в нуле единицей. Для гиперболических функций используются обозначения "гипсинус(a)" ; "гипкосинус(a)" ; "гиптангенс(a)" ; "гипкотангенс(a)". Формульные прорисовки обычные. Заметим, что степень тригонометрической функции обозначается не обычным "сокращенным" образом (например, $\sin^2 x$), а только в явном виде: $(\sin x)^2$.

Утверждения "целое(a)" , "натуральное(a)" , "четное(a)" , "рациональное(a)" , "простое(a)" уточняют тип вещественного числа a. Выражения "числитель(a)" и "знаменатель(a)" определяют, соответственно, целочисленный числитель и натуральный знаменатель рационального числа a (после сокращения соответствующей дроби). Утверждение "делит(m n)" означает, что целое число m делит целое число n. Формульная запись обычная. Выражения "нод($n_1 \dots n_k$)" и "нок($n_1 \dots n_k$)" означают, соответственно, наибольший общий делитель и наименьшее общее кратное целых чисел $n_1, \dots, n_k$. Утверждение "взаимнопросты(m n)" означает взаимную простоту целых чисел m и n. Выражение "целаячасть(a)" обозначает целую часть числа a. Формульная запись обычная. Для обозначения вычета целого числа m по натуральному модулю n служит выражение "вычет(m n)". Формульная запись - "$a \pmod b$)".

Выражение "числосочетаний(m n)" обозначает число сочетаний из m по n; выражение "факториал(n)" - факториал целого неотрицательного числа n. Формульные прорисовки обычные.

Формальные многочлены требуют в логическом языке специального представления - они, разумеется, не являются функциями, но не являются также и формулами. Последние суть слова в конечном алфавите и множество их всего лишь счетно, в то время как множество формальных многочленов над полем вещественных чисел континуально. Поэтому многочлены рассматриваются как абстрактные объекты, задаваемые тройкой (X, F, K). Здесь $X = (x_1, \dots, x_n)$ - упорядоченный набор символов переменных; F - поле; K - функция, определенная на некотором конечном подмножестве M множества n-ок целых неотрицательных чисел и принимающая значения из поля F. Эта функция сопоставляет набору степеней переменных $x_1, \dots, x_n$ коэффициент соответствующего одночлена, входящего в многочлен. Для обозначения многочлена используется выражение "многочлен(X F K)". Разумеется, в формульной записи предусмотрено более удобное обозначение: $\mu_{x_1 \dots x_n}(S)$, где S - обычным образом записанная сумма одночленов. Переход от одного представления многочлена к другому выполняется автоматически. Утверждения "Многочлен(A)" и "веществмногочлен(A)" используются для указания на то, что A есть многочлен (в первом случае - общего вида, во втором - над полем вещественных чисел).

Элементарная геометрия

Почти все вводимые в элементарной геометрии понятия относятся одновременно и к двумерному, и к трехмерному случаям. Использование в задаче понятия, имеющего смысл только для двумерного случая, автоматически означает, что эта задача целиком планиметрическая. Чтобы ускорить работу решателя при рассмотрении планиметрических задач, в список посылок таких задач обычно вводится (как правило, автоматически самим решателем) техническая пометка "планиметрия". Эту пометку можно рассматривать как вспомогательное фиктивное утверждение, указывающее на существование некоторой общей плоскости, к которой относятся все рассматриваемые в задаче точки.

Утверждение "точка(A)"означает, что A есть точка трехмерного пространства (в планиметрическом случае эта точка относится к некоторой не уточняемой плоскости данного пространства). Выражение "прямая(A B)"обозначает прямую, проходящую через точки A, B. В случае совпадения этих точек уславливаемся считать, что данное выражение обозначает какую-то произвольную прямую, проходящую через рассмат-

риваемую точку. Аналогично, выражение "плоскость(A B C)"обозначает плоскость, проходящую через точки A, B, C.

Обучение решателя выполнялось таким образом, что он может работать только с прямыми и плоскостями, заданными явным образом через пару (тройку) точек с помощью указанных выражений. Тем не менее, в языке предусмотрены используемые в некоторых специальных случаях утверждения "Прямая(A)"и "Плоскость(A)", указывающие, что A является, соответственно, прямой либо плоскостью.

Выражения "отрезок(A B)"и "интервал(A B)"обозначают, соответственно, отрезок и интервал с концами в точках A, B. Выражения "луч(A B)"и "обратныйлуч (A B)" обозначают, соответственно, луч с началом в точке A, проходящий через точку B и луч, обратный данному лучу.

Расстояние межу точками A и B обозначается "расстояние(A B)". Формульная прорисовка - $l(AB)$. Расстояние от точки A до прямой либо плоскости B обозначается, соответственно, "расстдопрямой (A B)"и "расстдоплоскости(A B)". Расстояние между двумя замкнутыми множествами точек A и B обозначается "расстмежду(A B)".

Величина угла с вершиной в точке B, стороны которого проходят через точки A и C, обозначается "угол(A B C)". Формульная прорисовка - $\angle(ABC)$. Эта величина измеряется от 0 до π. Угол как фигура в этом случае обозначается "Угол(A B C)". Утверждение о том, что луч AD является биссектрисой угла ABC, записывается в виде "биссектриса(A B C D)". Прямая, на которой лежит данная биссектриса, обозначается "Биссектриса(A B C)".

Утверждения "параллельны(A B)"и "перпендикулярно(A B)"используются для указания на параллельность либо перпендикулярность прямых либо плоскостей A, B (допускаются любые сочетания: прямая - прямая; прямая - плоскость; плоскость - плоскость). Формульные обозначения - $A \parallel B$ и $A \perp B$.

Для указания на то, что две точки A и B лежат по одну сторону от прямой либо плоскости C, используется утверждение "однасторона(A B C)". Утверждение "разныестороны(A B C)"указывает, что они лежат по разные стороны. Если C - прямая, то в обоих случаях утверждение указывает также и на принадлежность точек A, B общей плоскости с этой прямой. Если одна из точек попадает на прямую (плоскость), то считается, что точки одновременно лежат и по одну, и по разные стороны от прямой (плоскости).

Утверждение "треугольник(ABC)"означает, что точки A, B, C суть вершины треугольника (различны и не лежат на одной прямой). Аналогично, утверждения "параллелограмм($ABCD$)" , "ромб($ABCD$)" , "прямоугольник($ABCD$)" , "квадрат(AB CD)", "трапеция($ABCD$)"означают, что четверка точек A, B, C, D составляет набор вершин четырехугольника соответствующего типа. В первых четырех случаях допускаются произвольные циклические перестановки; в последнем случае вершины A, D должны лежать на большем (нижнем) основании трапеции. Заметим, что предусмотрены два обозначения для трапеции: указанное выше предполагает, что оба угла при нижнем основании не больше 90 градусов; в обозначении "Трапеция(AB CD)" это предположение отсутствует. Утверждение "четырехугольник($ABCD$)"означает, что четверка точек A, B, C, D образует вершины выпуклого четырехугольника. Если a - набор точек, то утверждения "многоугольник(a)" ; "правмногоугольник(a)" означают, что данные точки образуют последовательно проходимые вершины некоторого многоугольника (во втором случае - правильного).

Выражение "фигура(a)" обозначает множество всех точек многоугольника, ограниченного ломаной, проходящей через точки набора a. Утверждение "центр(P a)"означает, что точка P является центром области a.

Утверждения "Медиана($ABCD$)" и "Высота($ABCD$)" означают, что точка D является основанием, соответственно, медианы либо высоты треугольника ABC, проведенной из вершины A. Утверждение "Биссектреуг($ABCD$)"означает, что точка D является основанием биссектрисы этого же треугольника, проведенной из вершины B.

Выражения "площадь(a)"и "периметр(a)"обозначают площадь и периметр области a (во втором случае - только имеющей вид многоугольника). Выражение "длина(a)" обозначает длину линии a. Площадь прорисовывается как $S(a)$.

В двумерном случае окружность с центром в точке A обозначается либо как "окружность(AB)" , где B - некоторая точка на этой окружности, либо как "окр(A r)", где r - радиус. Аналогичные обозначения "круг(AB)"и "круградиуса(A r)"используются для круга. В трехмерном случае задание окружности либо круга дополняется указанием третьей точки, лежащей в ее (его) плоскости. Здесь используются обозначения "Окружность(ABC)"и "Круг(ABC)"; C - дополнительная точка, фиксирующая плоскость.

Дуга окружности с центром A, имеющая концевые точки B, C, обозначается либо "дуга(ABC)" (меньшая из двух дуг), либо "большаядуга(ABC)" (большая из двух дуг). В случае диаметрально противоположных точек B, C большая и меньшая (условно) дуги выбираются некоторым неуточняемым образом, причем различны. Выражение "дугаугла(ABC)"обозначает дугу, на которую опирается вписанный угол ABC.

Чтобы задавать область со сложной границей, образованной множеством фрагментов стандартного вида (отрезки прямых, дуги окружностей и т.п.), используется запись "Фигура(перечень(набор($A_1 \dots A_n$)))" , где $A_1, \dots, A_n$ - обозначения данных фрагментов.

Выражения "сектор(A B C)"и "сегмент(A B C)"обозначают, соответственно, сектор и сегмент круга, имеющего центр в точке A и определяемые крайними точками B, C, лежащими на окружности. Выбирается меньшая из дуг между данными точками (в случае равенства дуг берется неуточняемым образом одна из них).

Утверждение "касательная(A B)"означает, что прямая A является касательной к окружности B. Утверждения "внешкасательная(A B C)"и "внутркасательная(A B C)"означают, что прямая A является, соответственно, внешней либо внутренней общей касательной окружностей B и C. Утверждения "внешкасаются(A B)"и "внуткасаются(A B)"означают, что окружности A и B касаются друг друга, соответственно, внешним либо внутренним образом.

Утверждение "вписана(A B)"означает, что окружность A вписана в многоугольник B (последний обычно задается выражением вида "фигура(...)"). Аналогично, утверждение "описана(A B)"означает, что окружность A описана около многоугольника B.

Утверждение "подобны(A B)"означает, что многоугольники, наборы вершин которых суть A и B, подобны. Заметим, что здесь вместо обозначающего многоуольник выражения "фигура(A)"используется выражение A, определяющее набор его вершин. Утверждение "подобны(A B)"дополнительно фиксирует соответствие между

вершинами, при котором имеет место подобие: вершине A_i соответствует вершина B_i.

Утверждение "выпукло(A)"указывает на выпуклость множества A.

В отличие от планиметрии, в стереометрии не введены обозначения тел и поверхностей через их "опорные"точки. В задачах они должны обозначаться переменными.

Утверждения "куб(a)", "призма(a)", "параллелепипед(a)", "пирамида(a)", "усечпирамида(a)", "конус(a)", "шар(a)", "сфера(a)", "цилиндр(a)" указывают тип тела либо поверхности a. Дополнительно к этим утверждениям, можно применять также утверждения "прямой(a)" (случаи прямой призмы и прямого параллелепипеда), "правильный(a)" (случаи правильных призмы, пирамиды либо усеченной пирамиды) и "прямоугольный(a)" (применительно к параллелепипеду).

Выражения "объем(a)" , "площповерхности(a)" , "боковаяповерхность(a)"обозначают, соответственно, объем, полную площадь поверхности и площадь боковой поверхности тела a. Для обозначения площади поверхности a, как и в плоском случае, используется выражение "площадь(a)". Высота тела a (пирамида, конус, цилиндр и т.п.) обозначается "высота(a)" . Выражение "радиус(a)"обозначает радиус шара либо сферы a.

Утверждение "грань(a b)" означает, что многоугольник a является гранью многогранника b. Утверждение "основание(a b)"означает, что плоская фигура a служит основанием тела b. Утверждения "вершина(a b)" ; "ребро(a b)" означают, что точка либо, соответственно, отрезок a являются вершиной

Математический анализ

Предел функции вещественного переменного f в точке a обозначается "предел(f p a)" ; здесь p - указатель типа предела: $p = 0$ означает двусторонний предел; $p = 1$ - левый предел и $p = 2$ - правый предел. Предел числовой последовательности обозначается таким же образом, но в этом случае функция f определена на множестве натуральных чисел. Аналогичным образом используются обозначения "верхнийпредел(f p a)"и "нижнийпредел(f p a)". Формульные прорисовки стандартные. Лишь в случае, если тип предела p - переменная, прорисовка имеет вид "$\lim_{x \to a \backslash p} f(x)$".

Используемые в асимптотических оценках обозначения $f(x) = O(g(x))$ и $f(x) = o(g(x))$ представляют собой хороший пример контекстной семантики. Они имеют смысл только в сочетании с указанными в контексте допущениями о "стремлении" x к некоторому значению. Разумеется, можно было бы исключить из языка такую контекстную зависимость и сгруппировать указанное выше равенство с указателем на предельное поведение аргумента в одну синтаксическую конструкцию, однако это привело бы к неоправданно громоздким записям, так как обычно одни и те же указатели предельного поведения обслуживают множество различных равенств с O и o. Поэтому в языке решателя сохранено обычное применение таких равенств. Небольшое изменение здесь затрагивает лишь o: вместо указанного выше равенства, во "внутреннем"представлении используется запись "$o(f(x), g(x))$".

Значение производной функции f одного вещественного переменного в точке a обозначается "производная(f a)". Значение в точке a частной производной функции f, зависящей от n переменных, обозначается "частнпроизв(f K a)". Здесь $K = (k_1 \dots k_n)$ - набор кратностей дифференцирования по отдельным переменным; если

$k_i = 0$, то по i - му аргументу дифференцирование не выполняется. Значение повторной производной функции f одного вещественного переменного также обозначается посредством "частнпроизв(f K a)" ,однако здесь уже K - не набор, а величина кратности дифференцирования. Формульные прорисовки стандартные. Утверждение "Производная(f g)"означает, что значения функции g в области ее определения суть значения производной функции f. Аналогично, утверждение "Частнпроизв(f i g)"означает, что значения функции g в области ее определения суть значения частной производной функции нескольких переменных f по ее i - му аргументу. Значение полного дифференциала порядка n функции f нескольких переменных в точке x при наборе a приращений значений ее аргументов обозначается "дифференциал(f n x a)".

Для обозначения неопределенного интеграла функции f одного вещественного переменного используется чисто техническая запись "Интеграл(f)". Предполагается, что ее значением является какая - то одна из множества возможных первообразных функций для f. В процессе решения задачи эта первообразная находится, и тогда указанная запись устраняется. Разумеется, для использования в базе теорем следовало бы взять не одноместную операцию над f типа указанной выше, а двуместное отношение между функцией и ее первообразной, либо (как это обычно и делается) в качестве значения для неопределенного интеграла брать все множество первообразных. Однако, первый способ приводит при формальном интегрировании к обозначениям, сильно отличающимся от общепринятых, а второй заставляет неоправданно усложнять язык, вводя в него операции над множествами функций. Поэтому для вычислений была выбрана указанная выше запись, как технически наиболее удобная, хотя и требующая достаточно жестких ограничений на контекст, при которых ее использование не приводит к противоречиям. Определенный интеграл функции f одной вещественной переменной по ее области определения обозначается "интеграл(f)". При прорисовке на экране этого интеграла в обычной записи происходит "извлечение"из описания области определения f пределов интегрирования и явное их указание. Для двойного и тройного интегралов введены обозначения "двойнойинтеграл(f)"и "тройнойинтеграл(f)", где f - числовая функция, зависящая, соответственно, от двух либо от трех переменных. Интегралы берутся по всей области определения f. Формульные прорисовки интегралов обычные.

Утверждение "сходится(f)"означает сходимость последовательности (функции натурального аргумента) f. Числовой ряд рассматривается как функция натурального аргумента. Утверждение о сходимости ряда с общим членом A_i формулируется в терминах последовательности его частичных сумм как "сходится(отображение(n натуральное(n) суммавсех(i целое(i) & $1 \le i$ & $i \le n$ A_i)))". Сумма ряда с общим членом A_i обозначается как "суммавсех(отображение(n целое(n) & $k \le n$ A_n))".

Вспомогательное утверждение "стремится(x a p)"означает то же самое, что $x \to a$, причем p - указатель двусторонней ($p = 0$) либо односторонней ($p = 1$ - левая, $p = 2$ - правая) окрестности. Оно не имеет самостоятельной семантики, а служит для определения семантики других утверждений, встречающихся с ним в общем контексте (например, $f(x) = O(g(x))$).

Для указания локальных свойств числовой функции f введены обозначения "переменазнака(f a)" (функция меняет знак в окрестности точки a); "убываетвточке(f a)"и "возрастаетвточке(f a)". Убывание и возрастание здесь понимаются в строгом смысле. Утверждение "огрвточке(f a)"означает ограниченность функции f в окрестности точки a.

Утверждение "Максимум(f A B c)" означает, что B есть множество всех точек максимума функции f на множестве A, причем c - значение ее в этих точках. В случае минимума используется запись "Минимум(f A B c)". Утверждение "экстремум(f a b c)" означает, что функция f имеет экстремум в точке a; b - значение ее в этой точке, c - тип экстремума - логический символ "минимум"либо "максимум". Формульные прорисовки символов - $Max, Min, Extr$.

Утверждения "убывает(f A)" , "возрастает(f A)" , "неубывает(f A)" , "невозрастает(f A)"обозначают строгую и нестрогую монотонность функции f на множестве A. Для указания выпуклости либо вогнутости служат обозначения "Выпуклавверх (f A)"и "Выпуклавниз(f A)".

Для обозначения непрерывности и равномерной непрерывности числовой функции f на множестве A используются записи "непрерывна(f A)" , "равномернонепрерывна(f A)". Утверждение "периодична(f A)" означает, что функция f периодична на всей числой оси и число A является ее периодом (не обязательно наименьшим). Утверждения "четнаяфункция(f)"и "нечетнаяфункция(f)"указывают четную либо нечетную функцию.

Для операций сложения, умножения, и т.п. , применяемых к числовым функциям, а не к числам, в логическом языке должны быть введены независимые обозначения - двойники аналогичных обозначений для числовых операций. Так, для сложения числовых функций с одинаковой областью определения введено обозначение "плюсфунк" ; для умножения - "умножфунк" ; для изменения знака - "минусфунк" . Как и в случае чисел, первые две операции могут применяться к произвольному (большему единицы) числу операндов. На экране операции над функциями прорисовываются так же, как и соответствующие числовые операции.

2.4 Ввод утверждений и выражений в стандартной математической записи

В большинстве случаев утверждения и выражения вводятся в систему при помощи формульного редактора, в обычной математической записи. Как правило, для логического символа предусмотрена специальная клавиша либо группа клавиш, нажимаемых последовательно. Формульный редактор обеспечивает стандартную прорисовку формулы непосредственно в процессе ее набора, автоматически изменяя при необходимости размеры и размещение фрагментов формулы.

Приведем краткое описание работы с формульным редактором. При входе в него появляется прямоугольник курсора (коричневатого цвета). Этот курсор, в отличие от курсора текстового редактора, нельзя перемещать клавишами перемещений курсора - его положение зафиксировано и соответствует текущему вводимому символу формулы. Ситуация при использовании формульного редактора "линейная можно либо ввести очередной символ, либо отменить последний введенный символ нажатием "Backspace". В процессе ввода формулы происходят автоматические масштабирование, "центровка"и перенесение на новую строку элементов изображения.

Для завершения ввода формулы нажимается "Enter". Эта же клавиша служит для указания на завершение набора отдельных фрагментов формулы - дробей, радикалов, интегралов и т.п. Применение ее в таких случаях должно быть аккуратным,

так как повторное нажатие приведет к преждевременному обрыву набора формулы (впрочем, для возвращения в редактирование терма задачи достаточно выделить этот терм и нажать "ф"). Чтобы отменить начатое редактирование формулы и выйти из формульного редактора, нажимается "Esc". Заметим, что если формула заведомо не набрана до конца либо набрана с принципиальными ошибками, то формульный редактор игнорирует завершающие нажатия клавиши "Enter". В этом случае следует либо довести набор формулы до конца, либо вернуться (Backspace) к ошибке и исправить ее, либо вообще повторно набрать всю формулу целиком.

При входе в формульный редактор автоматически осуществляется переход в латинский режим ввода символов с клавиатуры; при выходе - автоматическое возвращение в "кириллицу". Несмотря на "латинский"режим, приводимые далее коды клавиш и сочетаний клавиш, используемые в формульном редакторе, часто даются через "кириллические"обозначения клавиш (никакого переключения режима клавиатуры при их использовании не предполагается - они приводятся в таком виде просто для удобства запоминания). В тех случаях, когда возможна путаница, явно указывается на использование кириллицы (в скобках ставится пометка "рус.").

Заметим, что в любых режимах логической системы для ручного перехода в латинский режим клавиатуры достаточно нажать F12, а для перехода в режим кириллицы - F11. Однако, использовать эти возможности рекомендуется только при работе в текстовом редакторе. Следует помнить, что все управляющие клавиши и сочетания клавиш в логической системе ориентированы на кириллический режим ввода, так что переход к латинскому режиму будет означать просто отключение соответствующих управляющих воздействий.

Переменные в формульном редакторе набираются либо без индексов (малые и большие латинские буквы), либо с числовыми индексами. Для набора индекса после ввода переменной нажимается клавиша "курсор вниз", и далее вводится индекс. По окончании ввода индекса нажимается "Enter".

Некоторые логические символы обозначаются в "стандартной"математической записи теми же словами и словосочетаниями, что и во внутренней скобочной записи. Это относится в первую очередь к понятиям, для которых в математике не предусмотрено специальной символики. Разумеется, по мере обучения решателя доля таких понятий постоянно увеличивается. В особых случаях, для наиболее часто встречающихся понятий, вводятся специальные клавиши либо группы клавиш (обычно 2 последовательно нажимаемые клавиши). В остальных случаях для ввода логического символа, изображаемого словом либо словосочетанием, нажимается клавиша "Ctr-Enter", переводящая в режим побуквенного набора данного символа текстовым редактором. По окончании набора нажимается клавиша "Enter", возвращающая в режим формульного редактора.

Переход на новую строку выполняется формульным редактором автоматически. Однако, при использовании указанного выше способа ввода логического символа через "Ctr-Enter" может оказаться, что для набора этого символа текстовым редактором на текущей строке места недостаточно. Тогда перед набором следует обеспечить переход к новой строке вручную, нажав клавишу "Ctr-курсор вниз".

Подробное перечисление поддерживаемых формульным редактором логических символов и способов их ввода приведено в справочнике по логической системе. Этот справочник доступен, например, из главного меню (по F1); в процессе набора формулы

также можно перейти к нему, нажав клавишу F1. После нажатия возникает оглавление справочника. В этом оглавлении следует найти корневой раздел, и в нем - выбрать пункт "Формульный редактор". Для возвращения в набор формулы из просмотра справочника нажимается "End"(из оглавления справочника - однократно, из концевого текста справочника - "End"нажимается дважды). Здесь мы приводим перечисление лишь некоторых возможностей формульного редактора. В остальных случаях следует (возможно, прямо из набора формулы) обращаться к справочнику. Дальнейший материал данного раздела имеет чисто справочный характер и при первом чтении может быть опущен. В конце раздела приводятся упражения на ввод формул, которые помогут запомнить основные группы клавишей.

Общелогические символы

Отрицание $\neg A$ утверждения A вводится путем последовательного нажатия клавиш n,o (лат.) и последующего набора "A". Символы "и","или"вводятся, соответственно, как & и Ctr-v. Символ "если-то", встречающийся только под квантором общности, изображается стрелкой и вводится нажатием клавиши "курсор вправо". Символ "эквивалентно"обозначается двусторонней стрелкой и вводится нажатием клавиши "курсор влево".

Для ввода кванторной записи
"$\forall x_1 \dots x_k(A_1 \& \dots \& A_n \to A_0)$"сначала дважды нажимается большая латинская буква A и появляется знак квантора общности. Затем перечисляются (без каких-либо пробелов)переменные $x_1, \dots, x_k$ кванторной приставки; нажимается "Enter"(здесь появляется открывающая скобка), после чего вводится конъюнкция $A_1 \& \dots \& A_n$. Далее нажимается "курсор вправо"(появляется стрелка для "если-то"), вводится утверждение A_0, и в конце ставится закрывающая скобка.

Для ввода кванторной записи "$\exists x_1 \dots x_k(A)$"сначала дважды нажимается большая латинская буква E и появляется знак квантора существования. Затем набирается кванторная приставка $x_1 \dots x_k$; нажимается "Enter"(появляется открывающая скобка); вводится утверждение A, и ставится закрывающая скобка.

Возможно использование квантора "существует и единственно". После знака квантора существования здесь идет восклицательный знак. Для появления такого обозначения следует вместо двукратного нажатия "E"нажать последовательно клавиши "E"и "T"(лат.).

Равенство вводится обычным образом. Логические константы "истина"и "ложь"вводятся, соответственно, двукратным нажатием клавиши t либо f.

Условное выражение (A при P, иначе B), принимающее значение A, если истинно условие P, и значение B в противном случае, набирается следующим образом. Ставится открывающая скобка; вводится выражение A; нажимается Ctr-e (e-кир., от слова "если"); вводится P; нажимается Ctr-и (и - от слова "иначе"); вводится B, и ставится закрывающая скобка.

Арифметические операции и отношения

Числовые константы набираются обычным образом; в случае десятичных дробей используется точка. Операции "плюс" (произвольное число слагаемых, большее или равное 2) и "минус"вводятся клавишами "+"и "−".

Умножение вводится нажатием клавиши "звездочка", причем в случае однократного нажатия этой клавиши (перед очередным сомножителем) фактической прорисовки чего-либо на экране не происходит. Если нужно убедиться в том, что нажатие клавиши и ввод операции умножения состоялись, то клавиша "звездочка"должна быть нажата еще дважды - в этом случае для умножения будет прорисована точка. Следует особенно аккуратно вводить произведения, часть сомножителей которых заключена в скобки. Если пропустить нажатие клавиши "звездочка"после A, то произведение $A(B+C)$ будет воспринято как значение функции A в точке $B+C$.

Дробное выражение $\frac{A}{B}$ вводится следующим образом. Сначала нажимается клавиша "двоеточие" - после этого прорисовывается красным цветом горизонтальная черта дроби, над которой размещается курсор. Затем вводится числитель A. В процессе ввода горизонтальная черта дроби удлиняется автоматически; если вводится многоэтажное выражение, то для обеспечения необходимого места происходит автоматическая коррекция размеров и размещения всей ранее введенной части формулы. После ввода числителя нажимается клавиша "Enter" - тогда курсор перемещается в начало знаменателя, и предпринимается ввод знаменателя B. Наконец, после ввода знаменателя нажимается клавиша "Enter", завершающая ввод дроби. При этом горизонтальная черта дроби из красной становится черной, и происходит автоматическая центровка числителя и знаменателя для получения симметричной записи.

Степенное выражение A^B вводится следующим образом. Сначала набирается выражение A (если оно само является результатом применения более чем одноместной операции, то обязательно заключается в скобки). Затем нажимается клавиша "курсор вверх" - курсор поднимается вверх для прорисовки показателя степени. После ввода выражения B (его не обязательно заключать в скобки) нажимается "Enter" - курсор опускается обратно, и ввод степени считается законченным.

Отношения "меньше","больше"вводятся с помощью клавиш $<$, $>$; отношения "меньшеилиравно","большеилиравно" - с помощью клавиш "левая квадратная скобка", "правая квадратная скобка". Для ввода утверждения "A - число"сначала набирается выражение A, затем нажимаются клавиши "/"и "ч".

Конечная сумма

$$\sum_{i=a}^{b} f(i)$$

вводится следующим образом. Сначала нажимается "Ctr-s" - появляется большая буква "сигма"малинового цвета, причем курсор находится под этой буквой. Здесь набирается $i=a$ и нажимается "Enter". Тогда курсор переводится в положение над буквой "сигма", где набирается выражение b и снова нажимается "Enter". После этого курсор оказывается справа от буквы "сигма", где набирается выражение $f(i)$. По окончании набора нажимается "Enter", перекрашивающее букву "сигма"в черный цвет - набор суммы на этом закончен. Если требуется задать множество допустимых значений индекса суммирования косвенным образом, то используется запись

$$\sum_{i,P(i)} f(i).$$

Ввод ее происходит аналогично предыдущему, но в положении курсора над "сигмой" сразу нажимается "Enter".

Конечные произведения

$$\prod_{i=a}^{b} f(i), \qquad \prod_{i,P(i)} f(i)$$

вводятся совершенно так же, как конечные суммы, но вместо клавиши "Ctr-s"нажимается клавиша "Ctr-p"(здесь p - латинское).

Элементарные функции

Для ввода квадратного корня $\sqrt{A}$ последовательно нажимаются клавиши "s","r"(появляется радикал малинового цвета), вводится выражение A и нажимается "Enter" - радикал перекрашивается в черный цвет. Если нужно ввести корень $\sqrt[n]{A}$, где n - целое от 3 до 9, то последовательно нажимаются клавиши "r","t"(появляется малиновый радикал, над которым расположен курсор), "n"(курсор переводится под радикал), и далее - как для квадратного корня.

Для ввода максимума либо минимума выражений $A_1, \ldots, A_n$ последовательно нажимаются, соответственно, латинские клавиши "m","a"либо "m","i", и далее в скобках перечисляются указанные выражения. Для ввода модуля выражения A сначала нажимается "Ctr-m", затем вводится A, затем снова нажимается "Ctr-m".

Выражение $\log_a b$ набирается следующим образом. Сначала последовательно нажимаются клавиши "l","o"(лат.). Затем набирается основание логарифма a. Далее нажимается "Enter" - курсор переводится в позицию справа от логарифма, и набирается выражение под логарифмом b. Если оно представляет собой многоместную операцию, то его обязательно следует заключить в скобки - иначе под логарифмом окажется лишь первый операнд этой операции. Это же замечание о скобках относится и ко всем приводимым ниже элементарным функциям. В случае натурального логарифма достаточно нажать клавиши "l","n"и ввести выражение под логарифмом.

Экспонента $\exp(A)$ вводится последовательным нажатием клавиш "e","x"(лат.).

Синус $\sin(A)$ вводится последовательным нажатием клавиш "s","i". Заметим, что степень синуса вводится не совсем стандартным образом: необходимо сначала набрать все выражение $\sin(A)$ (для большей наглядности, хотя и необязательно, заключенное в скобки), и лишь затем нажать "курсор вверх", переходя к вводу показателя степени. Помещать показатель степени сразу же после sin нельзя. Это же замечание относится и к другим элементарным функциям. Если аргумент синуса степенной, то его обязательно нужно помещать в скобки - иначе показатель степени будет отнесен к синусу. Это же замечание относится и к остальным тригонометрическим функциям.

Оставшиеся элементарные функции вводятся однотипным с синусом образом - последовательным нажатием двух либо трех латинских клавиш (в ряде случаев, как указано ниже, требуется вводить не малую, а большую букву). Мы просто перечислим для них эти пары либо тройки клавиш: косинус - "c","o"; тангенс - "t","g"; котангенс - "c","t"; секанс - "s","e"; косеканс - "c","s"; арксинус - "a","s"; арккосинус - "a","c","o"; арктангенс - "a","t"; арккотангенс - "a","c","t"; сигнум (минус единица для отрицательных, не определен в нуле, единица для положительных) - "s","g"; сигнум (0 для отрицательных, 1 для неотрицательных) - "S","g"; гиперболический синус - "s","h"; гиперболический косинус - "c","h"; гиперболический тангенс - "t","h"; гиперболический котангенс - "c","T".

Символьные константы

Константа "e" вводится двойным нажатием латинской клавиши "e" (важно выполнять эту операцию аккуратно, чтобы не получить вместо константы "e" переменную "e"; по внешнему виду они несколько отличаются друг от друга). Константа "пи"вводится двойным нажатием латинской клавиши "p". Константа "плюс-бесконечность"водится двойным нажатием клавиши "i". Для получения минус-бесконечности перед плюс-бесконечностью помещается знак "минус"(хотя во внутреннем представлении минус-бесконечность представлена отдельным логическим символом).

Мнимая единица вводится двукратным нажатием клавиши "c".

Операции и отношения для целых чисел

Утверждения "A - целое", "A - натуральное", "A - рациональное", "A - четное"вводятся следующим образом: сначала набирается выражение A, затем нажимается "/" , и далее, соответственно, клавиша "ц"(целое) либо "н"(натуральное), либо "q"(рациональное), либо "е"(четное;лат.).

Выражения "числитель(A)"и "знаменатель(A)"вводятся последовательным нажатием клавиш "ч","и"либо, соответственно, "з","н"(далее набирается A, при необходимости - в скобках).

Утверждение "$A|B$"(A делит B) вводится в следующей последовательности: сначала набирается A, затем нажимается "Ctr-д", затем B.

Выражения "нод(A, B)" и "нок(A, B)" (наибольший общий делитель и наименьшее общее кратное) вводятся, соответственно, последовательным нажатием клавиш "н", "д" либо "н","к"(кир.) и дальнейшим набором (в скобках и через запятую выражений A,B.

Аналогично вводятся утверждения "простое(A)" (двойное нажатие "п") и "взаимнопросты(A, B)" (клавиши "в","п").

Целая часть $[A]$ выражения A набирается следующим образом: сначала нажимается клавиша "Ctr-e"(e - лат.), что приводит к появлению левой квадратной скобки. Затем вводится A и снова нажимается "Ctr-e для правой квадратной скобки.

Для вычета $A(modB)$ сначала набирается выражение A, затем открывающая скобка, затем нажимаются клавиши "м","д"(кир.), затем набирается B, и в конце - закрывающая скобка.

Число сочетаний из n по k вводится следующим образом. Сначала нажимается "Ctr-c", что приводит к появлению большой буквы "C"малинового цвета; курсор находится справа в нижней части этой буквы. Затем набирается n и нажимается "Enter" - курсор перемещается в верхнюю часть справа от буквы "C". Далее набирается k и снова нажимается "Enter" - буква перекрашивается в черный цвет и ввод числа сочетаний завершается. При необходимости в процессе ввода вертикальные размеры буквы "C"автоматически увеличиваются.

Факториал $n!$ вводится последовательным набором n (при необходимости - в скобках) и нажатием "Ctr-f".

Многочлены

Многочлен P над полем вещественных чисел (см. пояснение про многочлены из предыдущего раздела) вводится следующим образом. Сначала дважды нажимается клавиша "m"(лат.) - появляется греческая буква "мю", справа от которой и чуть ниже располагается курсор. Далее вводятся все переменные, от которых формально зависит многочлен, и нажимается "Enter" - курсор поднимается вверх до уровня буквы "мю". После этого набирается сумма одночленов, определяющая значения многочлена, и в конце ставится закрывающая скобка.

Алгебра множеств

Утверждение "A - множество"вводится последовательным набором A, нажатием "/"и "s". Символ $\emptyset$ пустого множества вводится последовательным нажатием клавиш "e","m"(лат.). Символ $\cup$ объединения множеств вводится последовательным нажатием клавиш "пробел", "u", "s"; символ $\cap$ пересечения множеств - нажатием "пробел", "i", "s". Для ввода символа $\setminus$ разности множеств нажимается клавиша "\". Символ $\in$ принадлежности множеству вводится последовательным нажатием клавиш "пробел", "b", "e"(лат.); символ $\subset$ включения множеств - "пробел", "s", "u". Символ $\times$ прямого произведения множеств вводится нажатием клавиш "пробел", "p", "s"(лат.).

Конечное множество $a_1, \ldots, a_n$, заданное перечислением своих элементов, вводится путем последовательного набора этих элементов (через запятую) внутри фигурных скобок.

Конечное объединение

$$\bigcup_{i=a}^{b} f(i)$$

вводится следующим образом. Сначала нажимается "Ctr-u" - появляется большой знак "объединение"малинового цвета, причем курсор находится под ним. Здесь набирается $i = a$ и нажимается "Enter". Тогда курсор переводится в положение над "объединением", где набирается выражение b и снова нажимается "Enter". После этого курсор оказывается справа от "объединения", где набирается выражение $f(i)$. По окончании набора нажимается "Enter", перекрашивающее знак "объединение"в черный цвет - его набор закончен. Если требуется задать множество допустимых значений индекса объединения косвенным образом, то используется запись

$$\bigcup_{i,P(i)} f(i).$$

Ввод ее происходит аналогично предыдущему, но в положении курсора над "объединением"сразу нажимается "Enter".

Конечные пересечения

$$\bigcap_{i=a}^{b} f(i), \qquad \bigcap_{i,P(i)} f(i)$$

вводятся совершенно так же, как конечные объединения, но вместо клавиши "Ctr-u"нажимается клавиша "Ctr-x"(здесь x - латинское).

Класс $set_x P(x)$ всех объектов (если x - список переменных, то наборов) x, удовлетворяющих условию $P(x)$, вводится следующим образом. Сначала дважды нажимается клавиша "S" - появляется "*set*". Затем вводится переменная либо список переменных x; нажимается "Enter"и набирается условие $P(x)$.

Мощность $card(A)$ множества A вводится нажатием клавиш "c","a"(лат.) и набором выражения A. Заголовки утверждений "конечное(A)", "непересек(A, B)", "семействомножеств(A)", "разбиение(A, B)"вводятся, соответственно, последовательными нажатиями пар клавиш (везде кириллица): "к","о"; "н","п"; "С","м"; "р","а". Заголовок выражения "подмножества(A)"вводится последовательным нажатием клавиш "П","м".

Мощности счетного множества и континуума вводятся, соответственно, нажатиями клавиш "c","ч"и "к","м". Символ пустого слова вводится нажатием клавиши "Ctr-щ".

Утверждения "убывмножества(A)", "возрастмножества(A)"о том, что последовательность множеств A является убывающей либо возрастающей по включению, вводятся последовательными нажатиями клавиш "У","М"и "В","М"(кир.).

Числовые множества

Для ввода промежутка числовой оси с концами A, B (они могут являться символами бесконечности) сначала нажимается "Ctr-("(если конец A не относится к промежутку)либо "Ctr-["(если этот конец относится к промежутку). Затем набираются отделенные запятой выражения A, B. Если конец B не относится к промежутку, то далее нажимается "Ctr-)", иначе - "Ctr-]".

Все множество вещественных чисел вводится двойным нажатием клавиши "R". Множество рациональных чисел вводится двойным нажатием клавиши "Q"; множество целых чисел - двойным нажатием "Z"; множество натуральных чисел - двойным нажатием "N"; множество целых неотрицательных чисел - последовательным нажатием "Ц","Н"(кир.).

Конечный отрезок $\{A, \ldots, B\}$ натурального ряда, начинающийся с A и кончающийся B, набирается следующим образом: левая фигурная скобка, выражение A, запятая, "Ctr-точка", запятая, выражение B, правая фигурная скобка

Точная нижняя грань $\inf A$ множества A вводится с помощью нажатия клавиш "I","n"; точная верхняя грань $\sup A$ - с помощью "S","u".

Утверждения "нижняягрань(x, A)" (x - нестрогая нижняя грань числового множества A), "Нижняягрань(x, A)" (строгая нижняя грань), "верхняягрань(x, A)" (нестрогая верхняя грань), "Верхняягрань(x, A)" (строгая верхняя грань), "наибольший(x, A)" (x - наибольший элемент числового множества A), "наименьший(x, A)"вводятся, соответственно, с помощью последовательных нажатий пар клавиш (везде кириллица): "н","г"; "Н","г"; "в","г"; "В","г"; "Н","О"; "Н","Е". Утверждения "огрснизу (A)", "огрсверху(A)"об ограниченности числового множества A снизу либо сверху вводятся с помощью нажатий пар клавиш "г","н"и "г","в".

Выражения "внутренность(A)", "граница(A)", "замыкание(A)"вводятся, соответственно, при помощи нажатий пар клавиш "в","н"; "Г","р"; "З","З"(везде кир.).

Простейшие обозначения, связанные с функциями

Утверждение "A - функция"вводится последовательным набором A, нажатием "/"и "ф". Область определения $Dom(f)$ функции f вводится с помощью последовательного нажатия клавиш "d","o"(лат.). Множество значений $Val(f)$ функции f вводится с помощью последовательного нажатия клавиш "v","a". Для ввода значения $f(a)$ функции f в точке a сначала набирается обозначение функции f (если оно не односимвольное, то заключается в скобки), затем левая скобка, a и правая скобка. Выражения "образ(f, A)" (образ множества A), "прообраз(f, A)" (прообраз множества A) и "слой(f, a)" (полный прообраз элемента a) набираются при помощи последовательных нажатий пар клавиш "о","м"; "п","м"; "с","л"(везде - кириллица). Утверждение "взаимнооднозначно(f)" (f инъективно) вводится при помощи последовательного нажатия клавиш "в","о" (кир.). Множество корней числовой функции f на множестве A обозначается $roots(f, A)$ и вводится при помощи последовательного нажатия клавиш "r","o"(лат.).

Функция, значения которой определяются выражением t, а условие на принадлежность аргумента x (этот аргумент может быть как единственной переменной, так и набором переменных) области определения есть $P(x)$, обозначается $\lambda_x(t, P(x))$. Для ввода ее сначала дважды нажимается клавиша "L" - прорисовывается "лямбда", причем курсор располагается справа от нее и чуть ниже. Затем вводится x (переменная либо группа переменных, без запятых), нажимается "Enter"(курсор перемещается вверх на уровень "лямбды") и набираются в скобках $t, P(x)$.

Утверждение "Отображение(f, A, B)"(f есть отображение A в B) вводится при помощи двукратного нажатия клавиши "о"(кир.).

Функция "конст(A, b)", определенная на множестве A и принимающая во всех его точках значение b, вводится при помощи двукратного нажатия "к"(кир.). Функция "$a \to b$", определенная на одноэлементном множестве, состоящем из элемента a, и принимающая на этом элементе значение b, вводится следующим образом: сначала набирается a, затем "Ctr-курсор вправо", затем b. Тождественная функция "тождфунк(A)" с областью определения A вводится при помощи нажатия клавиш "т","ф".

Посредством "таблица$(A_1, \ldots, A_n)$"обозначается функция, график которой есть объединение графиков функций $A_1, \ldots, A_n$, принимающих на пересечениях своих областей определения одинаковые значения. Символ "таблица"вводится последовательным нажатием клавиш "т","а"(кир.).

Выражения "сужение(f, A)" (сужение функции f на множество A) и "доопределение(f, A, b)" (доопределение функции f на тех точках множества A, где она еще не определена, значением b) вводятся при помощи последовательных нажатий клавиш "с","у"и "д","о"(кир.). Выражение "обрфункция(f)"(функция, обратная к взаимнооднозначной функции f) вводится при помощи последовательного нажатия "о","ф".

Выражение "числопеременных(f)"вводится при помощи клавиш "ч","п"; выражение "подфункиця(f, A, B)"(функция, полученная из f подстановкой элементов набора B вместо переменных, номера которых образуют набор A; нумерация начинается с 1) вводится при помощи клавиш "п","Ф". Утверждение "существперем(i, f)"
(i-я переменная функции f является существенной) вводится при помощи клавиш "С","У".

Упорядоченные наборы

Всюду далее понимаем под "набором"только упорядоченный набор (вектор). Фактически "набор" здесь - это отображение начального отрезка натурального ряда на некоторое множество. Набор элементов $a_1, \ldots, a_n$ вводится простым перечислением через запятую этих элементов. Этот набор при вводе можно заключать в скобки, а можно и не заключать. Однако, для ввода одноэлементного набора ($n = 1$) необходимо использовать другой способ - сначала нажать "Ctr-н"(кир.) и затем ввести a_1; если a_1 неоднобуквенное, то оно должно быть заключено в скобки.

При вводе выражений "префикс(A, B)"(добавление элемента A в начале набора B) и "суффикс(A, B)"(добавление элемента A в конце набора B) логические символы "префикс","суффикс" набираются текстовым редактором (вход в него - через "Ctr-Enter"). Исключение здесь составляет случай выражений "перечень(префикс$(A\,B)$)", которые прорисовываются в виде $\{A; B\}$ и набираются с помощью клавиши ";". Инфиксная операция конкатенация (последовательное соединение наборов) также вводится при помощи фигурных скобок и клавиши ";". Соответственно, запись $\{A, B, C; D\}$ обозначает конечное множество, перечисляемое конкатенацией наборов (A, B, C) и D.

Всюду далее предполагаем, что нумерация элементов набора начинается с 1. Выражения "поднабор(A, i, j)"(поднабор набора A, образованный элементами начиная с i-го и кончая j - м) и "Вставка(A, i, b)" (результат вставки в набор A перед i-м элементом элемента b) вводятся при помощи последовательных нажатий клавиш "п","н"и "В", "с"(кир.). Выражение "выборка(A, B)"обозначает результат исключения всех разрядов набора A, номера которых не принадлежат множеству B. Логический символ "выборка"вводится текстовым редактором. Выражение "наложение(A, B, N)"обозначает такой результат перемешивания элементов наборов A,B (с сохранением относительного порядка элементов каждого из них), в котором элементы набора A расположены на местах, номера которых образуют набор N. Логический символ "наложение"вводится текстовым редактором.

Выражение "наборномеров(i, j)"(набор, образованный целыми числами начиная с i и кончая j) вводится при помощи последовательного нажатия клавиш "н","о"(кир.).

Выражения "упорядвозр(A, f)" (результат упорядочения элементов набора по возрастанию значений числовой функции f на этих элементах) и "упорядубыв(A, f)" вводятся при помощи последовательных нажатий клавиш "у","в"; "у","ы".

Математический анализ

Предел $\lim_{x\to a} f(x)$ вводится следующим образом. Сначала последовательно нажимаются клавиши "l", "i" - рисуется знак предела, а курсор оказывается справа от него и чуть ниже. Затем вводится переменная x, нажимается клавиша "курсор вправо"и набирается выражение a (в случае левого либо правого одностороннего предела набирается $a - 0$ либо $a + 0$; если нужно рассмотреть общую ситуацию, не уточняя тип предела, то вместо a набирается $a\backslash b$, где b - указатель типа предела: 0 - двусторонний, 1 - левый, 2 - правый). Далее нажимается "Enter" - курсор возвращается на уровень "предела", и вводится выражение $f(x)$. Если это выражение представляет собой сумму, произведение и т.п., то его обязательно следует заключить в скобки - иначе под знаком предела окажется лишь первый операнд.

Предел последовательности обозначается путем заключения выражения $f(x)$ в фигурные скобки: $\lim_{n\to\infty}\{f(n)\}$.

Верхний предел $\overline{\lim}$ и нижний предел $\underline{\lim}$ вводятся, соответственно, при помощи последовательного нажатия клавиш "L","i"и "l","I".

Используемые в асимтотических оценках обозначения $O(x)$,$o(x)$ вводятся, соответственно, при помощи двукратного нажатия клавиши "O"либо "o"(лат.).

Производная $\frac{df(x)}{dx}$ вводится как обычная дробь: сначала нажимается клавиша ":" ,затем "d", "звездочка", выражение $f(x)$, "Enter", "d", "звездочка", "x"и "Enter". Если требуется задать значение производной в точке a, то вместо dx в знаменателе помещается $d(x = a)$ (после d - обязательно нажимается звездочка). Заметим, что если a - снова переменная, то автоматически произойдет изменение обозначений, так что при повторной прорисовке вместо $\frac{df(x)}{d(x=a)}$ будет изображено $\frac{df(a)}{da}$. При рассмотрении производных высших порядков и частных производных используются обычные записи типа $\frac{d^3f(x)}{dx^3}$,$\frac{d^4f(x,y,z)}{dx^2dydz}$. Они вводятся аналогично первой производной; символ дифференцирования d рассматривается при этом как обычный множитель.

Утверждения "Производная(f, g)" (значения функции g в области ее определения суть значения производной функции f) и "Частнпроизв(f, i, g)" (значения функции g в области ее определения суть значения частной производной функции f по i-му аргументу) вводятся путем последовательных нажатий клавиш "П","р"и "Ч","п". Выражение "дифференциал(f, n, x, a)" (значение полного дифференциала порядка n функции f в точке x при наборе a значений приращений переменных) вводится путем последовательного нажатия клавиш "Д","и".

Неопределенный интеграл

$$\int f(x)dx$$

вводится следующим образом. Сначала нажимается "Ctr-j"(прорисовывается малиновым цветом знак интеграла), затем вводится подынтегральное выражение $f(x)$, нажимается "звездочка", "d", вводится переменная интегрирования x и нажимается "Enter" - знак интеграла из малинового становится черным.

Определенный интеграл

$$\int_a^b f(x)dx$$

вводится при помощи "Ctr-i". После этого курсор оказывается снизу от знака интеграла. Здесь набирается выражение a, затем нажимается "Enter" - курсор перемещается вверх от интеграла; набирается выражение b и снова нажимается "Enter". Далее - все как при наборе неопределенного интеграла.

Двойной интеграл

$$\int\int_P f(x,y)dxdy$$

вводится при помощи нажатия клавиши "Ctr-2". Курсор оказывается непосредственно под знаком двойного интеграла, где набирается выражение для области интегрирования P. Обычно здесь просто помещается вспомогательная переменная, а в посылках указывается равенство $P = M$, явно задающее область - иначе изображение интеграла становится чрезмерно громоздким. После ввода P нажимается "Enter" -

курсор перемещается вправо от знака интеграла. Здесь набирается подынтегральное выражение $f(x,y)$, "умноженное"на $dxdy$. В конце нажимается "Enter", перекрашивающее знак интеграла в черный цвет. Аналогичным образом вводится тройной интеграл - здесь используется клавиша "Ctr-3".

Ряд с общим членом $f(i)$ вводится как последовательность частичных сумм - $\lambda_n(\sum_{i=1}^{n} f(i), n -$ натуральное). Утверждение "сходится(f)"о сходимости последовательности (в частности, ряда; f - функция натурального аргумента, обычно вводимая через λ_n) вводится при помощи последовательного нажатия клавиш "с","д"(кир.). Сумма ряда с общим членом $f(i)$ обозначается обычным образом -

$$\sum_{i=m}^{\infty} f(i).$$

Правила набора здесь те же, что и для конечных сумм.

Утверждения "переменазнака(f,a)"(функция f меняет знак в окрестности точки a), "убываетвточке(f,a)", "возрастаетвточке(f,a)", "огрвточке(f,a)" (функция f ограничена в окрестности точки a) вводятся, соответственно, при помощи последовательных нажатий клавиш "п","з"; "У","т"; "В","т"; "О","Т"(кир.).

Запись $x \to a\backslash b$, уже встречавшаяся в связи с пределами, может использоваться как независимая посылка, для указания на то, что задача должна решаться в сколь угодно малой окрестности точки a ($\backslash b$ указывает тип окрестности и может отсутствовать либо быть замененным на $+0,-0$).

Утверждение "ограничено(A)"используется в контексте некоторого списка посылок и означает, что в области истинности этих посылок выражение A ограничено по модулю некоторой константой. Логический символ "ограничено"вводится последовательным нажатием клавиш "О","Г". Обычно это утверждение не встречается в формулировке задачи, а вводится при необходимости решателем в процессе решения.

Утверждения $Max(f,A,B,c)$ (функция f достигает на множестве A максимума c в точках подмножества B), $Min(f,A,B,c)$ вводятся при помощи последовательных нажатий клавиш "М","а"; "М","i" (лат.). Утверждение $Extr(f,a,b,c)$ (функция f имеет в точке a экстремум; b - значение ее в этой точке, а c - тип экстремума) вводится нажатием клавиш "Е","х"(лат.).

Выражения "стационарныеточки(f)"(точки из внутренности области определения функции, в которых все частные производные равны нулю) и "особыеточки(f)" (точки из внутренности области определения, в которых хотя бы одна частная производная не определена) вводятся при помощи клавиш "с","ц"; "о","ч".

Утверждения "Выпуклавверх(f,A)" (функция f выпукла вверх на множестве A) и "Выпуклавниз(f,A)" вводятся при помощи нажатий клавиш "в","в"; "в","з"(кир.). Утверждения "убывает(f,A)"(функция f строго убывает на множестве A), "возрастает(f,A)", "неубывает(f,A)", "невозрастает(f,A)"вводятся при помощи последовательных нажатий клавиш "У","ы"; "В","о"; "Н","у"; "Н","в".

Утверждения "четнаяфункция(f)", "нечетнаяфункция(f)", "периодична(f,T)" (функция f периодична на числовой оси и имеет периодом, не обязательно наименьшим, число T) вводятся при помощи нажатий клавиш "ч","ф"; "н","ф"; "п","Е" (кир.).

Утверждения "непрерывно(f, A)"; "равномернонепрерывно(f, A)" вводятся при помощи "н","н"; "р","н"(кир.). Утверждения "устранимыйразрыв(f, a)" (функция f имеет в точке a устранимый разрыв), "разрывпервогорода(f, a)", "разрыввторогорода(f, a)" вводятся с помощью текстового редактора.

Операции "умножфунк", "плюсфунк", "минусфунк"умножения, сложения и изменения знака числовых функций с общей областью определения вводятся формульным редактором так же, как обычные операции умножения, сложения и изменения знака чисел. Формульный редактор пытается усмотреть из контекста, что значениями операндов служат функции, и автоматически заменяет операции над числами на соответствующие операции над функциями. В тех случаях, когда усмотрение функционального типа операндов из контекста сомнительно, следует контролировать результат ввода, переходя к просмотру набранного терма в текстовом редакторе и осуществляя указанные коррекции вручную.

Планиметрия

В планиметрии фигуры и их числовые характеристики обозначаются только через соответствующие "опорные"точки ("прямая(AB)", "треугольник(ABC)"и т.п.); использование вспомогательных переменных для обозначения самих фигур ("прямая A"; "треугольник B"и т.п.) в приемах решателя не предусмотрено. В стереометрии, наоборот, тела обозначаются не с помощью их "опорных"точек, а с помощью вспомогательных переменных ("куб(A)", "пирамида(B)"и т.п.).

Утверждение "A-точка"вводится нажатием, после набора A, клавиш "\" и "р" (лат.). Обычно такие утверждения вообще не вводятся вручную, а создаются решателем самостоятельно в начале решения задачи.

Выражения "прямая(AB)", "отрезок(AB)", "интервал(AB)", "луч(AB)", "обратныйлуч(AB)" (луч с началом в точке A, противоположный лучу AB) вводятся последовательными нажатиями пар клавиш "п","р"; "о","т"; "и","н"; "л","у"; "о","л" (везде -кириллица). Заметим, что между A и B здесь не ставится запятая, в отличие от обычных многоместных операций и отношений формульного редактора. Такие исключительные случаи перечисления операндов без запятой используются только в планиметрии, причем лишь для некоторых (явно указываемых далее) логических символов. В качестве A, B могут быть использованы либо буквенные переменные, либо переменные с индексом. Если имеется хотя бы одна переменная с индексом, то между операндами необходим разделитель - клавиша "звездочка"(при ее нажатии на экране ничего не меняется, но становится возможен ввод следующего операнда).

Расстояние $l(AB)$ между точками A, B вводится при помощи двукратного нажатия клавиши "l". Величина $\angle ABC$ угла ABC вводится при помощи нажатия клавиш "у","г". В обоих случаях запятые между операндами не ставятся. Если нужно обозначить не величину угла, а множество точек плоскости, лежащих внутри него (включая граничные точки), то используется обозначение "Угол(ABC)", вводимое клавишами "У","г".

Выражения "расстдопрямой(A, B)"(расстояние от точки A до прямой B), "расстмежду(A, B)"(расстояние между двумя множествами точек A, B вводятся при помощи клавиш "Р","П"; "Р","Ы". Здесь уже нужна запятая между операндами.

Утверждение "биссектриса$(ABCD)$"(луч BD есть биссектриса угла ABC) вводится двойным нажатием клавиши "и". Запятая между операндами не ставится.

Утверждения "$a\|b$"(прямые либо плоскости a, b параллельны) и "$a\perp b$"(прямые либо плоскости a, b перпендикулярны) вводятся, соответственно, при помощи клавиш "Ctr-="и "Ctr-т"(т - кир.).

Утверждения "однасторона(A, B, a)" (точки A, B лежат по одну сторону от прямой либо плоскости a, возможно, попадая на a), "разныестороны(A, B, a)" (здесь тоже допускается попадание точек на a), "симметричны(A, B, a)" (точки A, B расположены симметрично относительно точки, прямой либо плоскости a) вводятся при помощи клавиш "О","С"; "Р","С"; "с","и"(кир.). Выражения "полоса(A, B)"(полоса между параллельными прямыми A, B), "полуплоскость(A, B)" (полуплоскость, ограниченная прямой A и содержащая точку B, не лежащую на B), "обрполуплоскость (A, B)" (полуплоскость, ограниченная прямой A и не содержащая точку B, не лежащую на прямой B) вводятся при помощи нажатий клавиш "п","с"; "п","П"; "о","П".

Для уточнения того, что вводимая задача относится к планиметрии - все рассматриваемые в ней точки лежат в общей плоскости - используется вспомогательная посылка "планиметрия". Обычно она вводится решателем автоматически после набора задачи (перед ее решением); вручную ее можно ввести последовательным нажатием клавиш "п","и".

Утверждения "$\triangle(ABC)$" (точки A, B, C образуют вершины треугольника), "параллелограмм$(ABCD)$" (точки A, B, C, D, взятые в данной последовательности, образуют вершины параллелограмма); "ромб$(ABCD)$"; "прямоугольник$(ABCD)$", "квадрат$(ABCD)$", "четырехугольник$(ABCD)$" (точки A, B, C, D образуют вершины выпуклого четырехугольника) вводятся последовательными нажатиями пар клавиш "т","р"; "п","а"; "р","о"; "п","р"; "к","в"; "ч","е" (кир.). Запятые между операндами не ставятся.

Утверждение "трапеция$(ABCD)$" (точки A, B, C, D образуют вершины трапеции, углы при большем основании которой - не тупые, причем точки A, D лежат на большем основании) вводится при помощи клавиш "т","п". Утверждение "Трапеция $(ABCD)$" (то же, но углы при большем основании - любые, а точки A, D лежат на основании, которое не обязательно большее) вводится при помощи клавиш "Т","п".

Утверждения "многоугольник(A)"(A - набор вершин простой замкнутой ломаной), "правмногоугольник(A)" (A - набор вершин правильного многоуольника) вводятся последовательными нажатиями клавиш "м","н"; "п","й". Множество точек внутри многоугольника (включая границу), определяемого набором вершин A, обозначается "фигура(A)". Точки набора A во всех перечисленных случаях отделяются друг от друга запятыми. Выражение "Фигура$\{A, B, \ldots, C\}$" (множество точек, ограниченное составной границей с частями $A, B, \ldots, C$) вводится при помощи клавиш "Ф","и".

Уиверждения "Медиана$(ABCD)$", "Высота$(ABCD)$", "Биссектреуг$(BACD)$" (точка D является основанием, соответственно, медианы, высоты либо биссектрисы треугольника ABC, проведенной из вершины A) вводятся при помощи пар клавиш "М","Е"; "В","Ы"; "И","Т".

Выражения "окружность(AB)", "круг(AB)"(окружность и круг с центром в точке A и точкой на окружности B) вводятся нажатием клавиш "о","к"; "к","р". Выражения "дуга(ABC)"(меньшая из дуг окружности с центром в точке A и концами B, C); "большаядуга(ABC)" (большая из указанных дуг); "дугаугла(ABC)" (дуга, на которую опирается вписанный угол ABC); "сектор(ABC)" (сектор окружности с центром в точке A и концами дуги B, C; берется меньшая из дуг); "сегмент(ABC)"

вводятся при помощи нажатий клавиш "д","у"; "д","У"; "д","в"; "С","е"; "с","г". Выражение "кольцо(ABC)" (кольцо с центром в A, внутренняя окружность которого проходит через B, а внешняя - через C) вводится при помощи "к", "О".

Площадь $S(A)$ плоской фигуры A (например, A может иметь вид "фигура(B)", "круг(BC)", "сектор(BCD)"и т.п.) вводится при помощи двукратного нажатия клавиши "S". Выражение "периметр(A)"(A обычно имеет вид "фигура(B)") вводится при помощи последовательного нажатия клавиш "п","е". Выражение "длина(A)" (длина кривой A) вводится при помощи "д","л".

Утверждение "центр(A, B)"(точка A является центром фигуры B) вводится при помощи "ц","е".

Утверждение "A - касательная к B" вводится набором A, нажатием "Ctr-q"и набором B. Утверждения "внешкасательная(A, B, C)" (прямая A является внешней касательной к окружностям B, C); "внутркасательная(A, B, C)" вводятся нажатиями "Ш","к"и "У","к"соответственно.

Утверждения "окружность(AB)вписана в фигура($C_1 \dots C_n$)"; "окружность(AB) описана около фигура($C_1 \dots C_n$)" вводятся нажатием клавиш "Ctr-э"и "Ctr-ф" соответственно (предварительно вводится "окружность(AB)").

Утверждения "внешкасаются(A, B)" (внешнее касание окружностей) и "внутркасаются(A, B)" вводятся последовательными нажатиями клавиш "ш","к"; "у","к".

Утверждения "выпукло(A)", "осьсимметрии(A, B)" вводятся при помощи клавиш "в","ы"; "С","и".

Утверждения $A \sim B$ (подобие фигур) и $A \equiv B$ (конгруентность фигур) вводятся при помощи клавиш "Ctr-п" и "Ctr-к" (кир.).

Стереометрия

Выражение "плоскость(ABC)" (плоскость, проходящая через точки A, B, C) вводится при помощи клавиш "п","л". Для ввода выражения "плоскостьфигуры(a)" (плоскость, в которой расположена плоская фигура a) используются клавиши "п","ф". Выражение "уголмежду(a, b)" (величина острого угла между прямыми либо плоскостями a, b) вводится при помощи "у","м".

В стереометрии для задания окружности мало указания ее центра A и точки B на окружности - нужно задать еще плоскость окружности, для чего в решателе используется ссылка на некоторую третью точку C, лежащую в данной плоскости. Соответственно, возникают выражения "Окружность(ABC)"и "Круг(ABC)", для ввода которых используются клавиши "О","к"и "К","р".

Утверждение "биссектрплоск(C прямая(AB) D E)"(биссекторная плоскость двугранного угла, опирающегося на прямую AB и ограниченного двумя полуплоскостями, содержащими, соответственно, точки C и D, проходит через точку E) вводится при помощи двукратного нажатия клавиши "И". Утверждение "плоскостьсимметрии
(A, B)" (A есть плоскость симметрии поверхности либо тела B) вводится клавишами "П","И".

Выражение "объем(a)" вводится двукратным нажатием клавиши "О"(кир.). Выражения "площповерхности(a)", "боковаяповерхность(a)" (площадь боковой поверхности тела a), "высота(a)"(высота тела a), "радиус(a)"(радиус сферы либо шара a) вводятся, соответственно, при помощи пар клавиш "щ","п"; "Щ","п"; "В","ы".

Утверждения "грань(a,b)" (фигура a является гранью многогранника b), "основание(a,b)" (фигура a является основанием многогранника b) вводятся при помощи клавиш "г","р"; "О","с"(кир.). Утверждения "ребро(a,b)" (отрезок a есть ребро многогранника b); "вершина(a,b)" (точка a есть вершина многогранника b); "Вершина(a,b)" (точка a является вершиной пирамиды либо конуса b); "диагональ(a,b)" (отрезок a есть главная диагональ параллелепипеда b) вводятся парами клавиш "р","е"; "в","ш"; "В","ш"; "д","и". Утверждение "осевоесечение(a,b)" (прямоугольник либо треугольник a является осевым сечением, соответственно, цилиндра либо конуса b) вводится клавишами "С","Е".

Утверждения "куб(a)", "призма(a)", "параллелепипед(a)", "пирамида(a)", "усечпирамида(a)", "конус(a)", "шар(a)", "сфера(a)"вводятся, соответственно, при помощи нажатий пар клавиш "к","у"; "з","м"; "п","д"; "р","м"; "к","с"; "ш","а"; "ф","р".

Утверждения "прямой(a)" (прямая призма, прямой параллелепипед); "правильный(a)" (правильная призма, правильная пирамида, правильная усеченная пирамида); "прямоугольный(a)" (параллелепипед) вводятся при помощи пар клавиш "п", "Я"; "п","в"; "п","я".

Выражение "трехгранугол($ABCD$)" (A - вершина трехгранного угла; AB,AC,AD - направляющие векторы сторон) вводится при помощи "т","у". Выражения "двугрУгол(A,B,C,d)" (двугранный угол между плоскостями A,B, выделяемый при помощи точки C и указателя d ее принадлежности: 0 - внутри угла, 1 - внутри угла, смежного с данным через A, 2 - внутри угла, смежного с данным через B, 3 - внутри угла, вертикального с данным), "двугругол(A,B,C,d)" (величина указанного выше двугранного угла), "двугранУгол(A,B,C)" (двугранный угол, в основании которого лежит прямая B, а направления сторон определяются лежащими на них точками A,C), "двугранугол(A,B,C)"(величина указанного выше двугранного угла) вводятся при помощи пар клавиш "Д","У"; "Д","у"; "Д","Г"; "Д","г".

Выражения "полупространство(A,B)" (полупространство, отделенное плоскостью A и содержащее точку B) и "обрполупространство(A,B)"(полупространство, отделенное плоскостью A и не содержащее точки B) вводятся при помощи клавиш "П","В"; "О","П". Выражение "Полоса(A,B)" (полоса между параллельными плоскостями A,B) вводится при помощи клавиш "П","С".

Упражнения по вводу утверждений и выражений

Прежде, чем перейти к упражениям, разберем один простой пример. Будем вводить выражение $(a+b)^2$. Обычно утверждения и выражения вводятся при создании задачи. Для наших упражнений создадим в задачнике особый раздел.

Из главного меню входим в оглавление задачника (клавиша "з"). Используя клавишу "курсор влево", быть может, несколько раз, попадаем в корневое меню оглавления. В этом меню выбираем, например, первый пункт - "Дискретная математика". Зайдя в него нажатием "курсор вправо", далее нажимаем клавишу "м". Появляется новый

пункт меню, после которого расположен курсор текстового редактора. Вводим название пункта - например, "Упражения", после чего нажимаем Enter. Если название пункта требуется изменить, нажатие "р" возвращает в его редактирование.

Заходим в новый пункт меню (курсор "вправо") и нажимаем клавишу "к". Это приводит к созданию в подменю "Упражнения" единственного концевого пункта - бланка новой задачи. В верхней части экрана прорисована горизонтальная полоса - отделяющая линия задачи, а под ней - пустой экран.

Теперь можно переходить ко вводу формулы. Нажимаем Enter, и на экране возникает курсор формульного редактора. Дальнейшие действия - согласно приведенным выше описаниям действий клавиш. При необходимости, в процессе набора нажимается F1, переводящее в оглавление команд формульного редактора. Возвращение в набор формулы - по клавише End.

В нашем примере последовательно нажимаем клавиши "(", "a", "+", "b", ")". На экране появляется "$(a+b)$". Для ввода показателя степени нажимаем "курсор вверх". Курсор перемещается на позицию, где будет вводиться показатель. В нашем примере нажимаем клавишу "2". Чтобы вернуть курсор на исходну, нижнюю строку, нажимаем Enter. Так как этим набор формулы завершен, нажимаем Enter еще раз. Курсор пропадает, а на экране остается набранная формула. Можно выйти обратно в оглавление задачника нажатием "курсор влево". Там обнаружится единственный концевой пункт - номер 1, точка и три тире. Это указатель на нашу задачу. Если теперь нажать "курсор вправо", то на экране восстановится набранная ее часть.

Можно продолжить вводить формулы из приводимых ниже упражнений в той же задаче, каждый раз нажимая Enter для ввода очередной формулы. По окончании - выйти обратно в оглавление задачника, нажать Ctr-Del для удаления задачи с набранными формулами. Чтобы совсем расчистить оглавление задачника, еще раз нажать "курсор влево", и на выделенном пункте "Упражнения" опять нажать Ctr-Del.

В качестве упражнения рекомендуется ввести следующие формулы:

1. $(a^2 - b^2 - c^2 + 2bc)/((a + b - c)/(a + b + c))$
2. $(\sqrt{a^2 - 4} + a + 2)/(-\sqrt{a^2 - 4} + a + 2)$
3. $\log_{a^2+2ab+b^2}(a + b)$
4. $\sin(2a + b)/\sin(a) - 2\cos(a + b) - \sin b/\sin a$
5. $3(\sin a)^2(\cos a)^2 + (\sin a)^6 + (\cos a)^6$
6. $\arccos(\cos(6\pi/7))$
7. $\sum_{m=1}^{n} m/2^m$
8. $\log_2(1/(\cos(xy))^2 + (\cos(xy))^2) = 1/(y^2 - 2y + 2)$
9. $\forall_x(0 \le x \ \&\ x \le 2\pi/3 \ \&\ x - \text{число} \to ||\sin x - 1/3| - 1/3| \le a)$
10. $d(\sqrt{\text{tg}(1/a + a) + 1}/da$
11. $\lim_{x\to\infty}(x^x/(x^{(}x^x))$

12. $\mathrm{Max}(\lambda_x(2^x, x - \text{число}), [-1, 5], y, z)$
13. $\triangle(ABC)$
14. $\angle(ABC) = \pi/7$
15. $\text{прямая}(AB) \parallel \text{прямая}(CD)$
16. $\text{прямая}(AB) \perp \text{прямая}(CD)$
17. $x = S(\text{фигура}(ABCD))$
18. $\text{внешкасаются}(\text{окружность}(AC), \text{окружность}(BC))$
19. $(x + 1)dy(x)/dx + xy(x) = 0$
20. $\text{коорд}(\text{прямая}(AB), K) = \mathrm{set}_{xy}(y - \text{число}\ \&\ y = x/2\ \&\ x - \text{число})$

Глава 3

Логическая формализация задач

Следующий вопрос после выбора логического языка - формализация понятия задачи, достаточная для работы с обучающим материалом.

3.1 Основные типы задач и их представление в программе

На начальном этапе исследований по искусственному интеллекту предпринимались многочисленные попытки дать математически строгие определения понятия задачи, решения задачи и ответа на задачу. В самом общем виде, задача представлялась как логическое условие $P(x)$ на элементы x некоторого универсального множества объектов U; ответ на задачу - как утверждение логического языка, определяющее в некоторых "явных"терминах конкретный элемент x множества U, для которого $P(x)$ истинно; решение задачи - как цепочка преобразований исходного условия, преобразующая его к виду ответа. Вводя в универсум U классы объектов, в таком виде легко представить не только задачу поиска единственного элемента некоторого множества, удовлетворяющего заданному условию, но и задачу описания всех таких элементов.

Однако, это простое и, казалось бы, весьма общее представление о задаче сразу же сталкивается с трудностями при сопоставлении его даже с простейшими реальными задачами по элементарной алгебре. Так, при решении задачи на упрощение алгебраического выражения вовсе не накладывается какого - либо строгого ограничения $P(x)$ на предъявляемый школьником ответ x - например, не требуется предъявления доказательства того, что этот ответ минимален по числу символов или по какому-либо другому функционалу сложности выражения. Все, что в таких случаях требуется - это применять определенный запас стандартных приемов упрощения до тех пор, пока дальнейшие попытки не будут давать каких-либо улучшений, и выдать полученное выражение в качестве ответа.

Этот и другие примеры приводят к выводу, что более адекватным является представление о задаче, как о сочетании некоторого строгого условия на допустимость ответа с рядом целевых установок на его оптимизацию, учитываемых при решении по мере возможности. Тогда понятие ответа становится зависящим от уровня обученности системы.

Предварительная классификация "логических типов" задач, возникшая при анализе математических предметных областей, привела к рассмотрению 4 основных типов задач: на доказательство, на преобразование, на описание, и на исследование. Любая

такая задача имеет, прежде всего, некоторый (возможно, пустой) список утверждений относительно встречающихся в ней "известных" объектов, истинность которых считается априори данной. Эти утверждения будем называть посылками задачи. Типичные примеры списков посылок - перечисление условий на известные параметры в системе уравнений; логическое описание чертежа в геометрической задаче на вычисление; список "данных" утверждений в задаче на доказательство. В зависимости от типа задачи, список посылок сопровождается рядом дополнительных элементов.

В случае задачи на доказательство добавляется то утверждение, относительно которого требуется установить, что оно является следствием посылок; будем называть такое утверждение условием задачи на доказательство.

В случае задачи на преобразование добавляется то выражение (называемое ее условием), которое должно быть преобразовано к некоторому специальному виду в предположении истинности посылок. В отличие от задачи на доказательство, здесь возникает необходимость сформулировать целевую установку, уточняющую желаемые вид и оптимизацию ответа (упростить; разложить на множители; проинтегрировать, и т.п.). Эта формулировка уже не относится к тому логическому языку "предметного уровня", на котором задаются посылки и условие. Хотя ее можно было бы задавать на некотором логическом языке, предназначенном для записи утверждений о логических структурах данных, на практике оказалось более удобным представлять целевую установку задачи как список специальных технических "пометок", называемых далее целями задачи.

Задача на описание - это обобщение таких типов задач, как решение системы уравнений или неравенств. У нее, кроме списка посылок, имеется также некоторый список утверждений с неизвестными - они называются далее условиями задачи. Требуется дать описание всех или части значений неизвестных, при которых выполнены условия. Как и в случае задачи на преобразование, списки посылок и условий задачи на описание приходится сопровождать целевой установкой. Она уточняет вид искомого описания; определяет, нужно ли получить полное описание или достаточно лишь частичного (например, единственного примера значений неизвестных). Ответ задачи на описание обычно достигается в процессе последовательных преобразований ее списка условий. Однако, во многих случаях для получения ответа бывает необходимо накапливать некоторое многообразие следствий объединенного списка ее условий и посылок. В таком режиме решаются системы уравнений: извлекая следствия из исходных уравнений, иногда удается получить равенства, указывающие значения неизвестных. Этот же режим определения значений неизвестных путем вывода следствий типичен для геометрических задач на вычисление. Занесение следствий непосредственно в список условий задачи нежелательно, так как впоследствии пришлось бы расчищать этот список, исключая все избыточные его элементы, что потребовало бы дополнительных вычислительных затрат на проверку избыточности. Поэтому в структуре данных задачи на описание предусмотрен специальный накопитель следствий условий и посылок. Роль такого накопителя играет вспомогательная задача на исследование (см.ниже), вводимая в процессе решения задачи на описание. Изначально она имеет своими посылками все посылки и условия задачи на описание.

Задача на исследование, в дополнение к списку посылок, имеет лишь целевую установку, уточняющую направленность логического вывода в этом списке. При решении ее исходное логическое описание некоторой ситуации в том или ином смысле "упрощается" и пополняется утверждениями, представляющими интерес в контексте целевой установки. Этот процесс обрывается либо по исчерпании возможностей

добавления к имеющейся "картине" каких-либо ценных новых фактов, либо при получении следствий, требующих немедленного возвращения к внешней задаче, для которой предпринимается исследование.

Заметим, что в логической системе задачи на исследование используются только как вспомогательные - например, в роли накопителя следствий условий и посылок задачи на описание; в роли накопителя следствий при доказательстве от противного некоторого утверждения, и т.д. Различные математические задачи "на исследование" - такие, как исследование поведения функции вещественного переменного с помощью пределов и производных; исследование вида кривой или поверхности, заданной своим уравнением, и т.п., - оказалось целесообразно представлять в виде задачи на описание, сводя ее решение практически целиком к выводу следствий в связанной с ней задаче на исследование, и отбирая затем в качестве результата лишь некоторые из полученных фактов.

Кроме логических структур данных, задача содержит также некоторые вспомогательные технические структуры данных, вводимые в процессе работы самим решателем и используеыме для сохранения информации, направляющей его действия. Прежде всего, это пометки, указывающие на различные ранее предпринимавшиеся неудачные попытки, блокирующие их повторение, а также сообщения управляющего характера, которыми обмениваются между собой приемы.

Структуры данных, используемые для представления задачи в программе

Задача Z любого из четырех указанных выше типов представлена набором $(p_1, \ldots, p_n)$ следующих элементов:

Первый элемент p_1 есть логический символ - название типа задачи. Эти названия суть логические символы "доказать", "описать", "преобразовать"и "исследовать".

p_2 есть набор $(f_1, \ldots, f_m)$ утверждений, называемых посылками задачи. Посылки перечисляют априори известные ограничения на фигурирующие в задаче объекты - то, что считается в задаче "данным". Список посылок обязательно непуст; если никаких исходных допущений не делается, то он предполагается состоящим из логической константы "истина".

p_3 есть набор $(a_1, \ldots, a_m)$ целых неотрицательных чисел, называемых весами посылок (a_i - вес посылки f_i) и используемых для переключения внимания при поиске очередного приема; в исходной ситуации веса посылок равны 0.

p_4 - набор $(K_1, \ldots, K_m, K_0)$, элементы K_i которого при $i \neq 0$ суть наборы комментариев к i -й посылке, а K_0 есть набор комментариев ко всему списку посылок. Комментарии суть технические пометки, делаемые решателем в процессе работы для сохранения различной информации, направляющей ход решения. В исходной ситуации все K_i пусты. Компоненты p_1 - p_4 являются общими для задач всех перечисленных выше типов. Прочие компоненты определяются в зависимости от типа задачи следующим образом:

1. Задачи на доказательство. В этом случае $n = 7$. Элемент p_5 представляет собой утверждение, называемое условием задачи. Его истинность и требуется доказать, предполагая истинность посылок. p_6 - целое неотрицательное число, называемое весом условия задачи; как и веса посылок, оно используется для переключения внимания в процессе решения задачи. p_7 - последний элемент задачи на доказательство

- набор комментариев к задаче в целом. Они, как и комментарии к посылкам, представляют собой технические пометки, вводимые решателем в процессе работы.

Решая задачу на доказательство, решатель может выдать в качестве ответа либо логический символ "истина если он убедится, что условие действительно является логическим следствием посылок, либо логический символ "отказ если его попытки останутся безуспешными. Заметим, что решатель при рассмотрении задачи на доказательство не выдает сообщений о ложности условия либо о том, что оно не является следствием посылок (хотя для ускоренной выдачи отказа он и может попытаться установить эти обстоятельства). Задача на доказательство служит только для "односторонней"попытки установления истинности утверждения - именно в такого рода вспомогательных попытках обычно и возникает надобность при решении других задач. Если же нужно провести "двустороннюю"проверку - выяснить, является ли некоторое утверждение истинным либо ложным, то для этого используется задача на описание с пустым списком неизвестных (подробнее об этом см. ниже).

2. Задачи на описание. Здесь $n = 9$. В виде задачи на описание оформляются различные задачи, у которых требуется преобразовать заданным образом некоторый список утверждений - "условий" задачи. Преобразования этого списка условий осуществляются в предположении истинности списка посылок. Как правило, условия содержат неизвестные, и в задаче требуется получить явное описание (полное либо неполное - в зависимости от целевой установки задачи) допустимых значений таких неизвестных. Элемент p_5 задачи на описание представляет собой список $(g_1, \ldots, g_k)$ ее условий. Элемент p_6 - набор $(b_1 \ldots, b_k)$ весов условий; элемент p_7 - набор $(Q_1, \ldots, Q_k, Q_0)$ списков комментариев. Здесь Q_i $(i = 1, \ldots, k)$ представляет собой список комментариев к условию g_i; Q_0 - список комментариев ко всей задаче в целом. Элементы p_5, p_6, p_7 совершенно аналогичны элементам p_2, p_3, p_4, характеризующим посылки задачи. Элемент p_8 представляет собой набор технических пометок, называемых целями задачи.

Ответ, получаемый в процессе решения задачи на описание, постепенно складывается в списке ее условий, так что при успешном завершении преобразований в конце решения выдается группа соединенных логической связкой "и" заключительных условий. При неудаче в качестве ответа выдается логический символ "отказ". Класс утверждений, являющихся ответами на задачу, определяется в зависимости от набора ее целей. При этом некоторые из целей могут не накладывать каких-либо явных ограничений на ответ, а лишь указывать те или иные установки на оптимизацию, влияющие на принятие решений в процессе преобразований. Фактически, список целей задачи представляет собой установку на управление базой приемов решателя, осуществляющей рассмотрение задачи. За исключением особых случаев, принимается ряд соглашений о связи ответа задачи на описание с ее исходными условиями и посылками. Обычно задача на описание имеет цель, указывающую множество ее неизвестных, причем вхождение этих неизвестных в посылки задачи не допускается. Условия задачи должны являться следствиями ответа и посылок задачи; при этом ответ можно рассматривать как частичное либо полное описание множества наборов значений неизвестных, удовлетворяющих условиям задачи.

Достаточно часто при решении задачи на описание может оказаться полезным и даже необходимым рассмотрение совместных следствий условий и посылок. Так как ответ задачи извлекается из списка ее условий, то регистрация таких вспомогательных следствий в списке условий нежелательна - она приведет к чрезмерному его

увеличению и потребует сложной процедуры "расчистки" условий после определения значений неизвестных. Поэтому в структуре задачи на описание пришлось ввести специальный накопитель совместных следствий условий и посылок. Роль этого накопителя (элемент p_9) играет вспомогательная задача на исследование, называемая блоком анализа задачи на описание. Изначально блок анализа отсутствует и вводится решателем в процессе рассмотрения задачи. В исходной ситуации элемент p_9 может быть равен либо логическому символу "пустоеслово"(общий случай), либо, в особых случаях, логическому символу "0" - последнее означает запрет на ввод блока анализа при решении.

3. Задачи на преобразование. В этом случае $n = 8$. p_5 - выражение, называемое условием задачи. Это выражение нужно преобразовать к некоторому виду, определяемому целевой установкой задачи. Как правило, требуется, чтобы исходное и результирующее выражения были равны, в предположении истинности списка посылок. Иногда это требование может нарушаться; самый простой пример такого рода - использование задачи на преобразование для получения асимптотической оценки, когда результирующее выражение лишь асимптотически равно исходному. Более того, из-за технических причин в некоторых случаях удобно применять вспомогательные задачи на преобразование, условиями которых являются не выражения, а утверждения. p_6 - целое неотрицательное число, называемое весом условия; p_7 - набор комментариев к задаче. p_8 - набор целей задачи. Некоторые из целей накладывают ограничения на вид ответа; другие - определяют установки на оптимизацию ответа.

Ответом задачи на преобразование служит либо результирующее ее условие - если оно удовлетворяет определяемым целями ограничениям, либо логический символ "отказ".

4. Задачи на исследование. В этом случае $n = 5$. p_5 - набор целей задачи. Эти цели управляют процессом вывода представляющих интерес следствий из посылок задачи, упрощения посылок и удаления избыточных посылок. В результате таких действий исходная задача на исследование Z преобразуется в некоторую новую задачу Z', и при исчерпании средств, отведенных для решения задачи Z, в качестве ответа на нее выдается задача Z'. По существу, такая выдача ответа является фикцией - в действительности внешняя процедура, обратившаяся к решению задачи на исследование, обычно имеет непосредственную ссылку на нее, и по этой ссылке способна получать всю необходимую информацию о произошедших изменениях.

В исходной ситуации веса посылок и условий задачи, предъявляемой решателю, равны 0; блок анализа и комментарии к задаче (ее посылкам, условиям) отсутствуют. Вспомогательные задачи, вводимые решателем, могут иметь ненулевые исходные веса посылок и непустые списки комментариев. Это обеспечивает необходимую преемственность - сохраняет ранее накопленную информацию, которая может понадобиться при решении вспомогательной задачи. Изменение весов, формирование комментариев и блока анализа осуществляются системой в процессе решения задачи.

Целевые установки задач

Целевая установка задачи определяется ее списком целей. Она имеется у всех типов задач, за исключением задач на доказательство. С технической точки зрения, целевая установка обеспечивает управление поведением многих тысяч независимо функционирующих приемов, решающих задачу. Так как каждый прием рассматривает в своих решающих правилах лишь сравнительно небольшой фрагмент целевой

установки, и достижение ответа происходит в результате интегрального действия многих различных приемов, то варьирование списка целей задачи предоставляет достаточно большие степени свободы для управления коллективным поведением приемов. Пополнение списка возможных типов целей задач осуществляется, как правило, в связи с рассмотрением конкретной предметной области и не приводит к какой-либо модификации приемов, относящихся к другим предметным областям.

Перечислим некоторые наиболее часто встречающиеся типы целей.

Заметим, что цель задачи обычно представляется либо логическим символом, либо набором $(\varphi,\ A_1, \ldots, A_m)$, где φ - логический символ, называемый заголовком цели; $A_1, \ldots, A_m$ - некоторые объекты. В последнем случае указанный набор будем записывать далее как "$\varphi A_1 \ldots A_m$".

Пусть сначала Z - задача на описание. Если Z имеет цель "неизвестные $x_1 \ldots x_k$" , где $x_1, \ldots, x_k$ - попарно различные переменные, то $x_1, \ldots, x_k$ называются неизвестными задачи Z. Прочие свободные переменные посылок и условий называются параметрами задачи.

Задача на описание может и не иметь неизвестных, причем в зависимости от прочих целей в этом случае необходимо либо установить эквивалентность условий одной из логических констант "истина" , "ложь" , либо осуществить определенную переформулировку условий.

Истинность или ложность посылок задачи Z должна однозначно определяться при указании значений всех их свободных переменных, отличных от неизвестных. За исключением особых случаев, неизвестные задачи Z вообще не встречаются в посылках.

Если Z имеет цель "параметры $y_1 \ldots y_k$" , где $y_1, \ldots, y_k$ - некоторые из неизвестных задачи, то переменные $y_1, \ldots, y_k$ называются несущественными неизвестными задачи Z. Несущественные неизвестные задачи представляют собой те из неизвестных, для которых требуется установить лишь сам факт существования удовлетворяющих условиям их значений, не находя эти значения явным образом. Более точно, произвольный ответ на задачу Z представляет собой такое утверждение f, что для любых значений свободных переменных посылок задачи Z, при которых эти посылки истинны, из истинности f вытекает существование таких значений не входящих в f несущественных неизвестных задачи, что все условия задачи Z истинны. Несущественные неизвестные задачи являются "исключаемыми"неизвестными и обычно не входят в ответ.

Если задача имеет цель "прямойответ", то ответ f не может содержать вспомогательных переменных, вводимых для обозначения объектов, существование которых вытекает из истинности посылок задачи. В этом случае описание допустимых значений неизвестных осуществляется только в терминах объектов, упоминавшихся в посылках и условиях задачи.

Если задача Z имеет цель "пример", то ответ f представляет собой утверждение, построенное при помощи логических символов "и" , "или" из утверждений вида "равно($x\ t$)" , где x - неизвестная задачи Z; t - выражение, не зависящее от неизвестных, а также из некоторых утверждений, не зависящих от неизвестных задачи Z. При преобразовании f к виду дизъюнктивной нормальной формы ни в одну из элементарных конъюнкций не входят два различных утверждения вида "равно($x\ t$)" ,соответствующих одной и той же неизвестной x. Данная цель используется в тех

случаях, когда требуется указать конкретный набор значений неизвестных задачи, при которых истинны ее условия, быть может, с дополнительными ограничениями на объекты, упоминаемые в посылках задачи. Если такие ограничения недопустимы, то вводится дополнительная цель "полный". Более подробно, если задача Z имеет цели "полный" , "пример" , то при замене всех входящих в ответ f утверждений вида "равно(x t)" , где x - неизвестная задачи, на логический символ "истина", должно получаться такое утверждение f', являющееся следствием посылок задачи. Если же задача Z, имеющая цель "полный" , не имеет цели "пример" , то из истинности посылок этой задачи вытекает эквивалентность истинности ответа f существованию значений несущественных неизвестных, не являющихся свободными переменными f, при которых истинны все условия задачи Z.

Если задача Z имеет цель "явное" , то ее ответ f представляет собой утверждение, относящееся к классу так называемых явных описаний значений неизвестных этой задачи. Для определения класса явных описаний в каждой из рассматриваемых предметных областей выделяются семейства конечных множеств $\{f_1, \ldots, f_s\}$ утверждений, называемых атомарными описаниями переменной x; эта переменная является параметром каждого из утверждений $f_1, \ldots, f_s$. Так, атомарными описаниями переменной x считаются, например, множества $\{x = t_1\}$, $\{x \in t_1\}$, $\{t_1 < x, x < t_2\}$, $\{\exists k(k - \text{целое } \&\ x = \pi k)\}$, $\{x - \text{целое}\}$ и т.п. Здесь t_1, t_2 - произвольные выражения, не имеющие параметра x. Явными описаниями значений неизвестных задачи Z считаются утверждения, построенные при помощи логических символов "и", "или"из элементов атомарных описаний неизвестных этой задачи и не зависящих от неизвестных утверждений, причем такие, что в результате преобразования к виду дизъюнктивной нормальной формы каждая их элементарная конъюнкция, при подходящей группировке членов, приобретает вид "и($A_1(x_1)A_2(x_2)\ldots A_s(x_s)B$)", где для любого $i = 1, \ldots, s$ $A_i(x_i)$ - конъюнкция утверждений атомарного описания неизвестной x_i задачи Z; x_i не является параметром утверждений $A_{i+1}(x_{i+1}), \ldots, A_s(x_s)$; B - утверждение, не зависящее от неизвестных.

В некоторых случаях бывает полезной цель "и" , при наличии которой ответ f на задачу Z имеет вид "и(равно($x_{j_1}t_1$) ... равно(x_{j_m} t_m) $g_1 \ldots$ g_s)" , где $m + s - 1 \geq 1$; $x_{j_1}, \ldots, x_{j_m}$ - различные неизвестные задачи Z; $t_1, \ldots, t_m, g_1, \ldots, g_s$ не зависят от неизвестных задачи. Указанный вид ответа определяет подстановку выражений $t_1, \ldots, t_m$ вместо переменных $x_{j_1}, \ldots, x_{j_m}$; такая подстановка может использоваться, например, при решении системы уравнений путем выражения части ее неизвестных $x_{j_1}, \ldots, x_{j_m}$ через остальные неизвестные.

Для ограничения зависимости ответа от использованных при формулировке задачи переменных применяется цель "известно $y_1 \ldots y_k$" , где $y_1, \ldots, y_k$ - переменные, не являющиеся неизвестными задачи; $k \geq 0$. При наличии такой цели каждая свободная переменная ответа f представляет собой либо неизвестную задачи Z, либо некоторую переменную $y_i; i = 1, \ldots, k$. Переменные $y_1, \ldots, y_k$, выделенные целью указанного вида, называются исходными данными задачи.

В качестве примера цели, определяющей установку на оптимизацию, приведем цель "упростить" , активизирующую попытки редактирования найденного ответа путем разрешения условий на параметры относительно этих параметров, упрощения входящих в ответ выражений и упрощения логической структуры ответа.

При формулировке исходной задачи может быть использована цель "одз" , указывающая на необходимость решения задачи в области допустимых значений неизвест-

ных и параметров. Эта цель, по сути дела, является лишь относящимся к интерфейсу системы техническим приемом, позволяющим в неявной форме задавать часть условий либо посылок задачи. Она инициирует расширение списков условий и посылок задачи рядом утверждений, характеризующих область допустимых значений переменных задачи, после чего удаляется.

Приведем несколько примеров часто встречающихся комбинаций целей задач на описание. Список { "неизвестные $x_1 \dots x_k$" , "полный" , "пример" } встречается у задач Z, введенных при решении задачи на доказательство существования значений переменных $x_1, \dots, x_k$, удовлетворяющих некоторому утверждению f. При этом в задаче Z требуется найти конкретные значения $x_1, \dots, x_k$, быть может, выразив их через вспомогательные обозначения для объектов, существование которых вытекает из посылок. Последнее объясняет отсутствие цели "прямойответ", блокирующей приемы, вводящие такие вспомогательные обозначения.

Список { "неизвестные $x_1 \dots x_k$" , "полный" , "явное" , "прямойответ" }, как правило, вместе с целью "упростить", встречается у задач, в которых требуется описать в явной форме все значения неизвестных, удовлетворяющие условиям (например, задачи на решение систем уравнений или неравенств; задачи на определение геометрического места точек в геометрии, и т.п.). Если некоторые из условий такой задачи представляли собой утверждения о существовании некоторых объектов $y_1, \dots, y_m$, то при ее решении возможно появление вспомогательных задач, неизвестными которых, помимо $x_1, \dots, x_k$, становятся также $y_1, \dots, y_m$. В этом случае вспомогательная задача имеет, наряду с перечисленными выше, цель "параметры $y_1 \dots y_m$".

Список "неизвестные $x_1 \dots x_k$" ,"полный" , "явное" , "прямойответ" , "известно $y_1 \dots y_k$"возникает в тех случаях, когда значения неизвестных требуется выразить через исходные данные $y_1, \dots, y_k$, представляющие собой, вообще говоря, лишь часть параметров задачи. Как правило, в этом случае список условий задачи имеет вид "равно(x_1 t_1)" , ... , "равно(x_k t_k)" , где $t_1, \dots, t_k$ - выражения, не зависящие от неизвестных, но зависящие от некоторых параметров задачи, не являющихся исходными данными $y_1, \dots, y_k$. Примером такого рода являются геометрические задачи на вычисление, посылки которых описывают некоторый геометрический чертеж; выделяются известные величины $y_1, \dots, y_k$ (переменные), характеризующие этот чертеж, и требуется определить ряд связанных с ним неизвестных величин $t_1, \dots, t_k$. Встречающиеся в описании чертежа переменные, обозначающие точки, хотя формально и являются известными, не должны входить в ответ задачи.

Если требуется проверить истинность утверждений при заданных посылках, то используется задача на описание, единственным условием которой служит данное утверждение, а цели суть "полный", "явное", "прямойответ"(обычно добавляется цель "одз"). Неизвестные задачи отсутствуют, и при решении ее будут предприниматься попытки заменить проверяемое утверждение на логическую константу "истина"либо "ложь". Эта константа и будет выдана в качестве ответа.

Цели задач на преобразование оказываются тесно связаны с конкретными предметными областями (например, разложение на множители либо преобразование к виду суммы одночленов в элементарной алгебре; приведение входящих в преобразуемое выражение тригонометрических функций к общему аргументу в тригонометрии; нахождение асимптотики выражения при отыскании пределов и т.п.). В одной задаче, как правило, сочетаются несколько таких целей (например, цель "разложитьнамножители"и цель "неизвестные $x_1 \dots x_k$", указывающая, что разложение на множители

выражения t необходимо для решения уравнения $t = 0$ относительно $x_1, \ldots, x_k$).

Единственной целью сравнительно общего характера для задач на преобразование является цель "упростить", выражающая тенденции к определенной стандартизации выражения. Впрочем, понятие "упростить"трактуется приемами решателя эвристически - в контексте прочих обстоятельств, сопровождающих задачу, так что в различных случаях упрощение одного и того же выражения может дать различные результаты. С другой стороны, цель "упростить"заменяет собой большое число целей частного характера: задачи на нахождение численного значения выражения, на вычисление производной, предела, интеграла и т.п. могут формулироваться как задачи на преобразование, имеющие единственную цель "упростить".

Как и в случае задачи на описание, допускается формулировка исходной задачи на преобразование с целью "одз", определяющей присоединение к списку посылок совокупности утверждений, описывающей область допустимых значений параметров.

Целевая установка задачи на исследование, возникающей в большинстве случаев как блок анализа задачи на описание, определяется спецификой этой внешней задачи. В такой целевой установке присутствует цель "неизвестные $x_1 \ldots x_k$, перенесенная из задачи на описание и направляющая вывод следствий таким образом, чтобы увеличить "степень разрешенности"утверждений относительно переменных $x_1, \ldots, x_k$. В случае задач на исследование, возникших при доказательстве утверждений "от противного", используется цель "противоречие", активизирующая попытки установить противоречивость посылок.

Примеры формулировки задач для решателя

Чтобы сделать более понятной приведенную выше общую схему формализации задач для решателя, приведем примеры постановки задач из различных разделов. Хотя в данном параграфе будет часто использоваться "скобочная"форма записи, соответствующая внутреннему логическому языку решателя, в действительности при постановке задачи и просмотре процесса ее решения применяется приближенная к стандартной математическая запись. Минимальные необходимые для ввода новых задач сведения об интерфейсе, позволяющем работать с решателем и использующем стандартную запись, будут приведены в следующем параграфе этой главы.

1. Алгебра множеств.

   Начнем с простейшего раздела - алгебры множеств. Если требуется доказать, например, истинность тождества $c \cup (b \setminus a) = (b \cup c) \setminus a$, предполагая истинными утверждения $a \subseteq b$ и $(b \cap c) = \emptyset$, то мы приходим к задаче на доказательство, имеющей посылки "множество(a)" , "множество(b)" , "множество(c)" ,"содержится(a b)", "равно(пересечение(b c)пусто)" и единственное условие "равно(объединение(c разность(b a)) разность(объединение(b c)a))". Заметим, что первые три посылки, указывающие тип значений переменных (множество), обычно формируются автоматически, и вручную их вводить не нужно. В последующих примерах будем для краткости опускать такие посылки.

   Следующий пример - задача на упрощение выражения $(c \cap b) \cup (c \cap a) \setminus d$ в предположении, что выполняется $a \subseteq b$. Она оформляется как задача на преобразование, имеющая посылку "содержится(a b)"и условие "объединение(пересечение(c b) разность(пересечение(c a)d))". Эта задача имеет цели "упростить"и "одз". Ответом на нее служит выражение "пересечение(b c)".

Простейшим примером задачи на описание может служить задача на решение уравнений $a \cap x = b, a \cup x = c$ в множествах, если для известных множеств a, b, c выполнены соотношения $b \subseteq a, a \subseteq c$. Посылками такой задачи служат утверждения "содержится(b a)", "содержится(a c)", дополненные утверждениями о том, что a, b, c - множества. Условиями являются утверждения "равно(пересечение(a x)b)" , "равно (объединение(a x)c)"и "множество(x)". Как и в случае посылок, обычно добавление условий, указывающих тип значения неизвестных (например, "множество(x)"), выполняется автоматически; в последующих примерах такие условия опускаем. Задача имеет цели "полный" , "явное" , "прямойответ" , "одз" , "неизвестные x"и "упростить". Ответом задачи служит утверждение "равно(x объединение(b разность(c a)))".

Другая ситуация в задачах на описание - когда требуется найти не полное описание всех допустимых значений неизвестных, а привести хотя бы один пример таких значений. Так, если нужно найти пример множества x, удовлетворяющего соотношению $a \cup x = b$ в предположении, что для известных множеств a, b выполняется $a \subseteq b$, то рассматривается задача на описание, имеющая посылку "содержится(a b)"и условие "равно(объединение(a x)b)". Эта задача имеет цели "полный", "пример", "неизвестные x" , "прямойответ"и "одз". Ответом на нее может служить, например, утверждение "равно(x b)".

2. Элементарная алгебра.

Примеры формулировки задач по элементарной алгебре начнем с вычисления значений константных выражений. В простейших случаях здесь применяется обычная задача на преобразование с целью "упростить". Она имеет единственную вырожденную посылку - логическую константу "истина". Условием служит упрощаемое выражение. Представление о том, что является упрощением - по существу, эвристическое. Оно возникло в процессе рассмотрения многих конкретных примеров на упрощение и соответствующих этим примерам коррекций действий решателя. Так, при упрощении выражения $\frac{(5+\sqrt{3})}{4-\sqrt{3}}$, решатель избавляется от иррациональности в знаменателе и выдает ответ $\frac{9\sqrt{3}+23}{13}$; при упрощении выражения $\arctan(\frac{1}{3}) + \arctan(\frac{1}{5}) + \arctan(\frac{1}{7}) + \arctan(\frac{1}{8})$ выдается ответ $\frac{\pi}{4}$, и т.п. В зависимости от задачи, исходное выражение может остаться неизмененным либо (что дает обучающий материал для продолжения оптимизации приемов) быть преобразованным к худшему виду. Впрочем, в стандартных задачах с "хорошим"ответом такое явление уже сейчас наблюдается достаточно редко.

Если требуется найти точное целочисленное значение выражения, в котором встречаются степени с большим показателем, факториалы и т.п., то рекомендуется цель "упростить"сопровождать целью "число"(в интерфейсе решателя для ввода такой целевой комбинации предусмотрен специальный пункт меню). В этом случае снимаются блокировки на действия, приводящие к длинным десятичным записям, имеющие место в случае обычной задачи на упрощение. Интерпретатор языка ЛОС позволяет выполнять точные арифметические действия с десятичными записями чисел, имеющими до 3000 знаков.

Если требуется получить приближенное значение константного выражения, содержащего арифметические операции, степени, логарифмы, прямые и обратные тригонометрические функции, причем в этом приближенном значении

нужно иметь заданное число точных знаков после запятой, то применяется задача на преобразование, имеющая цели "упростить"и "числоценка N", где N - число точных знаков после запятой. Производя вычисления, решатель получает гарантированные верхнюю и нижнюю оценки рассматриваемого выражения, увеличивая число цифр до тех пор, пока после округления "к ближайшему"в них не совпадут требуемые N цифр после запятой.

Наконец, для получения приближенной оценки константного выражения возможно использование встроенного в компьютер математического сопроцессора (14 значащих цифр). Здесь используется комбинация целей "числзначение"и "выч".

В случае задач на упрощение неконстантных алгебраических выражений, как и для константных, используется сочетание целей "упростить"и "одз". Например, для упрощения в области допустимых значений выражения $\frac{a}{a^2+b^2} - \frac{b(a-b)^2}{a^4-b^4}$ создается задача на преобразование, имеющая своим условием это выражение, посылками - утверждения "число(a)"и "число(b)"и указанные две цели. На нее выдается ответ $\frac{1}{a+b}$. Необходимые условия на о.д.з. (отличие знаменателей от 0) вводятся решателем в список посылок самостоятельно и в процессе решения изменяются так, чтобы не нарушалось соответствие между ними и "обслуживаемыми"ими подвыражениями условия задачи. Собственно в ответ результирующие условия на о.д.з. для параметров не включаются, но они легко могут быть считаны при пошаговом просмотре решения.

Задачи разложения на множители имеют целевую установку "упростить", "разложитьнамножители", "одз". Так, для разложения на множители выражения $a^2 - b^2 - c^2 - 2bc$ создается задача на преобразование с этими целями, условием которой служит данное выражение, а посылки указывают тип значения переменных a, b, c. Заметим, что задачи на преобразование тригонометрических выражений к виду, удобному для логарифмирования, формулируются с теми же целями, что и обычные "алгебраические"задачи разложения на множители. Если требуется разложить на множители выражение, разрешая использование комплексных чисел, то кроме указанных выше трех целей добавляется четвертая - "видУмножение".

Чтобы разложить алгебраическую дробь в сумму простейших дробей, используется задача на преобразование, имеющая цели "простейшиедроби", "упростить", "одз". Например, решая задачу на преобразование с указанными целями и условием $\frac{1}{(a^2-a+1)(a^2+2a+3)}$, решатель выдает ответ $\frac{3a+4}{19(a^2+2a+3)} + \frac{5-3a}{19(a^2-a+1)}$.

Наконец, для упрощения алгебраических выражений с "раскрыванием скобок" предусмотрена целевая установка "упростить", "раскрытьскобки", "одз". Так, решая задачу на преобразование с данными целями и условием $a(b-c)+c(a-d)+d(c-b)$, решатель выдает ответ $ab-bd$. Хотя в этом ответе и можно было бы вынести за скобку общий множитель b, но цель "раскрытьскобки"блокирует такое действие.

Если требуется упростить некоторое выражение, вместо параметров которого должны быть подставлены другие выражения, то такую подстановку не обязательно делать непосредственно - достаточно указать в посылках задачи

равенства, определяющие подставляемые выражения. Например, если имеется задача на преобразование с условием $-4a^2x^{\frac{1}{m}+\frac{1}{n}} + (x^{\frac{1}{m}} + x^{\frac{1}{n}})^2$ и посылкой $x = (\sqrt{a^2-1}+a)^{\frac{2mn}{m-n}}$ (не считая автоматически добавляемых посылок, указывающих тип значений переменных), то фактически будет упрощаться результат подстановки в условие выражения для x, определяемого посылкой.

При решении задачи на упрощение алгебраического выражения может возникнуть разбор подслучаев, в результате чего ответ будет иметь вид "условного"выражения. Так, при упрощении выражения $\sqrt{a+2\sqrt{2a-4}}+\sqrt{a-2\sqrt{2a-4}}$ решатель выдает ответ "$2\sqrt{2}$ при $a < 4$, иначе $2\sqrt{a-2}$"(для большей наглядности вместо скобочной конструкции "вариант(...)"мы использовали здесь стандартную запись, прорисовываемую формульным редактором решателя).

Задачи на суммирование также оформляются как стандартные задачи на упрощение. Например, задача на преобразование с целями "упростить", "одз", условием $\sum_{m=1}^{n}(2m-1)^3$ и посылкой "натуральное(n)"приводится к ответу $(2n^2-1)n^2$. Заметим, что в этих задачах важно указывать в посылках, что значениями параметров, определяющих границы суммирования, служат целые числа, и сопровождать такие параметры всеми необходимыми неравенствами.

Задачи на решение уравнений, неравенств, систем уравнений и неравенств практически всегда имеют целевую установку "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные $x_1 \dots x_n$". Например, чтобы решить уравнение $\frac{\sqrt{1+a^2x^2}-ax}{\sqrt{1+a^2x^2}+ax} = \frac{1}{b^2}$ относительно неизвестной x, создается задача на описание с указанной выше целевой установкой, условиями которой служат данное уравнение и утверждение "число(x)"(последнее создается автоматически процедурами интерфейса решателя). Эта задача имеет посылку "число(a)". Решая задачи с параметрами, решатель аккуратно анализирует все возможные случаи, объединяя полученные для них результаты в общем ответе. Так, в указанном уравнении ответ будет объединять два подслучая: $a \neq 0, b \neq 0, x = \frac{b^2-1}{2a|b|}$ и $(b = -1 \vee b = 1), a = 0$ с "вынесенной за скобку"общей частью "число(x)".

Если в задаче требуется найти все значения параметров, при которых система уравнений либо неравенств имеет хотя бы одно решение, то условие записывается с помощью квантора существования. Например, для нахождения всех значений параметра a, при которых имеет решение система уравнений $y(ax-1) = 2|x+1| + 2xy, xy + 1 = x - y$, вводится задача на описание с условием $\exists xy(y(ax-1) = 2|x+1| + 2xy \& xy + 1 = x - y)$. Целевая установка у нее - такая же, как и выше (но роль неизвестной играет a). Аналогично, если нужно найти все значения параметров, при которых система уравнений либо неравенств имеет заданное число решений, то используется описатель "класс", с помощью которого обозначается множество решений рассматриваемой системы. Так, для отыскания значений параметра a, при которых система $\log_2 y + 2 = \log_2(x+3y),\ y = 2(x-a)^2 + x + 2a - 4$ имеет ровно два решения, создается задача на описание с условием "мощность(класс(xy $(\log_2 y + 2 = \log_2(x+3y)$ & $y = 2(x-a)^2 + x + 2a - 4)$))"$= 2$. Если требуется найти значения параметров, при которых две системы (два уравнения, и т.п.) имеют одинаковые множества решений, то условие записывается в виде эквивалентности под квантором общности. Например, чтобы найти значения

параметра a, при которых множество решений уравнения $4(\cos x)^2 = a^2 - 6$ совпадает со множеством решений уравнения $1 - \cos(2x) = \frac{a}{6}$, используется задача на описание с условием $\forall x(4(\cos x)^2 = a^2 - 6 \Leftrightarrow 1 - \cos(2x) = \frac{a}{6})$. Заметим, что в перечисленных примерах под кванторами и описателем "класс"автоматически вводятся дополнительные утверждения "число(...)", уточняющие тип значений связанных переменных.

Если требуется определить число решений уравнения (системы) в зависимости от значений параметров, то возникает уже не задача на описание, а задача на преобразование. Условием такой задачи служит выражение, задающее мощность множества корней, а цели - стандартные: "упростить"и "одз". Решатель пытается получить явное представление для множества корней, решая рассматриваемое уравнение (систему) либо иными средствами определяет мощность этого множества (например, анализируя с помощью производных и пределов интервалы монотонности). Так, если нужно определить число решений уравнения $\sqrt{\sqrt{x+\frac{1}{4}} + x + \frac{1}{2}} = a$, то условием задачи на преобразование будет выражение "мощность(класс(x $\sqrt{\sqrt{x+\frac{1}{4}} + x + \frac{1}{2}} = a$))". Ответ, получаемый здесь решателем, имеет вид "1, если $\frac{1}{4} \leq a$, иначе 0".

При доказательстве тождества либо неравенства создается задача на доказательство, условием которой является это тождество либо неравенство, а посылки перечисляют ограничения на параметры. Так как задача на доказательство не имеет списка целей, то указание на пополнение списка посылок ограничениями на область допустимых значений передается ей не в виде цели "одз", а в виде комментария "одз". Это делается автоматически интерфейсом решателя. В качестве примера приведем задачу на доказательство с условием $64ab(a+b)^2 \leq (\sqrt{a} + \sqrt{b})^8$ и посылками "число(a)", "число(b)". Неотрицательность параметров a, b регистрируется в списке посылок уже при решении задачи, когда обрабатывается комментарий "одз".

3. Комбинаторика.

Задача по комбинаторике обычно формализуется как задача на преобразование выражения "мощность(A)". Целевая установка - стандартная, т.е. "Упростить в о.д.з.". Извлечение выражения "мощность(A)" из текста задачи выполняется вручную. Например, рассмотрим следующий текст: "Экскурсанты разделились на две равные группы для розыска заблудившегося товарища. Среди них есть только 4 человека, знакомые с местностью. Каким числом способов они могут разделиться так, чтобы в каждую группу вошло 2 человека, знающих местность, если всего их 16 человек?". Множество всех экскурсантов обозначаем через A; подмножество экскурсантов, знакомых с местностью - D. Тогда вводим следующие три посылки задачи: $\mathrm{card}(A) = 16$, $\mathrm{card}(D) = 4$, $D \subseteq A$. Если подсчитывается количество упорядоченных пар групп, то преобразуемое условие имеет вид $\mathrm{card}(\mathrm{set}_{BC}(B - \mathrm{set}\ \&\ C - \mathrm{set}\ \&\ A = B \cup C\ \&\ B \cap C = \emptyset\ \&\ \mathrm{card}(B) = \mathrm{card}(C)\ \&\ \mathrm{card}(B \cap D) = 2\ \&\ \mathrm{card}(C \cap D) = 2))$. Если же подсчитывать количество неупорядоченных пар групп (что, собственно, и требуется в задаче), то условие несколько усложняется: $\mathrm{card}(\mathrm{set}_x(\exists_{BC}(x = \{B, C\}\ \&\ B -$

set & C – set & $A = B \cup C$ & $B \cap C = \emptyset$ & card(B) = card(C) & card$(B \cap D) =$ 2 & card$(C \cap D) = 2)))$. Обе версии без труда доводятся решателем до ответа.

Приведем еще один пример, иллюстрирующий часто встречающиеся в задачах по комбинаторике конструкции с отображениями: "Из цифр 1,2,3,4,5,6,7,8,9 составляются всевозможные пятизначные числа, не содержащие одинаковых цифр. Определить количество чисел, в которых есть цифры 2,4 и 5 одновременно.". Пятизнач ное число можно рассматривать как отображение множества номеров цифр $\{1, \ldots, 5\}$ в множество цифр $\{1, \ldots, 9\}$. Условие наличия в числе x цифры m тогда запишется как $\neg$(слой$(x, m) = \emptyset)$; условие различия цифр числа - "взаимнооднозначно(x)". Таким образом, условие задачи на преобразование имеет вид card(set$_f$(Отображение$(f, \{1,$
$\ldots, 5\}, \{1, \ldots, 9\})$ & взаимнооднозначно(f) & $\neg$(слой$(f, 2) = \emptyset)$ & $\neg$(слой$(f, 4) = \emptyset)$ & $\neg$(слой$(f, 5) = \emptyset)))$.

4. Элементарная геометрия.

Вычислительные задачи по планиметрии внешне напоминают задачи на решение уравнений и неравенств - в обоих случаях требуется определить значения "неизвестных"величин. Однако, здесь имеются существенные различия, приводящие к необходимости особой логической формализации. Прежде всего, параметры посылок геометрической задачи бывают двух типов - числовые, про которые обычно предполагается, что они "известны", и параметры - обозначения точек чертежа. Последние, хотя и содержатся в посылках, "известными"никоим образом не считаются и в ответ входить не должны. Более того, те величины, которые требуется определить в геометрической задаче, обычно изначально явно выражены через ее "точечные"параметры, так что с точки зрения обычной "алгебраического типа"задачи на описание, ситуация вырожденная. Эти обстоятельства потребовали ввести для геометрических задач на вычисление специальную цель "известно $a_1 \ldots a_n$", которая указывает все параметры $a_1, \ldots, a_n$, которые могут встречаться в ответе (возможен случай $n = 0$, и тогда цель сводится к логическому символу "известно"). Изначально имеющаяся информация о чертеже перечисляется в списке посылок задачи; условиями служат равенства, явно выражающие неизвестные величины через обозначения точек чертежа.

В качестве простейшего примера рассмотрим следующую задачу. В треугольнике ABC длина стороны AC равна a; угол BAC равен b, а угол BCA равен c. Требуется вычислить площадь данного треугольника. Эта ситуация формализуется в виде задачи на описание, посылками которой служат следующие утверждения: "треугольник (ABC)"; "расстояние(AC)= a"; "угол(BAC)= b"; "угол(BCA)= c". Они дополняются вводимыми автоматически утверждениями о типе значения: "точка(A)", "точка(B)", "точка(C)", "число(a)", "число(b)", "число(c)". Задача имеет условия "x =площадь (фигура(ABC))"и "число(x)". Целевая установка ее состоит из целей "неизвестные x"; "известно abc"; "полный"; "прямойответ"; "явное"; "упростить"; "одз".

При формулировке утверждений, описывающих чертеж геометрической задачи, следует учитывать некоторую ограниченность созданного на текущий момент запаса понятий. Так, например, отсутствует в явном виде понятие "медиана треугольника". Если в условии задачи говорится про медиану, то нужно

вводить в рассмотрение новую точку - основание медианы, и указывать в посылках, что расстояния ее до концов стороны треугольника равны, а сама эта точка лежит на данной стороне (т.е. на отрезке с соответствующими концами). Впрочем, последнее условие можно ослабить - задать принадлежность точки не стороне, а прямой, на которой лежит эта сторона. Приведем несколько примеров простых задач, в которых встречаются такого рода "стандартные"для решателя формулировки геометрических условий.

Пусть имеется треугольник ABC, у которого длина стороны AB равна 6, длина стороны BC равна 8; медианы, проведенные из вершин A и C, взаимно перпендикулярны, причем требуется найти площадь треугольника. Посылки соответствующей задачи на описание таковы: "треугольник(ABC)"; "расстояние(AB)= 6"; "расстояние(BC)= 8"; "$D \in$ отрезок(BC)"; "расстояние(BD) = расстояние(CD)"; "$E \in$ отрезок(AB)"; "расстояние(AE) = расстояние(BE)"; "перпендикулярно(прямая(CE) прямая(AD))". Условием служит равенство "$x =$ площадь(фигура(ABC))". Задача имеет цели "неизвестные x", "известно", "полный", "прямойответ", "явное", "упростить", "одз".

Аналогичные примеры приведем для биссектрисы и высоты треугольника. В первом из них даны длины a, b, c сторон BC, AC, AB треугольника ABC. Биссектрисы треугольника, проведенные из вершин B и C, пересекаются в точке D. Требуется найти, в каком отношении точка D делит биссектрису, проведенную из вершины B. Посылками задачи для этого примера служат утверждения "треугольник(ABC)"; "расстояние(BC)= a"; "расстояние(AC)= b"; "расстояние(AB)= c"; "биссектриса ($ABCR$)"(введена точка R - основание биссектрисы угла B); "$R \in$отрезок(AC)"(точка R лежит на стороне AC); "биссектриса($ACBP$)"; "$P \in$отрезок(AB)"; (аналогично, для биссектрисы угла C введено основание P); "$D \in$отрезок(BR)"; "$D \in$отрезок(CP)" (введена в рассмотрение точка D пересечения биссектрис). Условие задачи имеет вид "$x =$расстояние(BD) / расстояние(DR)". Цели задачи суть: "неизвестныеx"; "известно abc"; "полный"; "явное"; "прямойответ"; "одз"; "упростить".

В другом примере известно, что длина стороны BC треугольника ABC равна 8; длины высот треугольника, проведенных из вершин A и B, равны, соответственно, 4 и 6.4. Требуется найти длины сторон AB и AC. Посылки задачи суть: "треугольник (ABC)"; "расстояние(BC)=8"; "$D \in$ прямая(BC)"(введено основание D высоты, проведенной из A); "перпендикулярно(прямая(AD) прямая(BC))"(высота перпендикулярна основанию); "$E \in$ прямая(AC)"(основание второй высоты); "перпендикулярно(прямая(BE)прямая(AC))"; "расстояние(AD)= 4"; "расстояние(BE)= 6.4".
Условия этой задачи имеют вид: "$x =$ расстояние(AB)"; "$y =$расстояние(AC)".

Заметим, что в планиметрических задачах обычно имеется фиктивная посылка "планиметрия". Как правило, она вводится автоматически после набора задачи, однако иногда ее приходится вводить вручную (последовательным нажатием клавиш "п","и"). Эта посылка необходима для ускорения проверок принадлежности рассматриваемых прямых и точек общей плоскости. В особых случаях ее отсутствие может также привести к выдаче отказа на задачу.

Для указания на то, что точка принадлежит внутренности фигуры, ограниченной системой лучей и отрезков, используются утверждения "однасторона(...)";

"разныестороны(...)". Так, рассмотрим следующую задачу. Внутри угла BAC, величина которого равна a, взята точка M. Из нее опущены перпендикуляры на стороны угла, причем известно, что основания перпендикуляров отстоят от вершины угла на расстояния p и q. Нужно найти длины перпендикуляров. Данная ситуация описывается следующей системой посылок: "угол(BAC)= a"; "однасторона(C, M,прямая($A\ \ B$))"; "однасторона(B, M,прямая(AC))"(условие принадлежности точки M углу сформулировано как пара условий: эта точка лежит по ту же сторону от одной стороны угла, что и направляющая точка другой стороны); "перпендикулярно(прямая(PM)
прямая(AB))"; "$P \in$ прямая(AB)"(P - основание первого перпендикуляра); "перпендикулярно(прямая(QM)прямая(AC))"; "$Q \in$прямая(AC)"(Q - основание второго перпендикуляра); "расстояние(AP)= p"; "расстояние(AQ)= q"; $0 < a$; $a < \pi$ (невырожденность угла); $0 < p$; $0 < q$ (эти два неравенства уточняют, что точка M находится во внутренности угла). Условия задачи суть: "x =расстояние(PM)"; "y =расстояние (QM)". Заметим, что условие принадлежности точки углу могло быть сформулировано и непосредственным образом, как "$M \in$Угол(BAC)". Приведенная выше более громоздкая формулировка нужна здесь лишь как иллюстрация возможного применения предикатов "однасторона"и "разныестороны".

Геометрические задачи на доказательство формулируются почти так же, как и задачи на вычисление: в посылках задается чертеж; условием служит утверждение, которое требуется доказать. Например, задача на доказательство того, что каждый выпуклый четырехугольник, диагонали которого суть биссектрисы его внутренних углов, является ромбом, имеет следующие посылки: "четырехугольник($ABCD$)"; "биссектриса($BADC$)"; "биссектриса($BCDA$)"; "биссектриса($ABCD$)"; "биссектриса($ADCB$)". Условие этой задачи - "ромб(
$ABCD$)".

Геометрические задачи на построение и на определение геометрического места точек хорошо формализуются в виде обычных задач на описание, имеющих стандартную целевую установку "неизвестные $x_1 \dots x_n$", "полный", "явное", "прямойответ", "упростить", "одз". В качестве примера рассмотрим задачу нахождения геометрического места всех таких точек C на данной прямой AB, сумма расстояний которых до точек A, B - наименьшая. Предполагается известным, что расстояние от A до B равно 50. Она формализуется как задача на описание, имеющая посылку "расстояние
(AB)= 50", и условие "Минимум(отображение(C точка(C) расстояние(AC)+ расстояние(BC)) прямая(AB) x y)". Неизвестные этой задачи суть x, y; значение первой из них равно наименьшей сумме рассматриваемых расстояний; значение второй - множеству точек, в которых достигается эта наименьшая сумма. Решатель выдает на задачу ответ "$y =$ отрезок(AB); $x = 50$".

Другой пример - задача на построение треугольника ABC по известным длинам a, b, c его сторон AB, AC, BC. Она формализуется в виде задачи на описание с условиями "треугольник(ABC)"; "расстояние(AB)= a"; "расстояние(AC)= b"; "расстояние
(BC)= c". Неизвестными этой задачи служат A, B, C. Решатель выдает на данную задачу ответ "$0 < a+b-c, 0 < a+c-b, 0 < b+c-a$, точка($C$), $A \in$окр(C b), $B \in$ окр(A a)$\cap$ окр(C c)". Напомним, что "окр(A b)"обозначает окружность с

центром в точке A, имеющую радиус b. Разумеется, в задачах на построение вместо циркуля и линейки используется ряд вспомогательных элементарных операций и отношений, позволяющих последовательно определять новые точки по ранее найденным.

5. Математический анализ.

Вычисление производных и пределов происходит при помощи задач на преобразование, имеющих цели "упростить"и "одз". Стандартный вид условия задачи на вычисление производной - "производная(отображение(y число(y) $f(y)$)x)". Здесь $f(y)$ - выражение, определяющее значение дифференцируемой функции в точке y. Заметим, что переменная y - связанная и используется только внутри конструкции "отображение(...)", задающей исходную функцию. Условие задачи здесь имеет своим значением не функцию - производную исходной функции, а лишь значение этой производной в точке x. В качестве x может фигурировать либо переменная, и тогда на экране условие будет прорисовано в виде $\frac{df(x)}{dx}$, либо некоторое другое выражение (например, константа), и тогда на экране условие будет прорисовано в виде $\frac{df(y)}{d(y=x)}$. Важным является то обстоятельство, что решатель не только выполняет формальное дифференцирование, но и осуществляет последующие упрощения полученного выражения в его области допустимых значений. Это занимает, как правило, значительно больше времени, чем само нахождение производной, но зато позволяет получать результаты удовлетворительного качества - по крайней мере для примеров из стандартных задачников.

При вычислении частных производных и производных высших порядков используется другой вид условия задачи на преобразование. Например, для функции двух переменных оно таково: "частнпроизв(отображение(u v и(число(u) число(v)) $f(u,v)$)набор(k m)набор(xy))". Здесь k, m - кратности дифференцирования по первой и второй переменным (одна из них может равняться 0); $f(u,v)$ - выражение, определяющее значение дифференцируемого выражения. На экране указанное условие прорисовывается как $\frac{d^{k+m}f(x,y)}{dx^k dy^m}$ Как и в случае одной переменной, если, например, x не является переменной, то запись преобразуется к виду $\frac{d^{k+m}f(u,y)}{d(u=x)^k dy^m)}$.

Если нужно продифференцировать функцию, заданную параметрически, то приходится выбирать одну из двух возможностей логической формализации. Либо нужно вводить специальное обозначение для параметрического "определения"новой функции $y(x)$ по двум функциям $x = f(t)$ и $y = g(t)$, либо работать только с одной функцией $y(x)$, для которой задано соотношение $y(f(t)) = g(t)$. В решателе выбрана вторая возможность, как технически более простая. При такой записи, для нахождения производной функции, заданной параметрическими соотношениями $y(t) = b\ sh(t),\ \ x(t) = a\ ch(t)$, вводится задача на преобразование, имеющая посылку "$\forall t$(одз(t) $\Rightarrow$ $y(a\ ch(t)) = b\ sh(t)$)". Здесь запись "одз(t)"имеет технический характер; она заменяется решателем на группу условий, определяющих принадлежность t области допустимых значений для выражений, расположенных справа от импликации. Условие задачи записывается как $\frac{dy(x)}{d(x=a\ ch(t))}$. Решатель выдает ответ $\frac{b\ cth(t)}{a}$.

В случае нахождения производной неявной функции применяется аналогичная запись. Так, чтобы продифференцировать функцию $y(x)$, заданную неявным

соотношением $\sqrt{y(x)}+\sqrt{x}=\sqrt{a}$, вводится задача на преобразование, имеющая посылку $\forall x\ (\text{одз}(x) \Rightarrow \sqrt{y(x)}+\sqrt{x}=\sqrt{a}))$. Условие имеет вид $\frac{dy(x)}{dx}$. Ответ выдается в виде $-\sqrt{\frac{y(x)}{x}}$.

При вычислении пределов условие задачи на преобразование имеет вид "предел (отображение(x число(x) $f(x)$)m a)", где a - точка, в которой определяется предел; m - указатель типа окрестности этой точки (0 - двусторонняя окрестность; 1 - левая; 2 - правая). Это условие прорисовывается, соответственно, как $\lim_{x\to a} f(x)$ либо $\lim_{x\to a-0} f(x)$ либо $\lim_{x\to a+0} f(x)$. Заметим, что в случае выражения с параметрами решатель может выдать ответ с перечислением подслучаев. Так, при решении задачи с посылками $0<a$, $0<b$ и условием $\lim_{x\to\infty}\left(\frac{ax+c}{bx+d}\right)^x$, получается ответ, состоящий из трех подслучаев: 1) если $0<-\frac{a}{b}+1$, то 0; 2) если $0<\frac{a}{b}-1$, то ∞; 3) в остальных случаях $\exp(\frac{c-d}{b})$.

При нахождении пределов последовательностей вид условия отличается от указанного выше только тем, что внутри конструкции "отображение(...)", определяющей рассматриваемую функцию, вместо утверждения "число(x)"берется утверждение "натуральное(x)". В этом случае можно формулировать также задачи на нахождение верхнего и нижнего пределов (логические символы "верхнийпредел", "нижнийпредел" вместо символа "предел"). Чтобы при стандартной прорисовке на экране можно было отличать пределы функций вещественного переменного от пределов последовательностей, в последних выражение под знаком предела заключается в фигурные скобки (это эквивалент использования во внутреннем представлении утверждения "натуральное(x)"вместо "число(x)").

Для нахождения точных верхних и нижних граней функций одного вещественного переменного на заданных множествах также применяются задачи на преобразование с целями "упростить", "одз". Например, для точной верхней грани условие задачи имеет вид "супр(образ(отображение(x число(x) $f(x)$)M))", где M - множество, на котором ищется точная верхняя грань. На экране оно прорисовывается как "sup(образ(λ_x($f(x)$, число(x)),M))".

Если нужно найти наибольшее либо наименьшее значение функции на некотором множестве, а также определить точки, в которых достигается такое значение, то применяются уже не задачи на преобразование, а задачи на описание. Целевая установка их стандартная - "неизвестные xy", "полный", "явное", "прямойответ", "упростить", "одз". Здесь x - множество, на котором достигается искомое наибольшее либо наименьшее значение, y - это значение. Условие задачи на описание (например, в случае наибольшего значения) имеет вид "Максимум(отображение(x число(x) $f(x)$) M x y)". Здесь M - множество, на котором ищется наибольшее значение. На экране такое условие прорисовывается как "Max($\lambda_x(f(x)$,число(x)),M, x, y)".

При поиске экстремумов тоже используется задача на описание. Условие ее имеет вид "экстремум(отображение(x число(x) $f(x)$)y z v)", где значением неизвестной y служит точка, в которой достигается экстремум функции; значением неизвестной z - значение функции в точке экстремума; значением неизвестной v - указатель типа экстремума - логический символ "минимум"либо "максимум". Область, в которой ищутся экстремумы, можно сужать, добавляя к списку

условий необходимые ограничения на y. Решатель пытается найти все экстремумы. Так, например, в задаче с условием "Extr($\lambda_x(\frac{\cos x}{\cos(2x)}$, число($x$)),$y$, z, v)" выдается ответ, состоящий из двух подслучаев: 1) $\exists k(y = 2\pi k$ & целое(k)), $z = 1, v = \min$; 2) $\exists k(y = \pi(2k+1)$ & целое(k)), $z = -1, v = \max$. При постановке задачи с экстремумами не обязательно требовать, чтобы указанные выше y, z, v были переменными. Вместо них могут быть подставлены любые выражения. Неизвестными, кроме входящих в y, z, v переменных, могут также быть какие-либо параметры, использованные при задании функции. Так, если вместо y подставить выражение без неизвестных, а неизвестными считать входящие в определение функции параметры, то смысл задачи будет состоять в отыскании таких значений параметров, для которых функция имеет в заданной точке y экстремум.

Чтобы получить качественное описание графика функции с помощью пределов и производных, применяются задачи на описание со списком целей "исследовать", "полный", "явное", "прямойответ", "функция", "неизвестные x". Условием такой задачи на описание является единственное равенство "$x =$ отображение(y число(y) $f(y)$)", определяющее исследуемую функцию. При решении задачи сразу же вводится ее блок анализа, в котором происходит логический вывод с постепенным накоплением информации о поведении функции x. В процессе вывода усматриваются те утверждения, которые целесообразно включать в итоговое описание свойств функции, и они переносятся из блока анализа в список условий задачи на описание. По исчерпании возможностей вывода выдается ответ - конъюнкция утверждений списка условий (в нем сохраняется и исходное равенство для x). В качестве примера приведем задачу с условием $x = \lambda_y(-y^3 + 3y,$ число(y)). Решатель выдает на нее следующий ответ (без учета повторения исходного условия): "функция(x)"; "Extr($x, 1, 2, \max$)"; "убывает ($x, (1, \infty)$)"; "Extr($x, -1, -2, \min$)"; "убывает($x, (-\infty, -1)$)"; "возрастает($x, (-1, 1)$)"; "область(x)= $\Re$ "; "мощность(корни($x, (1, \infty)$))=1"; "мощность(корни($x, (-1, 1)$))=1"; "мощность(корни($x, (-\infty, -1)$))=1".

Если в качественное описание графика нужно включить также указание на интервалы выпуклости-вогнутости, то к указанной выше целевой установке добавляется цель "выпуклавверх". Например, при наличии этой цели, на задачу с условием $x = \lambda_y(-y^3 + 3y^2,$ число(y)) решатель находит ответ: "функция(x)"; "Extr($x, 2, 4, \max$)"; "убывает($x, (2, \infty)$)"; "Extr($x, 0, 0, \min$)"; "убывает($x, (-\infty, 0)$)"; "возрастает($x, (0, 2)$)"; "область(x) = $\Re$"; "выпуклавниз($x, (-\infty, 1)$)"; "выпуклавверх($x, (1, \infty)$)"; "мощность(корни(x, $(2, \infty)$)) = 1"; "корни($x, (0, 2)$)=$\emptyset$"; "корни($x, (-\infty, 0)$)= $\emptyset$".

Кроме задач на общее качественное исследование функций, рассматриваются также некоторые дополнительные типы задач, сязанных с исследованием функций одного вещественного переменного. Все они имеют в составе своей целевой установки цели "исследовать", "полный", "явное", "прямойответ", "функция", "неизвестные x". К ним добавляется цель, уточняющая тип проводимого исследования, причем такая цель блокирует проведение общего качественного исследования (исключение здесь составляет указанный выше случай цели "выпуклавверх"). Решение этих задач осуществляется по указанной выше схеме - путем вывода следствий в блоке анализа и перенесении представляющих ценность результатов вывода в список условий внешней задачи на описание.

При исследовании функции на четность-нечетность и периодичность добавляется цель "четнаяфункция"; при исследовании на непрерывность - добавляется цель "непрерывно"; при исследовании на равномерную непрерывность добавляются цели "непрерывно"и "равномернонепрерывно". В случае исследования на непрерывность решатель анализирует типы точек разрыва, причем в случае исследования на равномерную непрерывность анализ точек разрыва блокируется. Приведем несколько простых примеров задач указанных типов. Так, при наличии цели "четнаяфункция"на задачу с условием

$$y = \lambda_x(\frac{\sin(2x)}{2} + \frac{\sin(3x)}{3} + \sin x, \text{число}(x))$$

решатель выдает ответ "область(y)= $\Re$"; "периодична$(y, 2\pi)$"; "нечетнаяфункция(y)". При наличии цели "непрерывно"на задачу с условием

$$y = \lambda_x(\left[\frac{1}{x}\right], \text{число}(x))$$

выдается ответ: "область(y)=$(-\infty, 0) \cup (0, \infty)$"; "$\forall n$(целое$(n)$ & $\neg(n = 0) \Rightarrow$ разрывпервогорода$(y, \frac{1}{n})$)"; "разрыввторогорода$(y, 0)$"; "непрерывно$(y, ((-\infty, 0) \cup (0, \infty)) \setminus$ класс$(x\ \exists n$(целое(n) & $\neg(n = 0)$ & $x = \frac{1}{n})))$)". При наличии целей "непрерывно" и "равномернонепрерывно"на задачу с условием

$$y = \lambda_x(\ln(x), \text{число}(x)\ \&\ 0 < x\ \&\ x < 1)$$

выдается ответ: "область$(y) = (0, 1)$"; "$\neg$(равномернонепрерывно$(y, (0, 1))$)"; "непрерывно$(y, (0, 1))$".

Для определения числа корней функции на заданном множестве создается задача на преобразование с целями "упростить", "одз" и условием "мощность(корни(отображение$(x\ \ A(x))\ \ f(x))))$)", где $A(x)$ - условие принадлежности аргумента x рассматриваемому множеству; $f(x)$ - выражение, определяющее значение функции. Если условие содержит параметры, то ответ задачи может иметь вид условного выражения, определяющего искомое число корней путем разбора случаев.

В случае функций нескольких переменных задачи на отыскание экстремумов и наибольших (наименьших) значений формулируются аналогично случаю одной переменной. Это задачи на описание с целями "полный", "явное", "упростить", "одз", "прямойответ", "неизвестные $z_1 \dots z_k$". В случае безусловного экстремума условие задачи имеет вид "экстремум(отображение$(x_1 \dots x_n$ и(число(x_1) ... число$(x_n))$
$f(x_1 \dots x_n))\ z_1\ z_2\ z_3)$"; в случае условного - к списку утверждений "число(x_1)", ..., "число(x_n)" внутри описателя "отображение" добавляются необходимые равенства и неравенства для варьируемых переменных $x_1, \dots, x_n$. Значением неизвестной z_1 служит набор значений указанных переменных, определяющий точку экстремума; z_2 - значение функции в этой точке; z_3 - тип экстремума (логический символ "минимум" либо "максимум"). В качестве примера приведем задачу на описание, имеющую единственную посылку $0 < a$ и условие "экстремум(отображение$(xyz$ и(число(x)число(y)число$(z)\ xy + xz + yz = a^2 \quad 0 < x \quad 0 < y \quad 0 < z)\ xyz)\ u\ v\ w)$". Решатель выдает на нее ответ: $u = (\frac{a}{\sqrt{3}}, \frac{a}{\sqrt{3}}, \frac{a}{\sqrt{3}}), v = \frac{a^3}{3\sqrt{3}}, w = \max$. Другой пример - задача с условием

"Максимум(отображение(xy и(число(x) число(y)) $-\sin(xy)+\sin(x)+\sin(y)$) класс($xy$ и(число(x) число(y) $0 \le x$ $0 \le y$ $x+y \le 2\pi$)) u v)", в которой неизвестными являются u, v (соответственно, множество точек, где достигается наибольшее значение, и само это значение). Решатель получает ответ $u = \left\{\left(\frac{2\pi}{3}, \frac{2\pi}{3}\right)\right\}, v = \frac{3\sqrt{3}}{2}$.

Задачи на вычисление интегралов любых типов (определенных, неопределенных, кратных) оформляются как задачи на преобразование со стандартными целями "упростить", "одз". В случае неопределенного интеграла условие имеет вид "Интеграл(
отображение(x число(x) $f(x)$))". В качестве ответа выдается какая-либо одна из первообразных, записанная в виде "отображение(x число(x) $g(x)$)". Разумеется, прорисовка интеграла выполняется формульным редактором в стандартном виде (после знака интеграла набирается подынтегральное выражение, и далее - как дополнительные множители - d,x; если подынтегральное выражение имеет вид дроби, то перенесение в ее числитель этих множителей не допускается). Так, на задачу с условием

$$\int \frac{1}{x\sqrt{x^4+2x^2-1}}\,dx$$

выдается ответ

$$\lambda_x\left(\frac{\arcsin(-\frac{1}{\sqrt{2}x^2}+\frac{1}{\sqrt{2}})}{2}, \text{число}(x)\right)$$

.

При вычислении определенного интеграла пределы интегрирования a, b извлекаются из списка утверждений, описывающих область определения интегрируемой функции. Условие задачи на преобразование при этом имеет вид "интеграл(
отображение(x и(число(x) $a \le x$ $x \le b$) $f(x)$))". На экране оно прорисовывается в обычном виде:

$$\int_a^b f(x)dx$$

.

Задача на вычисление двойного интеграла включает в себя описание области интегрирования, которое часто оказывается достаточно громоздким. Поэтому при формулировке таких задач для обозначения области интегрирования обычно вводится вспомогательная переменная, и в посылках задачи указывается определяющее ее равенство. Условие задачи имеет тогда вид "двойнойинтеграл(отображение(xy $(x,y) \in P$ $f(x,y)$))", где P - обозначение области интегрирования. Формульный редактор прорисовывает это условие в виде

$$\iint_P f(x,y)dxdy$$

(P располагается непосредственно под знаком двойного интеграла). Разумеется, можно и не выносить задание области интегрирования в список посылок, но тогда при изображении на экране получится громоздкая многоэтажная запись.

В качестве примера приведем задачу с посылкой $P =$ класс(xy число(x) & число(y) & $0 \le x$ & $0 \le y$ & $4x^2 - 3y^2 \le 4$ & $4y^2 - 3x^2 \le 4$) и условием

$$\iint_P (x^3y + xy^3)dxdy.$$

Решатель выдает для нее ответ 3.

Типичной при вычислении двойного интеграла является ситуация, когда область интегрирования задана не непосредственно с помощью неравенств, как в приведенном выше примере, а косвенно - через ограничивающие ее кривые. В этом случае она может быть задана с помощью выражения "областьграницы(G)", где G - выражение, определяющее объединение множеств точек указанных кривых. Здесь значением выражения "областьграницы(G)", как и значением выражения G, является не множество самих точек плоскости, а лишь множество их координат. Предполагается, что рассматриваемые кривые разбивают плоскость на некоторые компоненты связности, и в область интегрирования включаются все конечные компоненты (таким образом, она может оказаться и несвязной). В качестве примера приведем задачу с посылкой $P =$ областьграницы($set_{xy}(y = x$ & число(x) & число(y)) $\cup$ $set_{xy}(y = x + a$ & число(x) & число(y)) $\cup$ $set_{xy}(y = a$ & число(x) & число(y)) $\cup$ $set_{xy}(y = 3a$ & число(x) & число(y))) и условием

$$\iint_P (x^2 + y^2)dxdy.$$

Решатель выдает на нее ответ $14a^4$. Заметим, что использованное выше обозначение $set_{xy}F(x, y)$ - результат прорисовки формульным редактором выражения "класс(xy $F(x, y)$)".

При вычислении площадей плоских множеств, объемов и площадей поверхностей соответствующее множество уже является не множеством числовых пар либо троек, а множеством точек плоскости либо трехмерного пространства, имеющих своими координатами эти пары либо тройки. Поэтому вместо выражения вида "областьграницы(...)"для задания множества используется выражение вида "точки(областьграницы(...)K))", где K - прямоугольная система координат. В качестве примера вычисления площади плоского множества приведем задачу с посылками "$P =$ точки(областьграницы($set_{xy}((x^2 + y^2)^3 = a^2(x^4 + y^4)$ & число(x) & число(y))), K), "прямкоорд(K)", "$0 < a$" и условием "площадь(P)". Решатель выдает на нее ответ $3\pi\left(\frac{a}{2}\right)^2$.

Пример на вычисление объема - задача на преобразование с посылками "$P =$ точки($set_{xyz}(x^2+y^2 \le a^2$ & $0 \le az$ & $az \le a^2-2y^2$ & число(x) & число(y) & число(z)), K)", "прямкоорд(K)", $0 < a$ и условием "объем(P)". На нее выдается ответ $\frac{(\pi+4)a^3}{4}$.

Наконец, пример вычисления площади поверхности - задача с посылками "$P =$ точки($set_{xyz}(2az = x^2 + y^2$ & $x^2 + y^2 \le a^2$ & $0 \le y$ & $0 \le x$ & $y \le x$ & число(x) & число(y) & число(z)), K)", "прямкоорд(K)"и условием "площадь(P)". На нее выдается ответ $\frac{(2\sqrt{2}-1)\pi a^2}{12}$. Заметим, что при вычислении

площади поверхности допустимо ее параметрическое задание. Пример такого задания - задача с посылками
"P = точки($set_{xyz}(\exists uv(x = u\cos v \,\&\, y = u\sin v \,\&\, z = 4v \,\&\, x^2{+}y^2 \le 9 \,\&$ число(u) $\&$ число$(v) \,\&\, 0 \le z \,\&\, z \le 8\pi)), K)$", "прямкоорд$(K)$"и условием "площадь$(P)$".

В задачах на исследование сходимости рядов используется запись ряда как последовательности своих частичных сумм: "отображение(n натуральное(n) суммавсех (отображение(i и(целое(i) $1 \le i$ $i \le n)$ $a(i)$))))". На экране такая запись прорисовывается в виде:

$$\lambda_n \left(\sum_{i=1}^{n} a(i), \text{натуральное}(n) \right).$$

В качестве примеров задач, связанных с исследованием сходимости рядов, приведем задачу на доказательство с посылкой $|a| < 1$ и условием

$$\text{сходится}(\lambda_n \left(\sum_{i=1}^{n} (a^m \sin(bn)), \text{натуральное}(n) \right)),$$

а также задачу на описание с условием

$$\text{сходится}(\lambda_n \left(\sum_{i=2}^{n} (\sqrt{i+1} - \sqrt{i})^p \ln \frac{i-1}{i+1}, \text{натуральное}(n) \right)).$$

Эта задача имеет стандартный список целей "полный", "явное", "прямойответ", "упростить", "одз", "неизвестные p". Решатель выдает на нее ответ $0 < p$, число(p).

Для получения разложения в ряд Тейлора служит задача на преобразование с целями "рядтейлора x a", "упростить", "одз". Здесь x - переменная, по которой выполняется разложение; a - выражение, определяющее точку, в окрестности которой происходит разложение. Пример - задача с условием $\sin x + \cos x$, на которую выдается ответ

$$\sum_{n=0}^{\infty} \frac{x^n (-1)^{\left[\frac{n}{2}\right]}}{n!}.$$

Заметим, что бесконечная сумма во внутреннем представлении имеет вид "суммавсех (отображение(n и(целое(n) $k \le n)$ $a(n)$))", где k - значение, с которого начинается суммирование величин $a(n)$.

Если нужно получить лишь конечное число членов разложения в ряд Тейлора, то вместо цели "рядтейлора ..." используется цель "формулатейлора x a n". Здесь x, a - те же, что и выше; n - степень, до которой (включительно) выписываются члены разложения. Пример - задача с условием $\frac{x}{\exp x - 1}$ и целью "формулатейлора x 0 4", на которую выдается ответ $1 - \frac{x}{2} + \frac{x^2}{12} - \frac{x^4}{144}$.

Решатель может обращаться к задаче на получение конечного числа членов разложения в ряд Тейлора при решении других задач, например, при подборе таких значений параметров выражения $f(x)$, что оно приобретает заданный

"порядок малости"в окрестности заданной точки. Пример - задача на описание с посылкой "стремится(x 0 0)"(прорисовывается как $x \to 0$), условием $-a\sin x - b\tan x + x = O(x^5)$ и целями "полный", "явное", "прямойответ", "упростить", "одз", "неизвестные ab". На нее выдается ответ $a = \frac{2}{3}, b = \frac{1}{3}$.

При разложении в ряд Фурье используется целевая установка "рядфурье x a b", "упростить", "одз". Здесь x - переменная, по которой ведется разложение; a, b - концы отрезка, на котором происходит разложение. Пример - задача с условием x^3 и целью "рядфурье x $-\pi$ π". На нее выдается ответ

$$2\sum_{n=1}^{\infty} \frac{(-\pi^2 n^2 + 6)(-1)^n \sin(nx)}{n^3} \ .$$

Заметим, что для выбора целевой установки задачи создан специальный интерфейс, так что вводить указанные выше цели "рядтейлора ...", "формулатейлора ..."(или какие-либо другие стандартные комбинации целей) в явном виде вручную нет необходимости.

Наконец, отметим задачи на суммирование рядов. Они оформляются как обычные задачи на упрощение (цели "упростить", "одз"). Пример - задача с условием

$$\sum_{i=0}^{\infty} \frac{x^{4i+1}}{4i+1} \ ,$$

на которую решатель дает ответ

$$\frac{\ln\left|\frac{x+1}{-x+1}\right|}{4} + \frac{\arctan x}{2} \ .$$

6. Дифференциальные уравнения.

В случае дифференциального уравнения в качестве неизвестной выступает функция. Уравнение связывает значение этой функции y в некоторой точке x, значения ее производных различных порядков в этой же точке, и само значение x, причем данная связь имеет место для всех точек некоторой области G. Формальная запись такой связи должна была бы иметь вид $\forall x(x \in G \Rightarrow F(x, y(x), \frac{dy(x)}{dx}, \ldots))$. Однако, используемые при решении дифференциальных уравнений преобразования никак не затрагивают часть этой записи, находящуюся слева от импликации, а также сам квантор общности - они должны бы были просто переписываться "вхолостую". Поэтому в решателе дифференциальное уравнение записывается без внешнего квантора общности, а условия на область, в которой изменяется варьируемая переменная x, вынесены в список посылок. Фактически, это еще одно проявление "контекстной семантики" - для адекватной логической формализации нужно на каждом шаге выполнять обратный переход, преобразуя текущие записи уравнений к указанному выше виду с квантором общности. Чтобы явно указать на наличие такой модифицированной записи, задача на описание, в которой нужно решить дифференциальное уравнение относительно y, снабжается, кроме стандартных целей "неизвестные y", "полный", "явное", "прямойответ", "упростить", "одз", также целью "связка x". Эта цель указывает, что неизвестная y не должна зависеть от параметра

x, входящего в посылки. Фактически она несет несколько большую нагрузку, встречаясь только в задачах на решение дифференциальных уравнений и корректируя действия приемов в соответствии с обычными соглашениями, принимаемыми при их решении. Важным таким соглашением является представление ответа не в виде явного равенства для функции y, а лишь в виде равенства для значения $y(x)$; более того, последнее равенство может даже не быть явно разрешенным относительно $y(x)$. Обычно ответ (для дифференциального уравнения первого порядка) представляется в виде параметрического описания $\exists C(F(x, y(x), C))$ либо дизъюнкции нескольких таких описаний. Как и условие задачи, этот ответ для получения адекватной в логическом отношении записи должен преобразовываться к виду $\exists C(\forall x(x \in G \Rightarrow F(x, y(x), C)))$, причем сама область G (возможно, суженная в процессе решения по сравнению с исходными ограничениями на x, имевшимися в списке посылок)обычно в ответах явно не указывается. В качестве примера приведем задачу на описание с посылкой "число(x)", условиями "функция(y)"и "$x^2\frac{dy(x)}{dx} + y(x)^2 = xy(x)\frac{dy(x)}{dx}$. Эта задача имеет указанный выше список целей. Решатель выдает на нее ответ $\exists C(x\ln(Cy(x)) - y(x) = 0$ & число$(C)) \vee y(x) = 0$.

Если нужно решить дифференциальное уравнение с заданными дополнительными условиями, позволяющими идентифицировать произвольные постоянные, то дополнительные условия просто присоединяются к списку условий задачи на описание. Например, если требуется найти решение дифференциального уравнения $y(x)\frac{d^2y(x)}{dx^2} = 2x\left(\frac{dy(x)}{dx}\right)^2$, принимающее в точке 2 значение 2 и имеющее в этой точке производную 0.5, то дополнительно вводятся условия $y(2) = 2$ и $\frac{dy(x)}{d(x=2)} = 0.5$. Решатель выдает на эту задачу ответ $y(x) = -2^{\frac{4}{5}}\sqrt[5]{\frac{x+2}{2(x-3)}}$.

Если для уравнения, не разрешенного относительно производной, ответ получается в параметрической форме: $y = f(t, C); x = g(t, C)$, то такое параметрическое задание записывается решателем в виде $y(g(t, C)) = f(t, C)$. Эта запись позволяет экономить на вводе обозначений для вспомогательных функций $y(t)$,$x(t)$ (ведь $y(t)$ - попросту новая функция, связанная с искомой функцией y лишь косвенным образом). Пример: решая задачу с условием $\left(\frac{dy(x)}{dx}\right)^3 + y(x)^2 = xy(x)\frac{dy(x)}{dx}$, система выдает ответ:
$\exists C\left(y(-\frac{C+3t}{\sqrt{C+2t}}) = -\frac{t^2}{\sqrt{C+2t}}\ \&\ \text{число}(C)\right) \vee \exists C\left(y(\frac{C+3t}{\sqrt{C+2t}}) = \frac{t^2}{\sqrt{C+2t}}\ \&\ \text{число}(C)\right) \vee$
$\vee\ y(x) = 0$.

Для обозначения производной k - го порядка ($k > 1$) в дифференциальных уравнениях используется уже встречавшаяся выше запись "частнпроизв(отображение(y число(y) $f(y)$) k x)", прорисовываемая на экране как $\frac{d^k f(x)}{dx^k}$. Ответ дифференциальных уравнений k - го порядка представляестя в виде параметрического описания $\exists c_1 \ldots c_k(\ldots)$ либо дизъюнкции таких описаний (в вырожденных подслучаях длина связывающей приставки может быть меньше k или вообще отсутствовать квантор существования). Так, на уравнение $x^3\frac{d^2y(x)}{dx^2} = \left(-x\frac{dy(x)}{dx} + y(x)\right)\left(-x\frac{dy(x)}{dx} + y(x) - x\right)$ выдается ответ $\exists cd(y(x) = -x\ln(c\ln x + d)$ & число(c) & число$(d)) \vee \exists c(y(x) = cx$ & число$(c))$.

В случае системы дифференциальных уравнений применяется целевая установ-

ка того же типа, что и в случае одного уравнения - просто в списке неизвестных указывается несколько переменных.

7. Аналитическая геометрия и линейная алгебра.

Геометрические задачи на вычисление, использующие векторную алгебру либо непосредственно связанные с отысканием неизвестных векторов, формализуются аналогично рассмотренным выше вычислительным задачам по геометрии. Небольшое исключение здесь составляют задачи на решение уравнений в векторных операциях, формализуемые так же, как задачи на решение уравнений в элементарной алгебре. При использовании в задаче аффинной либо прямоугольной системы координат K в список ее посылок может быть занесено равенство $K = (A, B, C)$ либо $K = (A, B, C, D)$, вводящее обозначения A, B, C, D для начала координат и концов координатных векторов. Этого можно не делать, если каких-либо иных условий на точки A, B, C, D в задаче не накладывается. Во всяком случае, для прямоугольной системы координат требуется вводить посылку "прямкоорд(K)". Приведем два простых примера задач на координаты точек. В первой из них известны координаты (-4,2) точки окружности и координаты (2,0) точки ее касания с осью абсцисс прямоугольной системы координат; найти требуется координаты центра окружности. Эта задача формализуется как задача на описание с целями "полный","явное","прямойответ", "упростить","одз", "неизвестные x", "известно". Она имеет посылки "прямкоорд(K)", "$K = (A, B, C)$", "касательная(прямая(AB) окружность(DE))", "коорд(E, K) = (-4,2)", "$F \in$ прямая(AB)", "$F \in$ окружность(DE)", "коорд(F, K) = (2,0)", "планиметрия"и условие "$x =$ коорд(D, K)". Во второй задаче нужно найти координаты центра вписанной в треугольник окружности, если известны координаты его вершин. Здесь уже не нужно явно вводить обозначения для тройки точек, определяющих систему координат. Посылки задачи имеют вид: "треугольник(ABC)", "вписана(окружность(MN)фигура(набор(ABC)))", "прямкоорд(K)", "коорд(A, K) = (9,2)", "коорд(B, K) = (0,20)", "коорд(C, K) = (-15,-10)", "планиметрия".

Задачи на определение геометрического места точек формализуются в виде задач на преобразование. Условием такой задачи служит выражение "класс(X $P(X)$)", где $P(X)$ - условие на принадлежность точки рассматриваемому множеству (прорисовывается оно как $set_X P(X)$). Задача имеет цели "упростить", "одз", "класс", причем последняя цель указывает на необходимость преобразовать условие к виду, не использующему описателей "класс"и "отображение". В качестве примера рассмотрим задачу на нахождение геометрического места точек, сумма квадратов расстояний которых до двух заданных точек A, B равна $2a^2$; при этом предполагается известным расстояние $2c$ между точками A, B, и $c < a$. Посылки данной задачи суть: "$\neg(A = B)$"; "расстояние(AB) = $2c$"; "$c < a$". Условие имеет вид "set_X(точка(X) & $l(AX)^2 + l(BX)^2 = 2a^2$)". Решатель выдает ответ "окр(доляотрезка($B, A, 1/2$),$\sqrt{a^2 - c^2}$)", то есть окружность с центром в середине отрезка AB и радиусом $\sqrt{a^2 - c^2}$ (см. подраздел "элементарная геометрия"главы 2).

В задачах на уравнения кривых и поверхностей для обозначения уравнения кривой либо поверхности M используется запись вида "коорд(M K) =

$set_{xy}(F(x,y)=0$ & число(x) & число$(y))$" либо "коорд$(M\ K) = set_{xyz}(F(x,y,z) = 0$ & число(x) & число(y) & число$(z))$". В случае кривой в трехмерном пространстве вместо одного равенства $F(x,y,z) = 0$ используются два - $F_1(x,y,z) = 0$ & $F_2(x,y,z) = 0$. В качестве простого примера такой задачи приведем задачу на нахождение уравнения стороны BC треугольника ABC, у которого известны уравнения сторон AB, AC и координаты точки пересечения высот F. Посылки здесь имеют вид: "треугольник(ABC)"; "прямкоорд(K)"; "коорд(прямая$(AB)K)$ $= set_{xy}(x + 3y - 1 = 0$ & число(x) & число$(y))$"; "коорд(прямая$(AC)K) = set_{xy}(3x + 5y - 6 = 0$ & число(x) & число$(y))$"; "перпендикулярно(прямая(BE) прямая$(AC))$"; "перпендикулярно(прямая(AD)прямая$(BC))$";
"$F \in$ прямая(BC)"; "$F \in$ прямая(AD)"; "коорд$(F\ K) = (0,0)$". Условие задачи - "$z =$ коорд(прямая$(BC)K)$"; цели - "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные z". Решатель выдает ответ "$z = set_{xy}(39x - 9y - 4 = 0$ & число(x) & число$(y))$".

Ряд задач по аналитической геометрии связан с определением взаимного расположения прямых, точек и плоскостей. Для описания способа такого расположения в логический язык введены несколько специальных предикатных символов и символов констант. Так, при описании взаимного размещения двух прямых A, B используется запись "двепрямые$(A\ B\ s)$", где s - константа для конкретного типа размещения - "пересекаются", "параллельно", "равны". Если в задаче требуется определить, при каких условиях на параметры уравнений имеет место заданный тип взаимного расположения, то она формализуется как задача на описание, имеющая своим условием обозначение типа расположения и цели "полный", "прямойответ", "редакция". Например, для определения условия пересечения двух прямых, одна из которых задана каноническим уравнением, а другая - параметрическим, создается задача на описание с посылками "Прямая(P)", "коорд$(P,K) = set_{xy}(Ax + By + C = 0$ & число(x) & число$(y))$", "Прямая(Q)", "коорд$(Q,K) = set_{xy}(\exists t(x = at + p$ & $y = bt + q$ & число$(t)))$" и условием "двепрямые$(P, Q,$ пересекаются$)$". На нее выдается ответ "$\neg(aA + bB = 0)$".

Задачи на определение вида кривой либо поверхности второго порядка, заданной своим уравнением, формализуются аналогично задачам на качественное исследование вида графика функции вещественного переменного. Они представляются как задачи на описание с целями "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные E", "исследовать", к которым в случае кривых второго порядка добавляется цель "линия", а в случае поверхностей - цель "эллипсоид". Условием служит уравнение исследуемой кривой E. Решение такой задачи происходит путем вывода следствий в ее блоке анализа, с отбором и перенесением в список условий элементов стандартной характеризации кривой. В качестве примера приведем задачу с условием "коорд$(E,\ K) = set_{xy}(4x^2 - y^2 - 16x - 6y + 3 = 0$ & число(x) & число$(y))$"и посылкой "прямкоорд(K)". Решатель выдает на нее ответ (за вычетом повторного переписывания уравнения кривой): "гипербола(E)", "каноничкоорд(Q,E)", "$Q = (A,B,C)$", "коорд$(A,K) =$ (2,-3)", "коорд$(B,K) =$ (3,-3)", "коорд$(C,K) =$ (2,-2)",
"мнимаяполуось$(E) = 2$", "действполуось$(E) = 1$", "коорд$(E,Q) = set_{xy}(4x^2 - y^2 - 4 = 0$ & число(x) & число$(y))$".

Для матриц в задачах используется два типа обозначений. Если матрица име-

ет фиксированный порядок n, то она задается явным перечислением своих элементов, при помощи выражения "строки(набор($A_1 \dots A_n$))", где A_i - выражение вида "набор $(a_{i1} \dots a_{in})$", задающее i-ю строку. Фактически при этом матрица вводится и прорисовывается на экране в обычной записи. Если же порядок матрицы - переменная величина, то она задается выражением вида "отображение(ij и(принадлежит(i номера(1 n))принадлежит(j номера(1 n))) $a(i,j)$)". В обоих случаях решатель рассматривает матрицу как функцию от двух переменных, определенных на начальном отрезке натурального ряда, и применяет при работе с ней все приемы, которые связаны с такими функциями (например, при расшифровке записи "значение $(M\ (i,j))$", где M - явно заданная матрица). Обычно при задании матриц "общего вида"приходится использовать многоэтажные условные выражения, определяющие для различных частей матрицы различные аналитические зависимости элементов от их номеров. В качестве примера приведем задачу на вычисление определителя матрицы n- го порядка, у которой над главной диагональю расположены элементы, равные номеру столбца; под этой диагональю - равные номеру столбца, взятому с минусом; левый верхний элемент равен 1, а прочие элементы равны 0. Условие этой задачи на преобразование имеет вид "определитель(отображение(ij и(принадлежит(i номера(1 n))принадлежит (j номера(1 n))) вариант($i < j$ j вариант($j < i$ $-j$ вариант($i = 1$ 1 0))))))". Единственная посылка задачи - "натуральное(n)"; цели - "упростить", "одз". Решатель выдает ответ $n!$.

В задачах на нахождение собственных значений и собственных векторов условия имеют вид "собствзначение(A,x,y)", "собствектор(A,x,z)". Здесь матрица A задается равенством, помещаемым в посылки задачи; x,y,z - неизвестные задачи. Напомним (см. главу 2), что x - собственное значение; y - его кратность; z - собственный вектор, отвечающий данному собственному значению. В ответе собственный вектор z описывается как элемент множества линейных комбинаций некоторого списка векторов. Пример - задача с посылкой

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

и указанными выше двумя условиями. На нее выдается ответ, состоящий из двух подслучаев:

$$x = -2, y = 1, z \in \text{линкомбинации} \left\{ \begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\};$$

$$x = 2, y = 3, z \in \text{линкомбинации} \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \right\}.$$

Задачи на преобразование полиномиальной матрицы к нормальной диагональной форме и на приведение матрицы к жордановой нормальной форме суть задачи на описание с условиями "каноничматрица(A,k,x)"и "жордформа(A,x,y)"

соответственно. Здесь A - явно задаваемая в условии задачи матрица; k - переменная, относительно которой рассматриваются многочлены - элементы матрицы A. x - неизвестная, значением которой служит результат приведения матрицы к соответствующему виду; y - неизвестная, определяющая матрицу, приводящую исходную к жордановой форме. Цели задачи суть "полный", "явное", "прямойответ", "одз", упростить", "неизвестные x"(либо "неизвестные xy").

3.2 Задачник логической системы

Оглавление задачника

В системе имеется сборник задач, в котором сохраняются (для нужд тестирования и регулировки) рассматривавшиеся при обучении задачи либо вводятся новые задачи. Этот сборник снабжен древовидным оглавлением, разбивающим его на разделы, подразделы, и т.п. Вход в оглавление задачника осуществляется из главного меню при нажатии клавиши "з"(здесь и далее, если не оговорено особо, все буквенные названия клавиш - русские) либо нажатии левой кнопки мыши на пункте меню "Оглавление задачника". После этого происходит переход к тому пункту оглавления, с которым велась работа при предыдущем пребывании в задачнике. Переход между пунктами оглавления выполняется с помощью клавиш курсора: нажатие клавиши "курсор влево"приводит к переходу в надраздел текущего раздела; клавиши "курсор вверх"и "курсор вниз"позволяют выбирать текущий пункт в текущем разделе; клавиша "курсор вправо"приводит к просмотру содержимого текущего пункта. Для ускоренного выбора пункта можно использовать курсор мыши: нажатие левой клавиши на выбранном пункте приводит к переходу внутрь этого пункта; нажатие (в любом месте) правой клавиши мыши - к возвращению в надраздел. Пункты каждого раздела пронумерованы, причем если пункт соответствует входу в подраздел, то после его номера стоит закрывающая скобка, если же пункт соответствует отдельной задаче задачника, то после его номера стоит точка. Из любого пункта оглавления можно вернуться к главному меню, нажав клавишу "End". Пункты корневого раздела имеют названия "Дискретная математика", "Алгебра множеств", "Элементарная алгебра", "Элементарная геометрия", "Математический анализ", "Дифференциальные уравнения", "Аналитическая геометрия", и др.

Оглавление задачника организовано таким образом, что все его концевые пункты (соответствующие отдельным задачам) собраны в специальные концевые разделы. Все пункты такого концевого раздела - только концевые. Обычно в концевом разделе перечисляются только номера задач, после которых стоят три тире. Прочие (неконцевые) пункты оглавления имеют текстовые подзаголовки. В принципе, с любым пунктом оглавления (концевым либо неконцевым) можно связать произвольные текстовые примечания. Для этого достаточно выбрать нужный пункт и нажать клавишу "р" (рус.) - возникает курсор текстового редактора (особенности текстового редактора решателя - см. в последующих разделах, посвященных его интерфейсу). После завершения редактирования (оно происходит по нажатии "Enter") сохраняется измененный текст пункта.

После выбора текущего концевого пункта и нажатия клавиши "курсор вправо" происходит переход к просмотру задачи этого пункта. Все задачи одного и того же концевого раздела соединены при таком просмотре в одну "ленту", которую можно прокручивать вверх и вниз при помощи клавиш "курсор вверх - вниз" и "Page Up -

Down", не выходя обратно в оглавление задачника. Одна задача отделяется на этой "ленте" от другой сплошной горизонтальной линией. Кроме указанных выше, при просмотре задач одного концевого раздела удобно использовать клавиши "Ctr-Page Up" и "Ctr-Page Down". Первая из них позволяет переходить к просмотру начала предыдущей задачи, вторая - к просмотру начала следующей задачи.

Просмотр задач и основные операции над задачами

Область между горизонтальными линиями, ограничивающими описание задачи (для последней задачи нижняя линия отсутствует), называется полем задачи. Элементы задачи размещаются в ее поле следующим образом. Вначале идут посылки задачи. Затем располагается текст-формульный элемент, задающий целевую установку задачи. Далее (кроме задач на исследование) идут условия задачи (в случае задач на доказательство либо преобразование - единственное условие). Если задача ранее была решена системой, то в конце располагается найденный на нее ответ. Задача по геометрии может быть сопровождена чертежом, который размещается до ее посылок. Если задача не имеет невырожденных посылок (отличных от константы "истина" и простейших указателей на типы значений переменных), то целевая установка размещается непосредственно в начале поля.

В некоторых разделах (теория вероятностей, элементарная физика, комбинаторика) логическая формализация текста задачи выглядит громоздко и трудна для восприятия. Поэтому в решателе предусмотрена возможность сохранять также исходный текст задачи. Если этот текст был сохранен, и при просмотре задачи нажимается клавиша "н", то текст появляется на экране. Данная операция адресуется той задаче, к которой относится верхняя из имеющихся на экране отделяющих горизонтальных линий. Чтобы вернуться к просмотру формальной версии условия, нажимается "End" или "Esc". При нажатии клавиши "н" формальная запись задачи пропадает с экрана. Поэтому более удобным для ее сопоставления с исходным текстом задачи является другой режим, включаемый при нажатии на клавишу "+" или выборе в верхнем меню пункта "+". Тогда в нижней части экрана возникает выделенный голубым цветом текст задачи. Как и выше, он относится к задаче, определяемой по верхней горизонтальной линии на экране. Далее можно выполнять произвольную прокрутку - этот текст будет сохраняться до нажатия клавиши пробела.

Чтобы удобнее было анализировать логическую формализацию задачи, можно использовать специальный режим просмотра, при котором на экран будут выводиться пояснения к встречающимся в формулах понятиям. Для перехода в этот режим достаточно нажать клавишу "?" либо выбрать в верхнем меню пункт "?". Чтобы получить пояснения к какому-либо понятию, нужно поместить на него курсор мыши и нажать левую кнопку. Пояснения прорисовываются голубым цветом в нижней части экрана и исчезают при нажатии любой клавиши. Чтобы выполнять описываемые далее операции над задачами, связанные с выделением их элементов, следут сначала выйти из режим просмотра пояснений, для чего нажать клавишу пробела.

Независимо от режима просмотра пояснений к задаче, можно входить в оглавление логического языка системы из произвольного оглавления либо из общего просмотра списка задач. Для этого достаточно нажать клавишу F2.

Существуют различные режимы прорисовки задачи. По умолчанию, задачи прорисовываются с помощью стандартной математической записи и с пропуском ряда

простейших посылок и условий. Чтобы перейти к режиму, в котором используется внутренний логический язык (далее называемому скобочной записью), нажимается клавиша "Ctr-c"; она же возвращает обратно в режим стандартной математической записи. Для перехода в режим полного просмотра посылок и условий и выхода из этого режима служит клавиша "ы".

Для выполнения различных операций с задачей и ее элементами выполняется выбор задачи либо ее элементов с помощью мыши. Курсор мыши подводится к выбираемому элементу либо помещается в поле выбираемой задачи. При выборе элемента задачи нажимается левая клавиша мыши, при выборе всей задачи - правая. Чтобы произошел выбор задачи, на экране обязательно должна быть прорисована ее верхняя отделяющая линия. Выбранный элемент перекрашивается в голубой цвет. Чтобы отменить выбор элемента, выполняется повторно такое же нажатие клавиши мыши. Если нужно отменить выбор сразу всех выбранных на текущий момент элементов и задач, нажимается клавиша пробела. Можно выделять не только отдельную посылку либо условие задачи, но и ее фрагмент. Для этого, сразу после выделения посылки либо условия F, следует нажать клавишу "курсор вправо". Тогда будет выделен (перекрашен в голубой цвет) первый операнд F. Далее клавишами "курсор вправо - курсор влево" можно переходить от операнда к операнду на одном уровне; клавишей "курсор вниз" - переходить к подоперандам текущего операнда; клавишей "курсор вверх" - возвращаться к надоперанду. Нажатие любой клавиши, отличной от клавиш курсора, завершает данный режим выделения фрагмента.

Перечислим некоторые операции, которые можно совершать после выделения терма задачи. Прежде всего, нажатие клавиши "с" позволяет переходить от стандартной математической записи этого терма к скобочной и обратно. При нажатии клавиши "Enter", включается режим вставки формульным редактором нового терма задачи непосредственно перед выделенным термом (если он располагается до целевой установки, то становится посылкой задачи, если после - условием). Ниже будут приведены основные правила использования формульного редактора при вводе элементов задачи. Если нужно вставить новый терм перед выделенным термом с помощью текстового редактора (то есть в скобочной записи), то нажимается клавиша "Т" (русск.). Для изменения ранее набранного терма имеется три возможности. Первая из них - использование текстового редактора для работы с текстом скобочной записи этого терма. Для входа в этот режим нажимается "Ctr - т" (т - русск.). Вторая - использование формульного редактора для повторного набора новой версии терма. Новая версия набирается непосредственно под старой, которая во время набора сохраняется, а затем удаляется. Для входа в этот режим используется "Ctr-ф". Наконец, можно войти в формульный редактор для продолжения набора выделенного терма, начиная с его конца. При необходимости (используя Backspace) здесь можно сначала исключить часть ранее набранных последних символов, а затем добавить новые. Для входа в такой режим нажимается "ф". Чаще всего для изменения ранее введенного терма применяется первый режим (скобочной записи), так как он позволяет вносить изменения в любом месте текста. Чтобы удалить выделенный элемент задачи, нажимается клавиша "Ctr-Del". Если была выделена вся задача, то нажатие "Ctr-Del"удаляет всю задачу целиком. Выделенную задачу можно скопировать нажатием клавиши "к"(русск.). Если выделена только одна задача, то копия располагается в конце текущего списка задач (т.е. текущего концевого раздела оглавления задачника). Если нужно разместить эту копию перед некоторой ранее введенной задачей списка (текущего либо относящегося к другому разделу), то перед нажатием

клавиши "к"выделяется также последняя задача. Чтобы исключить из задачника ранее найденный на выделенную задачу ответ, нажимается клавиша "Ctr-F4". Выделенный терм задачи можно скопировать, выделив предварительно тот терм, перед которым нужно поместить копию (если ее нужно поместить в конец списка посылок либо списка условий, то выделяется вся задача) и нажав затем клавишу "Insert". Аналогичным образом можно перенести выделенный терм на новое место; для этого служит клавиша "Ctr-Insert".

Если выделить несколько термов либо фрагментов термов задачи, то их можно использовать при наборе новых термов формульным редактором. Следует лишь запомнить порядок, в котором онм выделялись. Если в режиме формульного редактора нажать сначала "Insert", а затем номер (начиная с 1 и не более 9) выделенного указанным образом терма, то произойдет автоматическая вставка его в набираемый терм начиная с текущей позиции. Такую вставку заданного терма в процессе набора можно выполнять многократно.

3.3 Ввод новой задачи в задачнике

Ввод новой задачи в задачнике инициируется одним из двух способов. Первый из них состоит в том, чтобы перейти в конец просматриваемого списка задач и нажать клавишу "з"(если нужно ввести новую задачу перед ранее введенной, то сначала выделяется последняя задача, а затем нажимается "з"). Тогда возникает новая горизонтальная линия, которая является первым элементом новой задачи. Второй способ начать ввод новой задачи - выйти из просмотра списка задач в концевой раздел оглавления задачника, соответствующий данному списку, и ввести новый его концевой пункт, который автоматически становится заготовкой новой задачи. Если этот пункт вводится в конце раздела, то нажимается клавиша "к"(русск.); если же его нужно ввести перед некоторым другим пунктом, то происходит выбор последнего (клавишами "курсор вверх- вниз") и нажимается "К". При входе в список задач через новый концевой пункт оглавления попадаем на начало новой задачи, которое состоит из единственного ее элемента - верхней горизонтальной отделяющей линии. Если новая задача вводится перед ранее введенной нажатием клавиши "з", то она автоматически при этом выделяется (ее верхняя линия окрашена в голубой цвет).

После того, как введена верхняя отделяющая линия задачи, можно переходить к вводу посылок, условий и целевой установки задачи. В принципе, их допустимо набирать в произвольной последовательности. Рекомендуемым является следующий порядок, при котором уменьшается число операций с клавиатурой: сначала вводятся посылки задачи, затем целевая установка, и в конце - условие (условия). Чертеж геометрической задачи обычно вводится в последнюю очередь либо формируется автоматически (путем нажатия клавиши "Ч"), хотя можно начинать ввод задачи и непосредственно с чертежа. Для ввода очередной посылки нажимается "Enter чтобы войти в формульный редактор,- либо "Т"(русск.) - чтобы войти в текстовый редактор (последнее выполняется лишь в исключительных случаях, если для рассматриваемых логических символов еще не обеспечена их прорисовка формульным редактором). Вырожденные посылки, указывающие тип значений переменных либо необходимые для указания области допустимых значений, вводить вообще не нужно - они создаются решателем автоматически (кроме случаев, когда тип значения переменной не подсказывается выражениями из описания задачи, в которых эта пе-

ременная встречается). По окончании ввода посылок (если посылок нет, то сразу же) начинается формирование целевой установки задачи. Здесь имеется два режима - с использованием оглавления типов целевых установок и путем непосредственного набора перечня целей текстовым редактором. Рекомендуется использовать первый режим. Для входа в него нажимается клавиша "ц". Эта же клавиша используется для изменения ранее введенной целевой установки - предварительно надо выделить задачу либо убедиться, что задача является последней в текущем списке задач. После нажатия клавиши "ц"на экране появляется раздел оглавления типов целевых установок, с которым происходила работа в предыдущий раз. Это оглавление (и вообще все оглавления решателя) устроено аналогично оглавлению задачника. Корневой раздел оглавления типов целевых установок содержит следующие пункты:

1) "Доказать утверждение". Этот пункт выбирается при создании задачи на доказательство. После выбора его ("курсор вверх-вниз") и нажатия "курсор вправо" автоматически выбирается тип задачи "доказать"и происходит возвращение в набор текста задачи.

2) "Проверить истинность утверждения". Здесь создается задача на описание, предназначенная для проверки истинности ее единственного условия. Цели ее суть "полный", "явное", "прямойответ", "одз". Возможные ответы (кроме отказа) - "истина"либо "ложь". Как и выше, для выбора данной целевой установки нажимается "курсор вправо".

3) "Преобразовать выражение". Этот пункт является подразделом оглавления, в котором собраны различные целевые установки задач на преобразование (см. ниже).

4) "Найти значения неизвестных". Этот пункт является подразделом оглавления, в котором собраны различные целевые установки задач на описание (см. ниже), связанных с нахождением неизвестных.

5) "Исследовать свойства объекта". Этот пункт является подразделом оглавления, в котором собраны различные целевые установки задач на общую характеризацию свойств объектов различных типов. Все они оформляются как задачи на описание, причем ответом служит некоторая группа утверждений, полученных в процессе анализа ситуации и дающих типовую характеризацию требуемого вида.

6) "Переформулировать условие". Создается задача на описание, имеющая цели "полный", "прямойответ", "редакция". Это - эквивалент словесной формулировки "найти условия, при которых ...", означающей необходимость эквивалентного преобразования исходной системы условий к более простому и явному виду.

7) "Построить график". Пункт позволяет создавать задачу на построение графика. Это - задача на преобразование с целями "числзначение", "график". Ее условие - выражение, определяющее зависимость, для которой строится график. При решении задачи инициируется диалог для определения варьируемой переменной, промежутка ее значений и значений фиксированных параметров. Для работы с возникающим после этого графиком предусмотрен специальный интерфейс, позволяющий изменять масштаб, параметры и область просмотра (подробнее о нем - в нижеследующих разделах). После выбора данного пункта и нажатия клавиши "курсор вправо"происходит возвращение в набираемую задачу, где прорисовывается строка "Построить график:".

8) "Игровые задачи". Пункт является подразделом оглавления, который на текущий момент используется для обучения решателя игре в шахматы. Предусмотрена ини-

циализация шахматной партии либо создание заданной позиции для ее анализа решателем. Обучение находится на начальном этапе, и использовать эту возможность интерфейса рекомендуется лишь тем, кто захотел бы самостоятельно продолжить развитие решателя.

Подраздел "Преобразовать выражение"оглавления типов целевых установок содержит следующие пункты:

1) "Упростить выражение в области допустимых значений". Этот пункт позволяет вводить задачу на преобразование с целями "упростить","одз". Такой набор целей - стандартный для большинства задач на преобразование, включая вычисление пределов, производных, интегралов и определителей матриц.

2) "Вычисление значения константного выражения". Этот пункт является подразделом оглавления, в котором собраны различные целевые установки задач на нахождение точного либо приближенного численного значения константного выражения (см.ниже).

3) "Раскрыть скобки". Создается задача на преобразование, ориентированная на приведение выражения к виду суммы одночленов. Она имеет цели "упростить", "раскрытьскобки", "одз".

4) "Разложить на вещественные множители". Создается задача на преобразование, в которой нужно разложить выражение на вещественные множители. Она имеет цели "упростить", "разложитьнамножители", "одз".

5) "Разложить на комплексные множители". Аналогично предыдущему, но допускается переход к выражениям с мнимой единицей. Задача имеет те же цели, что и выше, к которым добавляется цель "видУмножение". В случае появления комплексных множителей вместо вещественной операции "умножение"возникает комплексная операция "Умножение".

6) "Представить в виде суммы простейших дробей". Вводится задача на преобразование, в которой нужно представить выражение в виде суммы простейших дробей. Предполагается, что это выражение имеет единственную переменную (в действительности процедуры решателя рассчитаны на приведение к виду суммы простейших дробей выражения с несколькими переменными, относительно явно указанной одной из них, что применяется, например, при интегрировании, однако это - "внутренняя" возможность, не вынесенная во внешний интерфейс). Цели задачи - "упростить", "простейщиедроби", "одз".

7) "Разложить в степенной ряд". Создается задача на разложение заданного выражения по заданной переменной x в степенной ряд в окрестности заданной точки t. Здесь имеется в виду получение бесконечной суммы, содержащей все члены ряда. Если в посылках задачи содержится утверждение "комплексное(x)", то предпринимается попытка получить разложение в комплекснозначный ряд Тейлора либо Лорана. После выбора данной целевой установки ("курсор вправо") происходит возвращение в текст набираемой задачи, к которому добавляется строка "Разложить в степенной ряд по переменной". Под этой строкой в левой части экрана размещается курсор формульного редактора, которым нужно ввести переменную разложения x. После ввода этой переменной (ввод завершается нажатием "Enter") к указанной строке добавляется "x в точке", и снова в левой части возникает курсор формульного редактора. Здесь уже нужно ввести выражение t, определяющее точку разложения. По окончании ввода (нажатие "Enter") t перерисовывается в конце строки, задающей

целевую установку, и эта установка завершается двоеточием - далее можно вводить выражение, которое следует разложить в ряд. Такого рода диалоги используются в различных описываемых далее целевых установках. Цели задачи на разложение в степенной ряд - "упростить", "рядтейлора x t", "одз".

8) "Получить заданное число членов разложения в ряд Тейлора". Аналогично предыдущему, но требуется получить заданное конечное число n членов ряда. Диалог ввода целевой установки происходит по тому же сценарию, что и в предыдущем пункте. Однако, после ввода выражения t для точки разложения, к строке текста целевой установки добавляются слова "до членов степени", и далее вводится формульным редактором константа n. Цели получаемой задачи - "упростить", "формулатейлора x t n", "одз".

9) "Разложить в ряд Фурье". Вводится задача на разложение заданного выражения по заданной переменной x в ряд Фурье на отрезке $[a, b]$. Вначале диалога ввода целевой установки прорисовывается строка "Разложить в ряд Фурье по переменной", после чего формульным редактором вводится переменная разложения x. Далее (после x) к строке добавляются слова "на отрезке", и формульным редактором вводится выражение $[a, b]$ (для ввода числового промежутка используются клавиши "Ctr-квадратная скобка"). Цели получаемой задачи - "упростить", "рядфурье x a b", "одз".

10) "Получить явное описание класса". Это - целевая установка задач, в которых множество, изначально заданное с помощью описателя "класс", нужно переформулировать в явном виде, не использующем описателей "класс"и "отображение". К числу таких задач относятся, например, задачи на определение геометрического места точек. Возникающая здесь целевая установка состоит из целей "упростить", "класс", "одз".

11) "Получить асимптотическую оценку". Это - целевая установка задачи на преобразование, в которой требуется получить асимптотическую оценку исходного выражения для предельного поведения переменных, заданного посылками "стремится(...)". Установка состоит из целей "асимптоценка", "упростить", "одз".

Подраздел "Вычисление значения константного выражения" раздела "Преобразовать выражение" содержит следующие пункты:

1) "Найти точное целочисленное значение". В этом случае условие задачи на преобразование построено при помощи целочисленных операций (сложение, вычитание, умножение, степень с натуральным показателем, факториал, модуль и т.п.) из целочисленных констант, и требуется найти его точное численное значение. Число десятичных знаков ограничивается лишь возможностями интерпретатора (свыше 600) и ограничителями в приемах, которые легко ослабить либо вовсе убрать. Задача имеет цели "упростить"и "число".

2) "Вычислить с заданной точностью". Здесь определяется приближенное значение константного выражения, в котором гарантированы первые n знаков после запятой. После выбора данной целевой установки происходит возвращение в набор текста задачи, где прорисовывается строка "Вычислить с точностью до", и под ней - курсор формульного редактора. После набора формульным редактором числа n нажимается "Enter"; тогда n переносится в конец указанной выше строки, и после него добавляется "знаков после запятой:". Задача имеет цели "упростить", "числоценка n".

3) "Найти приближенное значение". Здесь определяется приближенное значение константного выражения без каких-либо гарантий относительно числа точных знаков. Для вычислений используется математический сопроцессор; числа представляются в формате с плавающей запятой и с двойной точностью. Задача имеет цели "числзначение"и "выч".

Подраздел "Найти значения неизвестных"содержит следующие пункты (все они относятся к задачам на описание):

1) "Получить полное явное описание значений неизвестных". Это - целевая установка для обычных задач "с неизвестными", например, для решения систем уравнений и неравенств из элементарной алгебры. Для геометрических задач на вычисление и задач на решение дифференциальных уравнений используются другие целевые установки (см.ниже). В целевую установку входят цели "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные $x_1 \dots x_n$". После выбора целевой установки на экране прорисовывается строка "Найти", под которой формульным редактором вводятся неизвестные задачи - переменные, отделенные друг от друга запятыми. Затем нажимается "Enter".

2) "Найти пример значений неизвестных". Это - задачи на нахождение конкретного примера значений неизвестных, при которых истинны условия. Они имеют цели "полный", "пример", "прямойответ", "одз", "упростить", "неизвестные $x_1 \dots x_n$". После выбора целевой установки на экране прорисовывается строка "Найти пример для". Далее формульным редактором перечисляются неизвестные и нажимается "Enter".

3) "Выразить значения неизвестных через заданные параметры". Это - задачи на вычисление, в которых требуется выразить значения неизвестных (уже явно выраженных через некоторые вспомогательные переменные посылок) через заданные известные величины. Такие задачи на описание - наиболее часто встречающиеся в элементарной и аналитической геометрии, в физике и т.п. Они имеют целевую установку "полный", "явное", "прямойответ", "одз". "упростить", "известно $a_1 \dots a_m$", "неизвестные $x_1 \dots x_n$". Здесь $x_1, \dots, x_n$ - неизвестные, $a_1, \dots, a_m$ - известные параметры. Возможен случай $m = 0$ - если искомые значения неизвестных суть константы; в этом случае соответствующая цель состоит из логического символа "известно". После выбора целевой установки на экране прорисовывается строка "Выразить", и под ней формульным редактором набираются неизвестные (через запятую). После этого неизвестные автоматически переносятся в конец указанной строки, к ней добавляется слово "через", и формульным редактором под строкой перечисляются известные параметры. После нажатия "Enter"(которое происходит сразу же в случае $m = 0$) известные параметры перерисовываются в конце строки. Если этих параметров не было, то вся строка перерисовывается виде "Найти значения $x_1, \dots, x_n$.

4) "Выразить значения неизвестных через известные параметры, предпочтительно в виде парамерического описания". Это - редко встречающаяся разновидность предыдущей целевой установки. Она возникла в аналитической геометрии, для задач на нахождение уравнений линий в параметрическом виде. Диалог по вводу установки такой же, как в предыдущем пункте. В конце диалога к строке целевой установки добавляются слова "Предпочтительно параметрическое описание". К списку целей (по сравнению с предыдущим пунктом) добавляется "вспомпараметр".

5) "Решить функциональные уравнения". Целевая установка задач на решение дифференциальных уравнений и других задач, в которых используется аналогичная кон-

текстная семантика (по умолчанию навешивается квантор общности по значениям варьируемых переменных). Здесь варьируемые переменные $x_1, \ldots, x_m$, от которых зависят неизвестные функции $y_1, \ldots, y_n$, указываются в цели "связка $x_1 \ldots x_m$". Остальные цели - "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные $y_1 \ldots y_n$". В процессе диалога прорисовывается строка "Найти", после которой формульным редактором должны быть набраны выражения $y_1(x_1, \ldots x_m), \ldots y_n(x_1, \ldots, x_m)$.

6) "Найти значения неизвестных, для которых существуют заданные объекты". Это - обычные задачи "с неизвестными"(см. пункт 1), у которых для части неизвестных не требуется найти их явное значение, а достаточно установить существование такого значения. Такие "несущественные"неизвестные, как правило, в ответ задачи вообще не входят. Цели задачи суть "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные $x_1 \ldots x_n$", "параметры $y_1 \ldots y_m$". Здесь $y_1, \ldots, y_m$ - те из неизвестных $x_1, \ldots, x_n$, для которых не требуется искать явное значение. При вводе целевой установки используется следующий диалог. Сначала прорисовывается строка "Найти"; затем формульным редактором вводятся неизвестные $x_1, \ldots, x_n$; далее к строке добавляются слова "для которых существуют", и формульным редактором вводятся несущественные неизвестные $y_1, \ldots, y_m$.

7) "Вычислить значения неизвестных при заданных значениях параметров". Целевая установка используется для задач на описание, ответом которых служит программа вычислений, определяющая численные значения неизвестных по заданным численным значениям параметров. Такая программа создается в два этапа. Сначала решается обычная задача на описание, преобразующая исходную систему утверждений, связывающих значения параметров и неизвестных, в систему утверждений, определяющих алгоритм вычислений. Обычно здесь применяются стандартные вычислительные формулы (метод трапеций, метод Рунге-Кутта, метод Ньютона и т.п.). Могут использоваться возможности решателя, связанные с задачами по физике, разумеется, достаточно скромные для текущего этапа его обучения. Полученные утверждения передаются компилятору ГЕНОЛОГа, который создает по ним ЛОС-программу для вычислений. Если задача не имела параметров, то полученная программа сразу же запускается, и выдается найденный ответ. В противном случае появляется интерфейс для задания параметров. В верхней части экрана прорисовываются сначала параметры, под ними - неизвестные. Для ввода значений параметров нажимается "в", и далее - через запятую перечисляются равенства, фиксирующие численные значения параметров. Точка в конце не ставится; по окончании ввода нажимается Enter. Этот интерфейс можно использовать многократно, не производя повторного синтеза программы. Если задача уже была решена ранее, то можно сразу входить в интерфейс вычислений по ее программе, нажимая вместо "о" (кир.) клавишу "Ctr - о".

Целевая установка состоит в рассматриваемом случае из целей "полный", "явное", "прямойответ", "одз", "упростить", "вычисление", "неизвестные ...", "параметры ...".

На текущий момент проработано несколько типов вычислительных задач, что позволяет использовать решатель, например, для вычисления определенных интегралов и решения систем уравнений - обычных или дифференциальных. Однако, в целом работа по вычислительным задачам в логической системе находится на самом начальном этапе. Основной ее целью является такое развитие компилятора ГЕНОЛОГа, которое

позволило бы создавать достаточно эффективные вычислительные программы непосредственно с уровня теорем, как это имеет место для логических приемов. Данная постановка проблемы представляется совершенно естественной, так как подлинным источником любых вычислительных процедур в математике служат теоретические знания (в обобщенном смысле - теоремы). Подключение к компиляции предварительных логических процедур, работающих с такими знаниями - в виде решения задач либо логического вывода теорем - существенно увеличит возможности системы по бытрому созданию вычислительных блоков и объединению их в более сложные вычислительные агрегаты. Следует выделить здесь также интересную перспективу освоения смешанных логико-вычислительных режимов при автономном исследовании решателем свойств различных математических моделей.

8) "Получить явное описание значений неизвестных, не обязательно полное". Установка применяется в задачах на описание, для которых требуется получить не полный, а лишь возможно более полный ответ. Она состоит из целей "явное", "прямойответ", "одз", "упростить", "неизвестные ...". Была использована пока в единичных случаях, и лишь совсем немногие приемы ее учитывают. По существу, представляет собой заготовку для последующего развития системы.

9) "Получить полное явное описание значений неизвестных, используя приближенные вычисления". Как и в предыдущем случае, это заготовка для последующего развития системы. Пока использована лишь для указания на режим использования приближенных вычислений в машинном формате "с плавающей запятой" при численном решении систем линейных уравнений. Процедура, выполняющая последнее, совершенно стандартна, и представляет интерес лишь как пример создания компилятором ГЕНОЛОГа обычной вычислительной программы с того же "теоремного" уровня, с которого создаются логические приемы.

Подраздел "Исследовать свойства объекта"содержит следующие пункты:

1) "Исследовать поведение функции". Это - задачи на качественное исследование графика функции одной вещественной переменной (область определения, промежутки монотонности, экстремумы, число корней на промежутках). Они имеют цели "полный", "явное", "прямойответ", "одз", "исследовать", "функция", "неизвестные x", где x - исследуемая функция. Условие такой задачи сводится к равенству $x = \lambda_y(f(y), y{-}\text{число})$, определяющему функцию. Диалог ввода целевой установки прост: сначала прорисовывается строка "Исследовать поведение функции", затем вводится переменная x.

2) "Исследовать поведение функции с учетом выпуклости и вогнутости". Этот пункт аналогичен предыдущему; при исследовании предпринимается дополнительно попытка найти промежутки выпуклости и вогнутости. К списку целей предыдущего пункта добавляется цель "выпуклавверх".

3) "Исследовать функцию на монотонность". Этот пункт аналогичен пункту 1, но предпринимается исследование только интервалов монотонности. К списку целей пункта 1 добавляется цель "монотонно".

4) "Исследовать функцию на четность и периодичность". Этот пункт аналогичен пункту 1, но предпринимается исследование только четности и периодичности (в случае 1 такое исследование не выполнялось). К списку целей пункта 1 добавляется цель "четнаяфункция".

5) "Исследовать функцию на непрерывность". Этот пункт аналогичен пункту 1, но исследуется только непрерывность: находятся интервалы непрерывности, точки разрыва, и определяются типы этих точек. К списку целей пункта 1 добавляется цель "непрерывно".

6) "Исследовать функцию на равномерную непрерывность". Этот пункт аналогичен пункту 1, но исследуется только равномерная непрерывность (без исследования точек разрыва). К списку целей пункта 1 добавляются цели "непрерывно"и "равномернонепрерывно".

7) "Исследовать свойства линии, заданной своим уравнением". Здесь выполняется определение типа кривой второго порядка, заданной своим уравнением, и предпринимается стандартная характеризация этой кривой (включая нахождение канонической системы координта и канонического уравнения). Цели задачи суть: "полный", "явное", "прямойответ", "одз", "исследовать", "линия", "неизвестные x", где x - исследуемая кривая. Условие задачи имеет вид "коорд$(x, K) = set_{xy}(x -$ число $\&$ $y -$ число $\&$ $f(x, y) = 0)$". В действительности элементы $x-$число,$y-$число этого условия можно не вводить - они автоматически добавляются решателем, если соответствующая переменная явно встречается в $f(x, y)$ под какой-либо арифметической операцией. В посылках задачи должно иметься утверждение "прямкоорд(K)". Диалог создания установки аналогичен пункту 1: после прорисовки строки "Исследовать кривую"вводится неизвестная x, и далее к строке добавляются слова "заданную своим уравнением".

8) "Исследовать свойства поверхности, заданной своим уравнением". Аналогично предыдущему пункту, но для поверхности второго порядка. Вместо цели "линия" берется цель "эллипсоид".

9) "Определить вид множества точек, заданного условием на координаты точек". Целевая установка на преобразование описания множества точек к бескоординатной форме. Состоит из целей "полный", "явное", "прямойответ", "одз", "упростить", "исследовать", "точки", "неизвестные ...". Была использована в задачах на качественную характеризацию вида подмножества точек комплексной плоскости, удовлетворяющих заданным соотношениям.

10) "Исследовать функцию комплексного переменного на особые точки". Целевая установка на нахождение и характеризацию особых точек заданной функции комплексного переменного. Состоит из целей "полный", "явное", "прямойответ", "одз", "исследовать", "особыеточки", "функция", "неизвестные ...". Как данная, так и предыдущая установка связаны с разделами, для которых обучение решателя лишь начато. Их могут использовать те, кто захотел бы научиться создавать свои приемы для новых задач - по аналогии с теми, которые уже имеются в решателе и обеспечивают решение аналогичных задач его задачника.

Кроме режима ввода целевой установки с помощью оглавления типов целей, можно набирать эту установку непосредственно текстовым редактором. Для входа в режи набора установки текстовым редактором служит клавиша "Ctr-ц". Сначала набирается тип задачи ("преобразовать","описать") и далее перечисляются в скобочной записи цели. Между целями оставляются пробелы; запятые не используются. Если цель имела вид набора $(fa_1 \dots a_n)$, то она вводится как $f(a_1 \dots a_n)$.

Чтобы изменить ранее введенную целевую установку, используются те же клавиши "ц"и "Ctr-ц", что и для ее изначального создания. При этом следует помнить, что

если задача, для которой изменяется целевая установка, не является последней в текущем списке задач, то предварительно ее следует выделить.

Можно просматривать оглавление типов целевых установок независимо от процесса ввода либо изменения задачи. Для входа в это оглавления достаточно нажать (находясь в просмотре списка задач) клавишу "Ц". Обычно данная возможность применяется для редактирования оглавления типов целевых установок.

После ввода целевой установки задачи выполняется ввод ее условия либо (в случае задачи на описание) нескольких условий. На этом процесс создания новой задачи завершается.

Если нужно сохранить текстовую версию формулировки задачи, нажимается клавиша "н", и далее эта версия вводится с помощью текст-формульного редактора.

3.4 Геометрический редактор

Кроме формульного редактора, при вводе условий задач часто бывает полезен также геометрический редактор. Он позволяет создавать простые чертежи, состоящие из некоторой системы обозначенных буквами точек, между которыми проведены линии (отрезки прямых либо окружности). Чертежи используются в задачах по геометрии и в описании геометрических приемов. Какой-либо функциональной нагрузки в работе решателя они пока не имеют, а используются лишь для обеспечения необходимой наглядности. Тем не менее, при выполнении по ходу решения задачи дополнительных построений решатель сам пополняет чертеж новыми точками и линиями. Все, что нужно для этого сделать - ввести в описании задачи чертеж, обозначения точек на котором согласованы с используемыми в условии задачи.

Если вводится новая геометрическая задача, для которой нужен чертеж, то лучше всего начинать ее набор прямо с чертежа. Для этого достаточно ввести бланк задачи (прорисована верхняя горизонтальная линия, обозначающая начало задачи) и нажать клавишу "ч". Если уже была набрана часть текста задачи, то вход в геометрический редактор осуществляется тем же самым нажатием клавиши "ч". В обоих случаях вся отведенная для задачи часть экрана занимается прямоугольником геометрического редактора, а набранная ранее часть текста задачи пропадает с экрана. Она будет восстановлена по окончании (либо отмене) создания чертежа и размещена под чертежом. Можно попробовать создать чертеж с помощью автоматической процедуры, имеющейся в решателе. Для этого нажимается клавиша "Ч". Если до этого была введена ручная версия чертежа, то она будет заменена автоматической; восстановление предыдущей версии чертежа при этом достигается путем выделения чертежа (левой клавишей мыши) и нажатия "Ctr-Del". Если нужно вручную изменить ранее введенный чертеж задачи (в том числе, автоматически введенный), то, как и при первоначальном вводе чертежа, нажимается "ч".

Заметим, что по окончании редактирования чертежа либо при отмене начатого редактирования автоматически выполняется прокрутка списка задач, так что в верхней части экрана оказывается начало обрабатываемой задачи. Указанным выше способом можно входить в редактирование чертежа только для последней задачи просматриваемого списка задач. Если же задача не последняя, то ее следует сначала выделить (нажатие правой клавиши мыши в любой точке поля задачи), и лишь затем нажать "ч".

В процессе редактирования чертежа можно получить информацию о действиях геометрического редактора, нажав клавишу F1.

Окно геометрического редактора представляет собой прямоугольник, ограниченный линией синего цвета. В правом верхнем углу отделена небольшая область, в которой появляется указатель текущего режима редактирования. Под ней размещены четыре логических символа, образующих меню геометрического редактора. Предусмотрены следующие режимы работы:

1. Нейтральный режим. Этот режим устанавливается при входе в геометрический редактор. У него прямоугольник указателя режима - пустой. Переход из любого другого режим в этот режим обеспечивается нажатием клавиши пробела. В нейтральном режиме можно выполнять следующие операции:

а) Перемещение точки либо сопровождающего ее буквенного обозначения. Для этого точка выделяется нажатием левой клавиши мыши после подведения к ней курсора мыши. Затем можно сдвигать точку нажатием клавиш курсора (постоянное удержание клавиши нажатой приводит к медленному движению точки в выбранном направлении). Аналогично, можно сдвигать обозначение выделенной точки нажатием клавиш "Ctr-курсоры". При движении точки корректируются все ведущие к ней отрезки прямых.

б) Изменение обозначения точки. Сначала нажимается буквенная клавиша (быть может, сопровождаемая несколькими цифровыми при указании индекса) для нового обозначения, затем выделяется нужная точка (левая клавиша мыши) и нажимается -". Как и в случае добавления новой точки (см. ниже), для ввода большой буквы на клавиатуре нажимается малая буква, и наоборот. Старое обозначение заносится в накопитель обозначений точек, и будет использовано при последующих вводе либо коррекции обозначения (если не сбросить накопитель с помощью клавиши "Delete").

Эти операции можно применять и в других режимах (но не в режиме ввода новых точек, так как тогда каждая попытка выделить точку будет приводить к созданию новой точки).

2. Режим ввода новых точек. Вход в режим ввода новых точек осуществляется при нажатии клавиши "Insert"либо выборе мышью пункта меню "точка". В прямоугольнике указателя режима появляется слово "точка". Для ввода новой точки следует выбрать курсором мыши ее позицию и нажать левую клавишу мыши. В качестве обозначения точки каждый раз выбирается первая не использованная большая буква. Однако, можно явно указывать обозначения точек, предварительно нажав последовательно некоторые буквенные клавиши. Тогда последующие нажатия левой клавиши мыши на выбранных позициях приведут к использованию для точек обозначений из данной последовательности. Так как обычно точки обозначаются большими буквами, предусмотрено автоматическое преобразование вводимых с клавиатуры малых букв в большие, а больших - в малые. Для сброса накопителя обозначения новых точек нажимается клавиша "Delete".

Обозначения точек можно сопровождать индексами - для этого после нажатия буквенной клавиши нажимаются одна или несколько цифровых клавиш, образующих индекс.

Если нужно удалить ранее введенную точку, то к ней подводится курсор мыши и нажимается правая клавиша мыши. При этом обозначение удаленной точки сохраняется в накопителе обозначений и будет использовано (если не сбросить этот накопитель) при вводе новой точки.

3. Режим сдвига точек. Переход в этот режим осуществляется нажатием клавиши "Ctr-c". Указатель режима прорисовывает слово "сдвиг". Далее можно изменить положение любой из точек, не прибегая к сравнительно медленному перемещению ее клавишами курсора. Для этого сначала точка выделяется мышью (нажатие левой клавиши после подведения к точке курсора мыши), затем курсор мыши подводится к новой позиции точки, и снова нажимается левая клавиша мыши.

4. Режим проведения отрезка. Переход в режим осуществляется при нажатии "Ctr-o"("о кир.) либо выборе мышью пункта меню "отрезок". Указатель режима прорисовывает слово "отрезок". Для проведения отрезка между двумя точками сначала выделяется первая из них, затем курсор мыши подводится ко второй и нажимается левая клавиша мыши. Если нужно не ввести, а наоборот, удалить ранее введенный отрезок, то после выделения первой точки отрезка на второй точке нажимается правая клавиша мыши.

5. Режим проведения перпендикуляра. Переход в режим осуществляется при нажатии клавиши "Ctr-т". Указатель режима прорисовывает слово "перпендикуляр". Возможны две операции: восставить перпендикуляр к прямой в заданной точке этой прямой, либо опустить из заданной точки перпендикуляр на заданную прямую.

В первом случае левой клавишей мыши выбираются две различные точки на прямой (вторая из них - та, через которую пройдет перпендикуляр). Чтобы доопределить направление перпендикуляра и ввести на нем еще одну точку, курсором мыши выбирается какая-либо из уже введенных ранее точек, и на ней нажимается правая клавиша мыши. Тогда перпендикуляр к прямой, проходящей через первые две точки, будет проведен из второй точки в направлении третьей точки, причем на нем будет введена ближайшая к третьей точке новая точка.

Во втором случае, как и в первом, сначала левой клавишей мыши выбираются две точки на прямой, к которой требуется провести перпендикуляр, и затем на третьей точке - из которой должен быть опущен перпендикуляр - нажимается левая клавиша мыши. При проведении перпендикуляра вводится новая точка - основание этого перпендикуляра.

6. Режим проведения окружности. Переход в режим осуществляется при нажатии клавиши "Ctr-к"("к кир.) либо выборе мышью пункта меню "окружность". Указатель режима прорисовывает слово "окружность". Для проведения новой окружности сначала левой клавишей мыши выбирается ее центр, затем берется какя-либо точка, через которую должна пройти окружность, и снова нажимается левая клавиша мыши. Чтобы удалить введенную ранее окружность, сначала левой клавишей мыши выбирается ее центр, затем на той точке окружности, с помощью которой она вводилась, нажимается правая клавиша мыши. Последняя операция возможна лишь в том случае, если ранее введенная окружность не перемещалась по экрану (такое перемещение происходит при перемещении ее центра); иначе для удаления окружности нужно удалить ее центр).

7. Режим параллельного перемещения всего изображения. Переход в режим осуществляется при нажатии клавиши "Home"либо выборе мышью пункта меню "перемещение". Указатель режима прорисовывает слово "перемещение". Для задания вектора перемещения сначала на некоторой позиции нажимается клавиша мыши, затем курсор мыши переносится на новую позицию, и снова нажимается клавиша мыши.

8. Режим изменения размеров рамки чертежа. Для выбора того края рамки, который требуется передвинуть, нажимается "правый Ctr - клавиша курсора, соответствую-

щая направлению к нужному краю". Новое положение края выбирается нажатием левой кнопки мыши после перемещения ее курсора на нужную позицию. Режим обычно используется для увеличения зоны чертежа, под которым размещены другие изображения. Это увеличение относится только к периоду редактирования; по завершении работы геометрического редактора нижняя граница чертежа выбирается автоматически по крайним снизу его точкам.

3.5 Текстформульный редактор

Для создания и просмотра форматированных текстов с вставленными в них формулами и чертежами применяется текстформульный редактор. В системе с его помощью реализованы общий справочник, справочные сведения для отдельных логических символов и (частично) интерфейс просмотра задач, в том числе интерфейс редактирования задач на текстовый анализ. Текст составляется из элементов, создаваемых текстовым, формульным и геометрическим редакторами. Между этими элементами могут вставляться указатели абзаца, пустой строки и пропуска заданного числа позиций (без перехода на новую строку).

Дадим краткое описание интерфейса текстформульного редактора. Завершение редактирования происходит единственным способом - по нажатии клавиши "End"; в этом случае результаты редактирования будут сохранены. Для отмены редактирования можно использовать клавишу "курсор влево"либо "Esc естественно, без сохранения результатов редактирования. При просмотре текстформульного массива можно применять обычную прокрутку с помощью клавиш "курсор вверх", "курсор вниз"и клавиш "PageUp", "PageDown".

В различных операциях используется выделение элемента текстформульного массива. Такое выделение осуществляется мышью: ее курсор подводится в поле рассматриваемого элемента, и нажимается левая клавиша мыши. Выделенный элемент перекрашивается в синий цвет. Повторное нажатие левой клавиши мыши отменяет выделение элемента. При выделении нового элемента старое выделение автоматически сбрасывается. Выделение элементов требует осторожности, так как оно подготавливает выполнение различных операций, и случайное нажатие клавиши может привести к нежелательным последствиям.

Для ввода нового текста, формулы либо чертежа используются, соответственно, уже описанные ранее текстовый, формульный либо геометрический редакторы. Если добавление нового элемента происходит в конце текстформульного массива, то нужно с помощью прокрутки освободить в нижней части экрана достаточное место и нажать одну из следующих клавиш:

а) Ввод нового текста - либо нажать "Enter", либо нажать "т".

б) Ввод новой формулы формульным редактором - нажать "ф" маленькое.

в) Ввод новой формулы текстовым редактором - нажать "Ф" большое.

г) Ввод чертежа - нажать "ч".

Это обеспечивает вход в необходимый редактор; по завершении редактирования элемент добавляется непосредственно после предыдущего (если оставалось место в строке, занятой предыдущим элементом, то - начиная с первой свободной позиции этой

строки; исключение составляет лишь чертеж, который всегда помещается начиная с новой строки). Возможна отмена редактирования (Esc).

Если нужно вставить новый элемент перед ранее созданным элементом, то этот элемент сначала выделяется, и затем выполняются указанные выше действия а) - г).

Если нужно, чтобы новый элемент начинался с новой строки, то перед его вводом нажимается клавиша "а"(абзац). Если элемент уже был введен, и требуется сделать его начинающимся с новой строки, то этот элемент выделяется левой клавишей мыши и нажимается "а". Аналогичным образом вводится пропуск строки - перед набором нового элемента нажимается "с"либо после выделения ранее набранного элемента нажимается "с".

Для вставки пробелов перед выделенным элементом следует выбрать пункт меню "Пробелы"либо нажать клавишу пробела, после чего нажать левую клавишу мыши на той позиции, до которой следует сдвинуть начало выделенного элемента. Сдвиг осуществляется только в рамках одной строки, без перехода на следующую строку.

Для удаление введенных перед выделенным элементом пробела, абзаца либо пропуска строки следует нажать клавишу "Ctr-п".

Для изменения текста, формулы либо чертежа предпринимается нажатие правой клавиши мыши в области данного элемента. Если нужно изменить формулу формульным редактором, то предварительно эта формула выделяется. Иначе ее изменение будет выполняться текстовым редактором.

Для удаления элемента он сначала выделяется, а затем нажимается "Ctr-Del". Пробелы, абзацы и пропуски перед удаленным элементом сохраняются. Поэтому перед удалением концевого элемента целесообразно сначала удалить предшествующие ему пробелы и пропуски.

Для перестановки выделенного элемента на новую позицию курсор мыши подводится в область того элемента, перед которым требуется расположить выделенный элемент. Затем нажимается правая клавиша мыши. Если правая клавиша нажимается вне областей элементов массива, то выделенный элемент переносится в конец массива.

Глава 4

Общая схема функционирования решателя

4.1 Сканирование задачи

При обычном программировании для решения задач заданного типа разрабатывается один общий алгоритм, который гарантирует получение за конечное число шагов ответа в случае любой конкретной задачи этого типа. Такой алгоритм представляется в виде нескольких взаимодействующих между собой блоков, обеспечивающих циклический процесс постепенного упрощения параметров решаемой задачи (в частности, уменьшения оставшегося объема перебора для процедур переборного типа) - вплоть до получения окончательного ответа. Каждый из блоков несет здесь вполне определенную функциональную нагрузку, как правило, основанную на том или ином теоретическом утверждении, связанном с обрабатываемыми алгоритмом объектами. Его можно рассматривать как прием для достижения определенной текущей подцели. В свою очередь, отдельный блок сам определяется схемой своих подблоков, и т.д. Здесь возникает иерархия уровней, в которой на каждом уровне несколько приемов (как правило, не более одного - двух десятков) оказываются связаны в вычислительный цикл, так что после выполнения очередного приема известно, к какому приему следует переходить дальше.

К сожалению, такого рода четкую программу действий удается составить лишь для достаточно узких классов задач. Хорошо известны результаты об алгоритмической неразрешимости различных общих проблем, и даже в случае наличия алгоритма он часто оказывается практически неприемлемым по своей трудоемкости. Для преодоления возникающих здесь трудностей приходится резко увеличивать количество используемых приемов, чтобы охватить алгоритмическими возможностями если не все многообразие задач предметной области, то хотя бы возможно больший класс задач, встречающихся в реальных ситуациях. Эти приемы создаются уже не путем теоретического проектирования алгоритма, а извлекаются из последовательности конкретных обучающих задач и складываются в одну общую базу приемов. Количество приемов, используемых для решения задач в предметной области, обычно существенно больше чем в алгоритмической процедуре указанного выше блочного типа - сотни и тысячи вместо десятков. Последовательность их работы заранее не фиксирована, и после срабатывания очередного приема необходим цикл поиска следующего применяемого приема. Разумеется, при увеличении числа приемов, работающих в общем процессе, существенно усложняется регулировка взаимодействия между ними,

и возникает необходимость в длительной эмпирической оптимизации их решающих правил.

Для организации цикла поиска очередного применяемого приема в решателе используется процедура сканирования задачи, которую можно представлять как своего рода модель внутреннего логического зрения. Для большинства приемов активизация их при рассмотрении задачи начинается с обнаружения в логических структурах данных некоторого логического символа, заранее выбранного для этого приема. В качестве такого ключевого логического символа берется обычно некоторый логический символ, появление которого необходимо для возможности применения рассматриваемого приема, причем при нескольких возможных выборах предпочтение отдается наиболее редко встречающемуся логическому символу (для уменьшения потерь времени при попытках применения приема). Указанное закрепление за приемами логических символов предопределяет организацию всей базы приемов решателя по принципу энциклопедии: она распадается на группы приемов, "принадлежащих" соответствующим символам, причем каждая группа реализуется в виде отдельной алгоритмической процедуры A_f - программы логического символа f. В особых случаях прием не удается связать с каким-либо конкретным логическим символом, появление которого является необходимым для срабатывания. Такие приемы (их совсем немного) распределены по четырем логическим символам - названиям "доказать", "описать", "преобразовать", "исследовать" типов задач, при решении которых возможно их применение.

Текущая ситуация, возникающая в процессе работы решателя, описывается последовательностью задач $Z_1, Z_2, \ldots, Z_N$, где Z_{i+1} - вспомогательная задача, введенная при решении задачи Z_i $(i = 1, \ldots, N-1)$. В этой последовательности (далее называем ее цепью задач) задача Z_1 является фиктивной - она создается автоматически при запуске логической системы и используется для организации интерфейса. Эта задача (далее называемая исходной задачей) имеет тип "исследовать"и единственную однобуквенную посылку "вход". При просмотре посылок данной задачи решатель обращается к программе логического символа "вход", которая и является программой интерфейса логической системы. С помощью интерфейса можно ввести некоторую задачу Z_2, которая далее и решается, порождая вспомогательные задачи $Z_3, \ldots, Z_N$. Будем называть задачу Z_N текущей задачей; Z_2 - корневой задачей. Исходная задача, кроме организации интерфейса, выполняет функции "доски объявлений" - различные процедуры решателя могут обмениваться между собой сообщениями, размещая их в списке комментариев к посылкам этой задачи.

Каждая задача Z_i $(i = 1, \ldots, N)$ характеризуется натуральным числом M_i, называемым ее максимальным уровнем и определяющим уровень средств, отведенных для ее решения (по исчерпании этих средств выдается отказ), целым неотрицательным числом m_i, $m_i \leq M_i$, называемым текущим уровнем этой задачи и определяющим уровень средств, среди которых в текущий момент ведется поиск очередного преобразования задачи Z_i, а также вспомогательной информацией, необходимой для возобновления прерванной процедуры решения задачи Z_{i-1} по окончании решения задачи Z_i.

Текущий уровень задачи является одним из входных параметров, получаемых приемами; он учитывается решающими правилами приемов и позволяет организовать необходимые приоритеты в их применении: при меньших значениях этого уровня срабатывают приемы с большим приоритетом. В процессе обучения решающие правила приемов корректируются таким образом, чтобы на каждом шаге выбирался

прием, наиболее целесообразный с точки зрения обучающего систему эксперта. При первоначальном обращении к задаче Z_i ее текущий уровень равен 0.

Изменение текущей ситуации происходит в следующем рабочем цикле решателя:

1) Происходит обращение к программе логического символа f - типа задачи Z_N. Если при этом не срабатывает ни один из приемов, либо сработавшие приемы изменили лишь комментарии задач и ни один из них не указал явно на необходимость повторного рассмотрения задачи (в таких случаях говорим, что процедура не внесла существенных изменений в текущую ситуацию), то переход к пункту 3, иначе - к пункту 2.

2) Если в результате срабатывания приема определился ответ на задачу Z_N либо был выдан отказ на нее, то возобновляется прерванный ранее прием решения задачи Z_{N-1}, в процессе реализации которого возникла задача Z_N. При $N = 2$ в этом случае выдается ответ либо отказ на решаемую задачу и возвращение в программу интерфейса решателя; при $N = 1$ - происходит выход из логической системы. При отсутствии ответа либо отказа на задачу Z_N текущий уровень этой задачи заменяется на 0, и переход к пункту 1. Это означает, что при наличии существенных изменений, внесенных приемом, решатель повторяет цикл анализа текущей ситуации с самого начала.

3) Осуществляется последовательный просмотр всех условий и посылок F задачи Z_N, вес v которых равен текущему уровню m_N этой задачи либо равен $m_N + 1$. Сначала просматриваются условия, затем посылки; порядок просмотра условий (посылок) - слева направо по соответствующим спискам задачи Z_N. Если $v = m_N$, то происходит однократный просмотр всех вхождений логических символов φ в F (слева направо); если же $v = m_N + 1$, то - серия таких просмотров, в процессе которых значение текущего уровня задачи Z_N полагается последовательно равным $0, 1, \ldots, m_N$. Для каждого рассматриваемого вхождения логического символа φ в F осуществляется обращение к программе логического символа φ (исходные данные этой программы содержат полную информацию о координатах вхождения символа φ в задачу Z_N). Если процедура не внесла существенных изменений в текущую ситуацию, то переход к рассмотрению очередного вхождения логического символа, иначе - к пункту 2. Если просмотр терма F закончился безрезультатно, то вес его увеличивается на 1 (новые либо видоизмененные посылки и условия задачи получают вес 0). Если просмотр всех условий и посылок задачи Z_N закончился безрезультатно, то текущий уровень задачи Z_N увеличивается на 1. Если в результате он становится больше, чем максимальный уровень M_N, то на задачу Z_N выдается отказ, иначе - переход к пункту 1.

Использование весов посылок и условий позволяет сузить область просмотра при поиске очередного приема, исключая из нее посылки и условия, имеющие большой вес (они уже были достаточно хорошо рассмотрены ранее, и срабатывание связанного с ними приема маловероятно). В результате происходит локализация рассмотрения задачи, направляемого в первую очередь на новые либо видоизмененные посылки и условия; рассмотрение же всей задачи в целом имеет место, как правило, лишь на начальном этапе ее решения. По мере повышения текущего уровня m_N задачи Z_N в просмотр вовлекаются ранее отложенные посылки и условия F, вес v которых больше m_N; это происходит при $v = m_N + 1$ (см. пункт 3), причем предварительно осуществляется поиск приема, срабатывающего при рассмотрении F для меньших, чем m_N, значений текущего уровня. Переключение внимания при рассмотрении задачи может происходить также в результате срабатывания приемов, уменьшающих

веса тех или иных условий и посылок.

4.2 Запуск решения задачи и его пошаговый просмотр

Запуск решения задачи происходит из просмотра списка задач некоторого концевого раздела задачника. Войдя в этот список и создав в нем новую задачу либо выбрав для решения одну из ранее имевшихся задач, следует прежде всего добиться, чтобы верхняя горизонтальная линия данной задачи была прорисована на экране, а верхняя линия предыдущей задачи - не была видна на экране. Альтернативный способ - выделение нужной задачи нажатием правой кнопки мыши на ее поле (чтобы задача оказалась выделена, ее верхняя горизонтальная линия опять же должна быть видна на экране), после чего можно произвольно перемещаться по списку задач и выполнять запуск выделенной задачи из любой его точки.

Если требуется получить ответ задачи без отображения процесса решения, то нажимается клавиша "o" (кир.). Если на задачу будет получен ответ, то он будет прорисован в верхней части экрана, с указанием времени решения (в секундах либо минутах), а также с указанием трудоемкости, измеряемой в числе шагов работы интерпретатора языка ЛОС (этот язык нижнего уровня, используемый для записи приемов, описывается в последующих разделах книги). Ответ и трудоемкость сохраняются в файлах задачника и впоследствии будут прорисовываться непосредственно под условием задачи (время решения не сохраняется). Для исключения их из файлов следует выделить задачу (правой кнопкой мыши) и нажать "Ctr-F4". Если ответ на задачу не получен, то происходит перерисовка ее текста с указанием под ним времени, затраченного на решение.

Если нужно отображать процесс решения по шагам, то нажимается клавиша "p" (кир.). Каждый последующий шаг обеспечивается нажатием "Enter". Если в некоторый момент нужно оборвать пошаговое отображение решения и далее решать задачу до получения ответа, то нажимается клавиша "0"(ноль). Если нужно вообще прервать решени, то нажимается клавиша "Esc", возвращающая к тексту задачи в задачнике.

При показе очередного шага решения в верхней части экрана прорисовывается описание текущего действия. Над этим описанием отображается вся цепь задач (кроме фиктивной исходной задачи)- сначала идет выбранная из задачника задача (возможно, измененная в процессе решения по сравнению с ее первоначальной версией, оставшейся в задачнике), затем вспомогательная задача, к которой произошло обращение от первой задачи, и т.д. Под последней задачей этой цепи задач и размещается описание текущего действия. При просмотре всех этих записей применяется та же прокрутка, что и при просмотре списка задач в задачнике.

Отображаемые на экране задачи приводятся с теми же сокращениями, что и в задачнике (для включения либо выключения полного просмотра нажимается клавиша "ы"). Кроме того, в просматриваемой цепи задач могут встречаться "замаскированные"под задачи обращения к вспомогательным процедурам (пакетным операторам, см. описание языка ГЕНОЛОГ), не являющиеся задачами. Такие вспомогательные операторы введены из соображений оптимизации: каждый из них содержит в

себе сравнительно небольшое число приемов, и поиск нужного приема в нем ускоряется на порядок по сравнению с поиском по всей базе приемов. Типы этих операторов аналогичны типам задач: проверочный оператор является аналогом задачи на доказательство, нормализатор - аналогом задачи на преобразование, синтезатор - аналогом задачи на описание и анализатор - аналогом задачи на исследование.

Под кадром, содержащим описание текущего действия, могут быть размещены несколько кадров со вспомогательными задачами - эти задачи решались в процессе выполнения данного действия. Можно повторно запустить решение любой из них - выделив ее либо разместив ее верхнюю линию в верхней части экрана и нажав "Enter". Эту процедуру можно применять любое число раз и внутри пошагового просмотра решения вспомогательных задач - таким образом создается подобие гипертекста решения. Для возвращения на предыдущий уровень данного гипертекста нажимается клавиша "End".

Комментарии к очередному действию решателя размещаются в скобках после описания этого действия. Вспомогательные задачи, сопровождающие текущее действие, также снабжаются комментарием, размещаемым в скобках перед описанием задачи. Иногда эти комментарии специально подготовлены для объяснения примененного приема, иногда они просто представляют собой подзаголовок того раздела оглавления базы приемов, в котором размещен примененный прием.

Если комментарий недостаточен для понимания выполненного действия или вообще непонятен (такое может случиться, если комментарий просто является подзаголовком концевого пункта оглавления приемов - некоторые из этих подзаголовков понятны только в контексте заголовков внешних разделов), то можно перейти к просмотру описания сработавшего приема. Конечно, здесь понадобится знакомство с языками, на которых задаются приемы решателя - ЛОСом и ГЕНОЛО́Гом (им посвящены последующие разделы книги). Для этого нажимается клавиша "б", переводящая в просмотр приема (если прием реализован на языке ГЕНОЛОГ) либо (если прием реализован на языке ЛОС) переводящая в тот концевой пункт оглавления программ, который связан с последней пройденной перед реализацией приема контрольной точкой. В обоих случаях можно также просмотреть ЛОС-программу примененного приема, нажав клавишу "ф" - она переводит в отладчик ЛОСа. Для возвращения в просмотр текущего действия решателя из просмотра описания приема на ГЕНОЛОГе следует нажать клавишу "End". Для возвращения из просмотра ЛОС-программы приема следует нажать клавишу "з".

При просмотре текущего шага решения можно посмотреть исходную задачу в задачнике, не прерывая процесса решения. Для этого достаточно нажать клавишу "Home"; возвращение к просмотру текущего шага решения из задачника - по нажатии "End".

Если при пошаговом просмотре возник промежуточный результат, представляющий самостоятельный интерес (даже в случае, когда ответ на решаемую задачу так и не будет получен), то можно выделить этот результат (целую формулу или ее часть, одну или несколько) - так же, как это делалось в задачнике, перейти в задачник (через "Home") и зарегистрировать выделенную формулу в любой из задач или введя новый бланк задачи. Как и обычно, для этого следует войти в формульный редактор и нажать "Insert" - "N", где N - номер выделенного элемента среди всех выделенных элементов; $1 \leq N \leq 9$. Далее можно нажать "End"и продолжить просмотр шагов решения.

Если возникла необходимость прервать процесс пошаговой трассировки решения некоторой задачи из цепи задач и перейти к очередному шагу для ее надзадачи, то следует выделить (правой кнопкой мыши) данную надзадачу и нажать "Enter". Тогда следующий отображаемый на экране шаг будет относиться уже к выбранной надзадаче.

Если требуется прервать затянувшийся шаг решения задачи, то нажимается клавиша "Break". В результате появится текст текущего выполняемого фрагмента ЛОС-программы, в котором текущий (еще не выполненный) оператор выделен малиновым цветом. Далее возможен анализ текущей ситуации с помощью средств отладчика ЛОСа (см. последующие разделы) либо возвращение в главное меню при нажатии клавиши Esc. Можно также просмотреть текущую цепь задач (нажатием клавиши "з") либо (до нажатия "з"либо вернувшись из просмотра цепи задач в ЛОС-программу нажатием "ф") запустить процесс пошаговой трассировки на уровне той задачи, которая при прерывании оказалась текущей. Это делается нажатием клавиш "пробел" и "Enter".

Иногда пошаговый просмотр решения оказывается неудобен из-за того, что на некотором этапе начинается решение трудоемкой вспомогательной задачи, обращение к которой не было вынесено в самостоятельный шаг. Разумеется, по завершении этого решения и отображении срабатывания приема, в рамках которого оно происходило, решение вспомогательной задачи можно повторить для ознакомления с подробностями. Однако, длительная пауза из-за неотображаемого процесса решения может оказаться нежелательной. В таких случаях можно повторно запустить пошаговый просмотр решения, изменив его режим непосредственно перед появлением паузы. Для выбора нужного режима трассировки следует нажать клавишу "т" (это делается либо до запуска решения, из задачника, либо уже в начатом процессе трассировки). После нажатия появляется диалог установки режима.

Для ввода либо отмены условия на трассировку, определяемого в одном из пунктов диалога, следует переместить курсор мыши внутрь прямоугольника этого пункта и нажать левую клавишу мыши (наличие плюса справа от прямоугольника означает, что условие включено, иначе - выключено). После выбора необходимой комбинации условий, курсор мыши перемещается в прямоугольник "Ввести"и снова нажимается левая клавиша мыши.

Для преодоления указанных выше пауз можно воспользоваться пунктом "ручной выбор входа в подпроцесс". Если этот пункт активирован, то каждая попытка обращения решателя к вспомогательной задаче (включая наиболее крупные пакетные операторы) будет вводиться на экран. Чтобы продолжить решение без входа в пошаговую трассировку этой вспомогательной задачи, нажимается "Enter" ;чтобы войти в нее, нажимается "Ctr-Enter". Заметим, что в этом режиме текущее действие решателя не сопровождается выдачей списка вспомогательных задач, решенных для его выполнения - сообщения об обращениях к этим задачам уже выдавались на экран до соуществления действия, и была возможность просмотреть их решение по шагам. Недостатком режима с ручным входом в подпроцессы является чрезмерное количество выдаваемых на экран сообщений о попытках решения вспомогательных задач, бóльшая часть которых оказывается неудачной. Действительно ценные шаги тонут в этом потоке сообщений. Поэтому данный режим является лишь техническим, в обычных ситуациях отключенным.

Отладочные режимы трассировки описываются в разделе, посвященном отладчи-

ку ЛОСа. С помощью этих режимов можно, в частности, обеспечить прерывание процесса решения при попытке применения заданного приема, что часто используется при оптимизации его решающих правил.

Возможен серийный запуск решения задач из задачника. Такой запуск бывает необходим для контроля за изменениями в поведении решателя при его обучении. Простейшая форма этого запуска - выбор в оглавлении задачника нужного подраздела и нажатие клавиши "Ctr-з" либо клавиши "Ctr-э". В первом случае из цикла решения будут исключены все задачи, которые при предыдущем запуске решателем не были решены; во втором случае будут решаться все задачи подряд.

При серийном решении последовательно решаются сначала все задачи из текущего пункта просматриваемого меню (выделенного в момент запуска) оглавления задачника, затем - все задачи из следующего пункта этого меню, и т.д. до конца подраздела. На экране при этом последовательно сменяются формулировки решаемых задач. Если решатель слишком долго решает какую-либо задачу (возможно, "залипает"на ней из-за плохих решающих правил), то последовательное нажатие клавиш "Break" и "Esc" переводит в решение следующей задачи. Если до конца решения всех задач подраздела произойдет "зависание"программы и понадобится ее перезапуск, то после перезапуска автоматически восстанавливается прерванный цикл решения задач - вплоть до достижения конца меню.

Для обрыва серийного режима следует нажать клавишу "Break" (на экране появится фрагмент ЛОС-программы, в котором произошло прерывание, и установится пошаговый режим отладочной трассировки), и далее - нажать клавишу "Ctr-з". Лишь после этого нажатие клавиши "Esc"выведет в главное меню.

По окончании серийного запуска обновляется статистика о результатах решения задач подраздела. Такая статистика позволяет находить "особые точки"в задачнике (например, выявлять задачи, которые перестали решаться или решение которых замедлилось после внесенных в приемы изменений). Для просмотра статистики следует войти в меню нужного подраздела и нажать клавишу "з". После небольшой паузы, необходимой для просмотра всех задач подраздела, на экране возникает таблица, в которой указываются следующие сведения:

а) Пять наибольших величин замедления в решении задач по сравнению с предыдущим запуском. Эти величины измеряются в числе шагов интерпретатора ЛОСа, как и величины трудоемкости решения задач, сохраняемые в задачнике - отсюда легко извлекается процент замедления. Если нужно просмотреть имеющие самые большие значения замедления задачи, то нажимается клавиша "з", переводящая в просмотр первой из этих задач. Переход к очередной задаче (отобранные для просмотра задачи упорядочены по убыванию замедлений) осуществляется нажатием клавиши "ш". Для просмотра в таком режиме отбираются не более 40 задач подраздела (это же ограничение распространяется на другие приводимые ниже случаи отбора серий задач).

б) Число нерешенных задач раздела. Для просмотра этих задач нажимается клавиша "н". Чтобы перейти к просмотру очередной нерешенной задачи, следует нажать "ш".

в) Число утерянных решений задач подраздела (только по сравнению с предпоследним циклом решения). Для просмотра серии таких задач сначала нажимается "у", и далее - через нажатия "ш".

г) Пять наибольших величин ускорения в решении задач. Просмотр задач с ускорившимся решением - через нажатие клавиши "с",и далее к каждой следующей задаче - через нажатие "ш".

д) Суммарное замедление в решении задач подраздела (отрицательная его величина означает суммарное ускорение).

е) Число изменившихся по сравнению с предпоследним циклом решения ответов на задачи подраздела. Для просмотра серии соответствующих задач сначала нажимается клавиша "и", и далее - через нажатия "ш".

ж) Число сомнительных ответов. Для просмотра серии соответствующих задач - нажатие "о", и далее - через "ш".

з) Число задач, замедлившихся более чем на 10000 шагов интерпретатора ЛОСа, и число задач, замедлившихся более чем на 100000.

и) Можно просмотреть список всех ответов на задачи подраздела. Для этого нажимается клавиша "О". Если на выделенном ответе нажать клавишу "курсор вправо", то происходит переход к просмотру условия соответствующей задачи.

к) Пять наибольших величин трудоемкости решения задач подраздела (в числе шагов интерпретатора). Для просмотра задач в порядке убывания трудоемкости (не более 40 задач) - нажатие "т", и далее - через "ш".

Серийный запуск позволяет косвенно оценивать "холостой ход" приема - суммарную трудоемкость попыток его применения, не приводивших к срабатыванию приема. Фактически здесь оценивается относительная трудоемкость попыток применения приема на одно его срабатывание. Чтобы получить такую статистику, требуется запустить серийное решение не через "Ctr-з", а через "Ctr-х"(кир.). Тогда в указанной выше таблице статистики будут отображены данные о пяти наихудших случаях такой относительной трудоемкости ("холостого хода"). Эти данные суть пары (суммарная трудоемкость попыток применить прием - число срабатываний приема), для которых отношение первого элемента ко второму (если второй равен 0, то вместо него берется 1) наибольшее. Возможен просмотр серии наихудших приемов (по убыванию указанной характеристики) - через нажатие клавиши "х"(кир.). Переход к каждому очередному приему - через "ш". При просмотре приема повторный вызов на экран указанной пары чисел - через "Х"(кир.); пара (A_1, A_2) прорисовывается в виде двух термов - "пассив(A_1)"и "актив(A_2)".

4.3 Параллельная прокрутка решателя по задачнику

Прокрутку по задачнику можно распараллелить. Для этого следует заблаговременно создать в той же директории, где находится файл logsust.exe, поддиректории EX1,EX2,...,EXn. Число n не превосходит уменьшенного на 1 числа потоков, реализуемых процессором машины. При этом оно не должно превосходить 23. В названии поддиректорий сначала n записывается как цифра от 1 до 9, а далее - как латинская буква начиная с a. "Наибольшая" допустимая буква - n.

В каждой поддиректории EXn полностью копируются все директории GEN, INF, LOS, TER, TXT, TCH. К ним добавляется файл logsyst.exe, в названии которого буква t заменена числом (или буквой) n.

Для корректной прокрутки следут установить на компьютере пакет AutoIt, который бесплатно скачивается в интернете. Он необходим для автоматического перезапуска программой fenix.exe тех потоков, в которых при прокрутке произойдет непредвиденный сбой. Кроме того, нужно позаботиться, чтобы Windows не заблокировала главному потоку доступ к боковым потокам (установить для всех потоков разрешение доступа "Все").

Собственно запуск параллельной прокрутки по задачнику начинается с того, что в некотором (возможно, корневом) меню клавишей Ctr-1 выбирается начальный раздел прокрутки, а затем клавишей Ctr-2 - последний раздел. В случае корневого меню не рекомендуется выбирать последним раздел с номером, большим 17. Если используются 2 машины, нажимается Ctr-3, если 3 машины - Ctr-4. Иначе нажатие пропускается. Далее нажимается Ctr-л. На экране появятся небольшие окна для потоков, в которых будет отображаться текущая трудоемкость задачи. Отобранные задачи распределяются по этим потокам равномерно, и каждая порция прокручивается независимо. По мере завершения прокрутки окна потоков исчезают. Если поток остановлен операционной системой, а программа fenix.exe его не перезапустила, такой перезапуск необходимо выполнить вручную. Главный поток, после завершения прокрутки, приобретает зеленый цвет. Пока все потоки не завершатся, возвращение в однопотоковый режим не происходит. Перед возвращением выдерживается пауза, необходимая для пересылки данных из боковых потоков в главный. Дальнейший анализ итогов прокрутки - такой же, как в последовательном режиме прокрутки.

4.4 Упражнения по вводу и решению задач

Ввести перечисленные ниже задачи в соответствующие разделы задачника и просмотреть процесс их решения.

Элементарная алгебра

1. Упростить выражение:

$$\sqrt{\frac{4}{x}+\frac{1}{4x^{-1}}-2}+\sqrt{\frac{1}{4x^{-1}}+\frac{2^{-2}}{x}+\frac{1}{2}}$$

2. Решить уравнение:

$$\frac{x^2}{x+2}+1=\frac{4}{x+2}$$

3. Решить неравенство:

$$\frac{\sqrt{x^2+x-6}+3x+13}{x+5}>1$$

4. Решить систему уравнений:

$$\begin{cases}(x+y)(x^2-y^2) &= 16\\ (x-y)(x^2+y^2) &= 40.\end{cases}$$

5. Решить уравнение:

$$(tgx)^{\cos^2 x} = (ctgx)^{\sin x}$$

6. Решить неравенство:

$$|x-2|^{\log_4(x+2)-\log_2 x} < 1$$

7. Для всех a решить неравенство $ax^2 + (a+1)x + 1 > 0$.

8. При каких a неравенство $\sin^6 x + \cos^6 x + a \sin x \cos x \geq 0$ выполнено для всех значений x ?

9. Найти все a, при которых из неравенства $x^2 - a(1+a^2)x + a^4 < 0$ следует неравенство $x^2 + 4x + 3 > 0$.

10. При каких a система

$$\begin{cases} x^2 + y^2 &= 2(1+a) \\ (x+y)^2 &= 14 \end{cases}$$

имеет ровно два решения ?

Указания

1. Находясь в главном меню, выбрать пункт "Оглавление задачника" (клавиша "з" либо левая кнопка мыши в прямоугольнике указанного пункта). Используя клавиши курсора, найти в корневом меню оглавления раздел "Элементарная алгебра" (возможно, сначала понадобится несколько раз нажать "курсор влево"). Войти в этот раздел, далее войти в подраздел "Упрощение выражений", затем - в любой из подразделов "Упрощение иррациональных выражений - 1,2,3". В действительности выбор раздела несущественен; решатель будет работать вне зависимости от того, в каком разделе находится задача, и классификация задач по разделам нужна лишь для упрощения поиска нужной задачи вручную. Используя "PageDown" либо "курсор вниз", вывести на экран последний пункт выбранного концевого раздела (все его пункты - номера с тремя штрихами). Затем нажать клавишу "к" (кир.) для создания бланка новой задачи. Далее нажимается "ц" и в оглавлении типов целевых установок выбираются разделы "Преобразовать выражение" - "Упростить выражение в области допустимых значений", причем после выбора последнего пункта нажимается "курсор вправо". Экран расчищается, и в верхней его части возникает текст "Упростить в о.д.з. выражение:". Далее нажимается "Enter", и формульным редактором вводится упрощаемое выражение. В процессе набора можно получать справки о клавиатуре формульного редактора, нажимая F1. Точнее, нажатие F1 переводит в общее оглавление справочника по системе, и из его корневого меню нужно перейти в раздел "Формульный редактор". Далее полезно прочитать раздел "общие сведения"; для получения информации о вводе конкретных символов - переходить в соответствующие подразделы. Чтобы вернуться в набор формулы, достаточно нажать "End". По завершении набора упрощаемого выражения нажимается "Enter". В данном примере задача оказывается полностью введенной, и для решения ее без пошаговой трассировки

нажимается "о" (кир.), а для входа в пошаговую трассировку (она отображает лишь верхний уровень процесса решения - срабатывания приемов; такую трассировку, в отличие от отладочной трассировки, называем далее семантической) нажимается "р". Каждый очередной шаг трассировки - нажатие "Enter". Если в некоторый момент под кадром, поясняющим текущее действие, оказываются расположены другие кадры, то эти последние суть кадры обращений к вспомогательным задачам, решавшимся для выполнения текущего действия. Можно выбрать любой из них для детального просмотра решения вспомогательной задачи. При этом верхняя отделяющая линия кадра должна быть в точности верхней границей экрана; такое выравнивание обеспечивается клавишами "Ctr-курсор вверх либо вниз". Вход в решение вспомогательной задачи - нажатие "Enter". По завершении трассировки решения вспомогательной задачи нажатие "Enter"возвращает на предыдущий уровень (такое возвращение можно обеспечить в процессе трассировки нажатием "End").

2. Действия аналогичны предыдущим, но выбирается раздел "Элементарная алгебра" - "Решение уравнений" - "Рациональные уравнения 1,2". Нажимаются "к","ц", и выбирается раздел оглавления целевых установок "Найти значения неизвестных" - "Получить полное явное описание значений неизвестных". На экране появляется текст "Найти", под которым размещен курсор формульного редактора. Этим редактором вводятся неизвестные задачи (отделенные запятой) - в данном примере единственная переменная x. После нажатия "Enter"ввод целевой установки завершается. Для ввода уравнения снова нажимается "Enter", и далее происходит набор уравнения.

3. Действия аналогичны предыдущим, но выбирается подраздел для неравенств.

4. Для вводы системы уравнений и (или) неравенств сначала вводится целевая установка (аналогично предыдущему, но число неизвестных может быть более одной). Затем по отдельности вводятся условия - уравнения и неравенства; ввод каждого нового условия начинается с "Enter" и завершается "Enter". После ввода последнего условия задача готова к запуску решателя.

5. Аналогично задаче 2; тригонометрические операции вводятся последовательным набором двух (в особых случаях трех) первых латинских букв обозначения этих операций: синус - s,i; косинус - c,o; тагенс - t,g; котангенс - c,t, и т.д. Заметим, что степень тригонометрической операции набирается нестандартным образом: сначала набирается вся операция, затем - курсор вверх, затем - показатель степени и "Enter". Тригонометрическая операция относится к наименьшему расположенному после нее осмысленному выражению (суммы и произведения под такой операцией следует заключать в скобки), и степень оказывается относящейся не к аргументу операции, а ко всей операции.

6. Аналогично задаче 3; вертикальные отрезки модуля вводятся с помощью клавиш "Ctr-м" (лат.), для ввода логарифма последовательно вводятся буквы l,o, после чего набирается основание логарифма. После набора основания нажимается "Enter", и набирается выражение под логарифмом (суммы и произведения следует заключать в скобки).

7. Аналогично предыдущей задаче; параметр a не включается в число неизвестных задачи, и никак более не выделяется.

8. Неизвестной задачи служит переменная a. Условие ее набирается в виде

$$\forall_x(x - \text{число} \rightarrow \sin x^6 + \cos x^6 + a \sin x \cos x \geq 0).$$

Для ввода "x – число" сначала вводится x, затем нажимается /, затем "ч".

9. Аналогично предыдущему; условие набирается в виде:

$$\forall_x(x^2 - a(1 + a^2)x + a^4 < 0 \rightarrow x^2 + 4x + 3 > 0.$$

Заметим, что понятие "следует" здесь трактуется таким образом, что если левая от стрелки часть ложная (в частности, неравенство слева вообще не имеет решений), то вся импликация истинна. Это приводит к тому, что концевые точки указанного в ответе промежутка (для них левое неравенство не имеет решений) отнесены к ответу.

10. Условие задачи состоит в том, что множество пар (x, y), удовлетворяющих уравнениям, имеет мощность 2. Оно записывается в виде

$$card(set_{xy}(x^2 + y^2 = 2(1 + a)\ \&\ (x + y)^2 = 14)) = 2.$$

Планиметрия

1. Основание равнобедренного треугольника равно a, угол при вершине равен b. Найдите биссектрису, проведенную к боковой стороне.

2. В трапеции $ABCD$ с основаниями AD и BC имеем $AD = 3$, $BC = 1$. Точка P лежит на стороне AB, а точка Q - на стороне CD, причем отрезок PQ параллелен основаниям и проходит через точку пересечения диагоналей трапеции. Найти длину отрезка PQ.

3. Известно, что $ABCD$ - ромб и радиусы окружностей, описанных около треугольников ABC и ABD соответственно, равны R и r. Найти площадь ромба $ABCD$.

4. В параллелограмме $ABCD$ биссектриса угла BAD пересекает сторону CD в точке M, причем $DM/MC = 2$. Известно, что угол CAM равен a. Найти угол BAD.

5. Окружность проходит через вершины A и C треугольника ABC, пересекает сторону AB в точке D и сторону BC в точке E. Известно, что $AD = 5, AC = 2\sqrt{7}, BE = 4, BD/CE = 3/2$. Найти угол CDB.

Указания

1. Вычислительные задачи по геометрии вводятся в следующем порядке: сначала набираются посылки задачи, перечисляющие известные свойства чертежа; затем вводится целевая установка задачи; затем указываются неизвестные величины, которые должны быть вычислены. Перед набором посылок можно ввести чертеж, однако чертеж может быть создан системой и автоматически (по окончании набора задачи нажимается "Ctr-ч").

При наборе посылок следует обязательно обозначить все точки, участвующие в задаче, буквами (обычно большими; можно использовать натуральные индексы). Ссылаться на плоские фигуры (треугольники, многоугольники, окружности, прямые, и т.д.) можно только через их "базисные" точки. В нашем случае обозначим вершины треугольника буквами A, B, C и введем первую посылку "$\triangle(ABC)$". Для ввода посылки нажимается "Enter", затем последовательно нажимаются клавиши "т","р"(обычно используются первые две буквы вводимого понятия). Это приводит к прорисовке символа $\triangle$ с расположенной после него открывающей скобкой. Далее последовательно нажимаются A,B,C и закрывающая скобка, после чего - завершающее набор формулы "Enter". Никакие разделители между буквами не ставятся. Заметим, что если в обозначении треугольника либо другой фигуры используется буква с индексом, то после такой буквы, перед набором следующей, обязательно нужно нажать на клавишу "умножение" (звездочка).

После ввода первой посылки (она указывает только на то, что три точки A, B, C образуют вершины некоторого треугольника, то есть не лежат на одной прямой) нужно ввести посылку, определяющую, что треугольник равнобедренный. Выберем в качестве основания треугольника вершины A, C и введем посылку $l(AB) = l(BC)$. Для обозначения расстояния дважды нажимается клавиша "l", что приводит к появлению рукописной латинской буквы l с идущей после нее открывающей скобкой. Затем (как в обозначении треугольника) подряд вводятся буквы для точек, между которыми рассматривается расстояние, и ставится закрывающая скобка.

Далее вводятся: посылка $l(AC) = a$, обозначающая длину основания треугольника через a, и посылка $\angle(ABC) = b$ - для величины угла при основании. Чтобы получить обозначение угла, последовательно нажимаются клавиши "у","г". Далее - как для обозначения треугольника. Как и обычно, вершина угла размещается в обозначении угла посередине.

Для ввода основания D биссектрисы треугольника ABC, проведенной из угла BAC, можно использовать посылку "Биссектреуг($BACD$)". Другой способ (более громоздкий, но не использующий специального обозначения "Биссектреуг") - пара посылок "$D \in$ прямая(BC)", "биссектриса($BACD$)". В первом случае нажимаем "И","Т", и далее - как в случае обозначения треугольника, но для четырех букв. Во втором случае сначала вводим D, затем нажимаем пробел, b, e (лат.; появляется символ $\in$). Далее нажимаем "п","р" (кир.; появляется слово "прямая" с открывающей скобкой), вводим буквы B, C и закрывающую скобку. Для ввода "биссектриса(...)" нажимаем клавиши "и","т".

На этом ввод посылок завершен. Для ввода целевой установки нажимаем клавишу "ц", переводящую в оглавление типов целевых установок. Из корневого меню этого оглавления переходим к пункту "Найти значения неизвестных" - "Выразить значения неизвестных через заданные параметры". Заметим, что применявшийся в элементарной алгебре пункт "Получить полное явное описание значений неизвестных" в планиметрических задачах на вычисление НЕ следует использовать. Это объясняется принципиальным различием понятий "известная" и "неизвестная" в задачах из двух этих разделов. В элементарной алгебре все переменные из списка посылок по умолчанию считались известными и могли входить в ответ задачи. В геометрической задаче на вычисление посылки содержат обозначения точек $A, B, C, \ldots$, которые не должны появляться в

ответе. Соответственно различаются и типы выбираемых целевых установок. После выбора указанного типа целевой установки нажимается "курсор вправо". Далее сначала вводится буква (или несколько отделенных запятыми букв) для неизвестной (неизвестных) и нажимается "Enter". Затем вводятся разделенные запятыми буквы для известных числовых параметров, через которые должны быть выражены неизвестные (если таких параметров вообще нет, сразу нажимается "Enter", иначе оно нажимается после ввода параметров). В нашем примере обозначим неизвестную длину биссектрисы через x; параметры суть a, b.

После ввода целевой установки вводим равенства, связывающие неизвестные (переменные) с теми выражениями, которые они обозначают и которые нужно вычислить. В ответ войдет сама неизвестная, а не обозначаемое ею выражение.

Как уже говорилось выше, после ввода задачи по планиметрии можно ввести чертеж - либо попробовать сделать это автоматически (нажатием "Ctr-ч"), либо ввести чертеж вручную (нажать "ч" и далее воспользоваться геометрическим редактором; чертеж появится перед списком посылок, в начале задачи). Можно также скорректировать вручную чертеж, созданный автоматически (вход в редактирование уже созданного чертежа - снова через "ч"; удаление чертежа - выделить его левой кнопкой мыши и нажать "Ctr-Del").

2. Напомним, что в решателе предусмотрены два варианта обозначения трапеции: "трапеция($ABCD$)" и "Трапеция($ABCD$)" - первый из них для случая трапеции с большим основанием AD и острыми углами при основании; второй - для случая, когда углы при основании не обязательно острые. В обоих случаях указанные записи предствляют собой даже не обозначения трапеции, а лишь утверждения о том, что точки A, B, C, D являются вершинами соответствующей трапеции. Сама трапеция (как и любой другой четырехугольник) обозначается посредством выражения "фигура($ABCD$)". В нашем примере годится любой из указанных способов записи. Например, введем посылку "трапеция($ABCD$)". Затем вводятся посылки $l(AD) = 3, l(BC) = 1$. Принадлежность точки P стороне AB, а точки Q - стороне CD, записываются в виде $P \in$ отрезок(AB), $Q \in$ отрезок(CD). Параллельность отрезка PQ основаниям трапеции записывается как прямая(PQ)$\|$прямая(AD). Чтобы сформулировать условие о том, что отрезок PQ проходит через точку пересечения диагоналей трапеции, нужно ввести обозначение для этой точки. Например, обозначим ее через M. То, что M является точкой пересечения диагоналей, записываем как $M \in$ прямая(AC), $M \in$ прямая(BD). Далее добавляем посылку $M \in$ отрезок(PQ). На этом ввод посылок завершается. Как и в предыдущей задаче, выбираем целевую установку и вводим единственное условие $x = l(PQ)$ для неизвестной x.

3. Начинаем со ввода посылки "ромб($ABCD$)". Так как в задаче речь идет об описанных окружностях, надо ввести обозначения для этих окружностей, то есть обозначить центр окружности и выбрать какую-либо точку на окружности (в планиметрии ссылки на окружности могут быть только такими). Например, обозначим центры через M, N. В случае описанной окружности, которая должна проходить через вершины треугольника, в качестве второй точки можно взять какую-либо вершину треугольника (например, A). Тогда добавятся посылки "окружность(MA) описана около фигура(ABC)"; "окружность(NA) описана около фигура(ABD)". Заметим, что хотя эти тексты кажутся доста-

точно длинными, набираются они весьма малым числом нажатий клавиш: сначала нажимаем "о","к" (кир.) - появляется слово "окружность" с открывающей скобкой. Затем вводим буквы M, A и закрывающую скобку. Далее нажимаем "Ctr-ф" - появляются слова "описана около". Наконец, нажимаем "ф","и" - появляется слово "фигура"с открывающей скобкой. В заключение вводим A,B, C и закрывающую скобку. Радиусы окружностей указываем с помощью посылок $l(MA) = R, l(NA) = r$. При вводе целевой установки указываем, что значение неизвестной x должно быть выражено через R, r. Наконец, набираем условие $x = S(\text{фигура}(ABCD))$. Заметим, что знак площади S здесь вводится двукратным нажатием малой латинской буквы s; если его ввести нажатием клавиши большой латинской S, то задача окажется набранной неверно и не будет решена (вместо площади окажется введенным значение какой-то неопределенной функции S).

4. Вводятся посылки "параллелограмм$(ABCD)$", "биссектриса$(BADM)$", $M \in$ отрезок(CD), $l(DM)/l(MC) = 2$, $\angle(CAM) = a$. Затем выбирается целевая установка "Выразить x через a", и вводится условие $x = \angle(BAD)$.

5. Вводятся посылки $\triangle(ABC)$, $C \in$ окружность(PA), $D \in$ окружность(PA), $D \in$ отрезок(AB), $E \in$ окружность(PA), $E \in$ отрезок(BC), $l(AD) = 5, l(AC) = 2\sqrt{7}, l(BE) = 4, l(BD)/l(CE) = 3/2$. Затем вводятся целевая установка и условие $x = \angle(CDB)$.

Математический анализ

1. Вычислить производную функции

$$\frac{e^{-x^2}\arcsin(e^{-x^2})}{\sqrt{1-e^{-2x^2}}} + \frac{1}{2}\ln(1-e^{-2x^2})$$

2. Вычислить предел функции

$$(2e^{\frac{x}{x+1}} - 1)^{\frac{x^2+1}{x}}$$

при $x \to 0$.

3. Исследовать поведение функции $y = (x-5)\sqrt[3]{x^2}$.

4. Найти точки экстремума для функции $y = x^2 - ln(x^2)$.

5. Исследовать на непрерывность функцию $y = \text{arctg}(1/x + 1/x - 1 + 1/x - 2)$.

6. Найти неопределенный интеграл

$$\int \frac{2\sin x - \cos x + 3}{3\sin x + \cos x + 1} dx$$

7. Вычислить определенный интеграл

$$\int_0^{ln2} \sqrt{e^x - 1} dx$$

8. Вычислить двойной интеграл от $\sqrt{|x-y^2|}$ по области $(x,y) : |y| \le 1, 0 \le x \le 2$.

9. Найти площадь области, заключенной между параболой $y^2 = \frac{b^2}{a}x$ и прямой $y = \frac{b}{a}x$; $0 < a, 0 < b$.

10. Найти объем тела $x^2 \le ay \le bx, x^2 + y^2 \le hz \le 2x^2 + 2y^2$; $0 < a, 0 < b, 0 < h$.

11. Найти площадь поверхности $z = xy, x^2 + y^2 \le R^2$.

12. При каких значениях параметра p сходится ряд

$$\sum_{i=1}^{\infty} \frac{1}{i^p} \sin \frac{\pi}{i}?$$

13. Разложить в ряд Тейлора по переменной x в точке 0 функцию

$$y = (x - \operatorname{tg} x) \cos x$$

14. Разложить в ряд Фурье на отрезке $[-\pi, \pi]$ функцию $y = (\pi^2 - x^2)^2$.

15. Найти сумму ряда

$$\sum_{n=1}^{\infty} \frac{3n^2 + 3n + 1}{n^3(n+1)^3}.$$

Указания

1. Прежде всего нужно войти в оглавление целевых установок и выбрать пункт "Упростить выражение в области допустимых знчений". Затем набирается производная указанного выражения. Она вводится как дробь вида $\frac{dA}{dx}$; A - дифференцируемое выражение. После символа d нужно ставить знак умножения; дифференцируемое выражение заключается в скобки. В принципе, переменную d можно использовать и внутри выражения A, и даже взять ее в качестве x. Если нужно найти значение производной в некоторой точке t, то вместо dx набирается $d(x = t)$, причем и здесь после d ставится знак умножения.

2. Снова вводится установка "Упростить выражение в области допустимых значений". Затем набирается условие: сначала нажимаются клавиши l, i - появляется символ lim, справа внизу от которого размещен курсор. Набираем $x \to 0$ (стрелка вводится клавишей "курсор вправо") и нажимаем Enter - курсор перемещается вверх в ту же строку, в которой расположен символ lim. Здесь набираем выражение под знаком предела. Это выражение обязательно нужно заключить в скобки, иначе знак предела будет отнесен к минимальному осмысленному его началу. В нашем примере, если не заключить все выражение в скобки, то степень окажется вне предела и ответ будет другим.

3. Выбирается целевая установка "Исследовать поведение функции", в которой указывается обозначение функции - переменная y. Затем вводится условие $y = \lambda_x((x-5)\sqrt[3]{x^2}, x - \text{число})$.

4. Выбираем переменные, которые будут обозначать, соответственно, точку экстремума, значение функции в этой точке, и тип экстремума (максимум либо минимум). Например, пусть это будут переменные u, v, w. Вводим целевую установку "Найти полное явное описание значений неизвестных" в которой указываем выбранные неизвестные. Затем вводим условие

$$Extr(\lambda_x(x^2 - \ln(x^2), x - \text{число}), u, v, w).$$

Аналогично вводятся задачи на поиск множества точек минимума либо максимума некоторой функции внутри заданной области, но число неизвестных здесь будет равно двум (искомое множество точек и значение функции в этих точках). Вместо $Extr$ используются символы Min, Max, причем после функции должна быть указана область, по которой берется минимум или максимум.

5. Отличие от общего исследования поведения функции - только в том, что выбирается целевая установка "Исследовать функцию на непрерывность".

6. Выбирается целевая установка "Упростить выражение в области допустимых значений". Затем набирается неопределенный интеграл: нажимается "Ctr-j"и вводится подынтегральное выражение, которое умножается справа на произведение dx.

7. Аналогично неопределенному интегралу, но нажимается "Ctr-i". Тогда курсор сначала оказывается под знаком интеграла, где набирается нижний предел. После нажатия "Enter" курсор переводится вверх, где набирается верхний предел. Еще одно нажатие "Enter" - и набирается подынтегральное выражение, умножаемое на dx.

8. При вычислении двойных интегралов область интегрирования обычно задается в списке посылок. Выбираем для нее обозначение - например, переменную P, и введем посылку $P = set_{xy}(|y| \le 1 \ \& \ 0 \le x \ \& \ x \le 2)$. Затем вводим целевую установку "Упростить выражение в области допустимых значений". Затем набираем двойной интеграл - нажимаем "Ctr-2"; под интегралом вводим обозначение области P, нажимаем "Enter", и далее набираем подынтегральное выражение, умноженное на произведение $dxdy$.

9. Плоскую область определяем в списке посылок, обозначив ее вспомогательной переменной (например, P). Эту область задаем, используя ссылку на прямоугольную систему координат, которую тоже обозначаем вспомогательной переменной (например, K). После этого в нашем примере вводим посылки

$$P = \text{точки}(\text{областьграницы}(set_{xy}(y^2 = \frac{b^2}{a}x) \cup set_{xy}(y = \frac{b}{a}x)), K),$$

$0 < a, 0 < b$, "прямкоорд(K)". Заметим, что операция "областьграницы" применяется к теоретико-множественному объединению пар координат точек кривых, ограничивающих область, а значением этой операции служит множество пар координат точек области. Для перехода от пар координат к точкам плоскости используется операция "точки". Далее выбирается целевая установка "Упростить выражение в области допустимых значений" и вводится условие $S(P)$. Как и в случае геометрических задач, символ площади S вводится двойным нажатием клавиши s.

10. В случае нахождения объемов применяется аналогичная конструкция, но множество троек координат трехмерной области задается непосредственно, с помощью неравенств. В нашем примере вводим посылки

$$P = \text{точки}(set_{xyz}(x^2 \le ay \ \&\ ay \le bx \ \&\ x^2 + y^2 \le hz \ \&\ hz \le 2x^2 + 2y^2), K),$$

$0 < a$, $0 < b$, $0 < h$, "прямкоорд(K)". Целевая установка - та же, что и в предыдущем примере; условие имеет вид "объем(P)" (двукратное нажатие клавиши "О", кир.).

11. Посылки вводятся аналогично предыдущей задаче, причем вместо множества троек координат точек трехмерной области задается множество троек координат точек поверхности. Условие имеет вид $S(P)$.

12. Целевая установка при наличии параметров ряда - "Получить полное явное описание значений неизвестных" (при их отсутствии - "Проверить истинность утверждения"); неизвестная - параметр ряда p. Условие имеет вид утверждения о сходимости последовательности частичных сумм ряда:

$$\text{сходится}(\lambda_n(\sum_{i=1}^{n}(\frac{1}{i^p}\sin\frac{\pi}{i}), n - \text{натуральное}))$$

13. Выбирается установка "Разложить в ряд Тейлора", в которой указываются переменная разложения и точка разложения. Условием задачи служит выражение $(x - \operatorname{tg} x)\cos x$.

14. Аналогично предыдущему - с установкой "Разложить в ряд Фурье".

15. Установка - "Упростить выражение в области допустимых значений". Условие задачи набирается непосредственно в виде бесконечной суммы.

Аналитическая геометрия и линейная алгебра

1. Даны три вектора $a(4, 1, 5)$, $b(0, 5, 2)$ и $c(-6, 2, 3)$. Найти вектор x, удовлетворяющий системе уравнений $(x, a) = 18, (x, b) = 1, (x, c) = 1$. Система координат прямоугольная.

2. Даны координаты двух вершин треугольника $A(-1, 3), B(2, 5)$ и точки пересечения его высот $H(1, 4)$ в прямоугольной системе координат. Найти координаты третьей вершины треугольника и составить уравнения его сторон.

3. Составить уравнения плоскостей, проходящих через прямую $\frac{x-1}{3} = \frac{y-1}{5} = \frac{z+2}{4}$ и равноудаленных от точек $A(1, 2, 5)$ и $B(3, 0, -1)$.

4. Составить уравнение касательной к параболе $y^2 = -8x$, отрезок которой между точкой касания и директрисой делится осью Oy пополам. Система координат прямоугольная.

5. Найти уравнение плоскости, пересекающей эллипсоид $x^2 + 2y^2 + 4z^2 = 9$ по эллипсу, центр которого находится в точке $C(3, 2, 1)$. Система координат прямоугольная.

6. Поверхность задана уравнением $6xy-8y^2-z^2+60y+2z+89=0$ в прямоугольной системе координат. Найти каноническую систему координат и каноническое уравнение этой поверхности. Определить тип поверхности.

7. Решить матричное уравнение:

$$\begin{pmatrix} 2 & -3 & 1 \\ 4 & -5 & 2 \\ 5 & -7 & 3 \end{pmatrix} \cdot X \cdot \begin{pmatrix} 9 & 7 & 6 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & -2 \\ 18 & 12 & 9 \\ 23 & 15 & 11 \end{pmatrix}$$

8. Вычислить определитель

$$\begin{vmatrix} \sin a & \cos a & \sin(a+d) \\ \sin b & \cos b & \sin(b+d) \\ \sin c & \cos c & \sin(c+d) \end{vmatrix}$$

9. Найти собственные значения и собственные векторы линейного преобразования, заданного матрицей:

$$\begin{pmatrix} 3 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 3 & 0 & 5 & -3 \\ 4 & -1 & 3 & -1 \end{pmatrix}$$

10. Найти каноническую жорданову форму для матрицы:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Указания

1. В задачах по аналитической геометрии обычно приходится вводить обозначение для используемой системы координат. Обозначим ее в нашем примере через K и введем посылку "прямкоорд(K)". Напомним, что в решателе системы координат на плоскости отождествляются с тройками точек общего положения (первая точка - начало системы координат, две последние - концы координатных векторов), а в пространстве - с четверками таких точек. Если в задаче имеются ссылки на координатные оси либо плоскости, то приходится явным образом вводить такие тройки либо четверки точек. В нашем примере этих ссылок нет, поэтому далее вводим посылки, определяющие координаты векторов a, b, c относительно K: "коорд(a, K) = (4, 1, 5)"; "коорд(b, K) = (0, 5, 2)"; "коорд(c, K) = (−6, 2, 3)". Затем вводим посылки "скалумнож(a, x) = 18", "скалумнож(b, x) = 1", "скалумнож(c, x) = 1". Выбираем целевую установку "Выразить значения неизвестных через заданные параметры" и вводим переменную y для неизвестной. Наконец, вводим условие задачи "y = коорд(x, K)".

2. Вводим обозначение K для прямоугольной системы координат и заносим посылки "прямкоорд(K)", $\triangle(ABC)$, "коорд(A, K) = (−1, 3)", "коорд(B, K) =

$(2,5)$". Чтобы определить H как точку пересечения высот, проводим две высоты - из вершины A и из вершины B. Это можно сделать, добавив две посылки "Высота$(ABCM)$" и "Высота$(BCAN)$"; M, N - основания высот. Далее добавляем посылки, связанные с точкой H: $H \in$ прямая(AM); $H \in$ прямая(BN); "коорд$(H,K) = (1,4)$". Выбираем в качестве неизвестных переменные x, y, z, v - для уравнений сторон треугольника и координат его третьей вершины. Наконец, создаем целевую установку (так же, как в предыдущей задаче) и вводим условия: $x =$ коорд(прямая$(AB), K)$, $y =$ коорд(прямая$(AC), K)$, $z =$ коорд(прямая$(BC), K)$,$v =$ коорд(C, K).

3. В этой задаче система координат не предполагается прямоугольной. Поэтому выбираем обозначающую ее переменную K, но никаких специальных посылок для K не вводим. Так как в условии задачи говорится о прямой, а прямые в решателе обозначаются только с помощью пары точек, выбираем переменные P, Q для обозначения этих точек. После этого можно ввести посылку, задающую уравнение прямой: "коорд(прямая$(PQ), K) = set_{xyz}$(пропорцнаборы$((x-1, y-1, z+2), (3,5,4)))$". Заметим, что использование отношения "пропорцнаборы" пропорциональности двух числовых наборов при задании канонического уравнения прямой в пространстве является для решателя стандартом. Чтобы обозначить плоскость, выбираем переменные C, D, E для трех точек общего положения на этой плоскости. Затем вводим посылку, выражающую включение прямой PQ в искомую плоскость: "прямая$(PQ) \subseteq$ плоскость(CDE)". Вводим посылки, определяющие координаты указанных в задаче точек A, B: "коорд$(A, K) = (1,2,5)$", "коорд$(B,K) = (3,0,-1)$". Наконец, указываем на равноудаленность плоскости CDE от точек A, B: "расстдоплоскости$(A,$ плоскость$(CDE)) =$ расстдоплоскости$(B,$ плоскость$(CDE))$". Затем вводим целевую установку (так же, как в предыдущих задачах), указывая в ней неизвестную x для координат плоскости CDE. Завершаем набор задачи условием $x =$ коорд(плоскость$(CDE), K)$.

4. Так как система координат прямоугольная, вводим посылку "прямкоорд(K)". Обозначаем параболу через E и указываем ее уравнение: "коорд$(E,K) = set_{xy}$ $(y^2 = -8x)$". Выбираем обозначение "прямая(AB)"для касательной и вводим посылку "прямая$(AB) -$ касательная к E". Для точки касания можно было бы выбрать специальное обозначение, однако без ограничения общности можно считать, что этой точкой служит точка A. Поэтому вводим посылку $A \in E$. Для директрисы вводим обозначение "прямая(FG)". Так как в решателе упоминание о директрисе связано с обязательным упоминанием фокуса, которому она соответствует, то выбираем обзначение C для фокуса и заносим посылки "фокус(C,E)", "директриса(прямая(FG),$C, E)$". В качестве точки пересечения касательной с директрисой можно без ограничения общности выбрать точку B; соответствующая посылка имеет вид $B \in$ прямая(FG). Для обозначения середины отрезка AB выбираем переменную D и заносим посылки $D \in$ отрезок(AB), $l(AD) = l(BD)$. Чтобы указать на принадлежность точки D оси ординат, вводим явное обозначение для тройки точек, образующих систему координат K : $K = (M,N,P)$. Теперь принадлежность точки D оси ординат записывается в виде $D \in$ прямая(MP). Чтобы предотвратить недопустимый в задаче случай совпадения прямой AB с осью ординат, добавляем посылку $\neg(A \in$ прямая$(MP))$. Далее вводим целевую установку

(как в предыдущей задаче) с упоминанием неизвестной x и добавляем условие "x = коорд(прямая(AB), K)".

5. Вводим посылки "прямкоорд(K)" и "коорд(E, K) = $set_{xyz}(x^2 + 2y^2 + 4z^2 = 9)$"; E - обозначение эллипсоида из условия задачи. Обозначаем искомую плоскость "плоскость(ABC)", а сечение ею эллипсоида - F, после чего вводим посылки $E \cap$ плоскость(ABC) = F и "эллипс(F)". Обозначив центр эллипса через D, добавляем посылки "центр(D, F)" и "коорд(D, K) = $(3, 2, 1)$". Далее вводим целевую установку с неизвестной x и условие "x = коорд(плоскость(ABC), K)".

6. Вводим посылку "прямкоорд(K)" и выбираем целевую установку "Исследовать свойства поверхности, заданной своим уравнением". В этой установке обозначаем исследуемую поверхность через E, и вводим условие задачи: "коорд(E, K) = $set_{xyz}(6xy - 8y^2 - z^2 + 60y + 2z + 89 = 0)$".

7. Для решения матричного уравнения используем обычную целевую установку уравнений "Получить полное явное описание значений неизвестных". Для набора матрицы нажимаем клавиши "м","а" (кир.) - появляется левая скобка матрицы. Затем вводится первая строка, причем после каждого элемента нажимается "Enter". Чтобы перейти к следующей строке, после набора последнего элемента вместо "Enter" нажимается "Page Down". После набор последней строки нажимается "End" - возникает правая скобка матрицы. Для отката в случае ошибочных действий при наборе матрицы используется "Backspace". После набора первой матрицы вводится обычный символ умножения (клавиша "звездочка"), затем X, затем снова умножение, далее - вторая матрица, и после знака равенства - третья. Преобразование умножения чисел в матричное умножение предпринимается автоматически при запуске решения задачи.

8. Для вычисления определителя выбирается целевая установка "Упростить выражение в области допустимых значений". При наборе условия сначала нажимаются клавиши "d","e"(лат.) - возникает символ det. Затем водится матрица.

9. Выбираем обозначение для матрицы - переменную A. Затем вводим посылку - равенство с переменной A в левой части и матрицей в правой. Выбираем в качестве неизвестных переменные x, y, z; x - собственное значение, y - его кратность, z - собственный вектор, отвечающий данному собственному значению. Затем выбираем целевую установку "Получить полное явное описание значений неизвестных" для указанных неизвестных и вводим условия "собствзначение(A, x, y)", "собствектор(A, x, z)". В ответе для z приводится условие принадлежности множеству линейных комбинации базиса собственных векторов, соответствующих собственному значению x.

10. Выбираем целевую установку "Получить полное явное описание значений неизвестных" для неизвестных x (жорданова форма) и y (матрица, преобразующая к жордановой форме, т.е. такая, что произведение обратной к ней матрицы, исходной и снова матрицы y, равно x). Затем вводим условие "жордформа(A,x, y)", где A - исходная матрица.

Дифференциальные уравнения

1. Решить уравнение

$$2(y - 2xy - x^2\sqrt{y}) + x^2y' = 0$$

2. Решить уравнение
$$xyy'' + xy'^2 - yy' = 0$$

3. Найти решения уравнения $y = xy' - 2y'^2$, проходящие через точку (4,2).

4. Найти решение системы дифференциальных уравнений:
$$\begin{cases} \frac{dx}{dt} = x - y + 4\cos(2t) \\ \frac{dy}{dt} = 3x - 2y + 8\cos(2t) + 5\sin(2t). \end{cases}$$

Указания

1. Выбираем целевую установку "Решить функциональные уравнения" (эта установка берется и для остальных задач на решение дифференциальных уравнений). В качестве неизвестной указываем уже не переменную y, а выражение $y(x)$. Затем вводим условие задачи - уравнение, в котором везде вместо функции y записывается ее значение $y(x)$ в точке x, а вместо производной y' - $\frac{dy(x)}{dx}$. Напомним, что по всем произвольным постоянным, возникающим при интегрировании уравнения, в ответе навешивается квантор существования.

2. Задача вводится аналогично предыдущей; производная второго порядка набирается в виде $\frac{d^2y(x)}{dx^2}$. Символ дифференцирования d здесь рассматривается как переменная - то есть после него или его степени нужно нажимать клавишу "звездочка" для умножения.

3. После ввода дифференциального уравнения добавляем условие $y(4) = 2$.

4. В целевой установке указываем две неизвестных функции - $x(t), y(t)$. Уравнения системы набираются аналогично предыдущим задачам; вместо x, y в них используются записи $x(t), y(t)$.

4.5 Логический калькулятор

Для ускоренного ввода и решения задач создан так называемый логический калькулятор. Вход в него происходит через пункт "Решить задачу" главного меню системы. После выбора нужного концевого пункта возникающего здесь оглавления нужно зайти в него нажатием "курсор вправо" и далее действовать согласно появляющимся инструкциям. В нижней части экрана размещаются сведения по набору формул рассматриваемого раздела. Если этих сведений недостаточно, следует нажать F1, перейти в справочник по системе и из его корневого меню использовать раздел "Формульный редактор". Иногда может быть полезен также раздел "Логический язык системы".

По завершении ввода задачи решатель сохраняет ее в разделе "Буфер - Последние задачи" оглавления задачника и сразу приступает к решению. После получения ответа либо отказа можно продолжить работать с сохраненной задачей стандартными средствами - запустить просмотр шагов решения, перенести задачу в другой раздел и т.п. Для расчистки буфера достаточно, находясь в любой точке задачника, нажать "Shift-o" ("o" кириллица).

Разумеется, возможности логического калькулятора ограничены текущими средствами решателя. Иногда они достаточно для решения задач стандартных типов, иногда - нет. Последние случаи даже более ценны, так как подсказывают возможности дальнейшего пополнения базы приемов.

4.6 Анализ траекторий решения задач при обучении решателя

Пополнение базы приемов решателя происходит при ручном обучении только за счет анализа траекторий решения задач. Создавать какие-либо приемы из общих соображений, без примерки на задачах, не рекомендуется. Велика вероятность того, что такой прием не будет использоваться решателем - либо из-за того, что он окажется избыточным и его срабатывание предвосхитят другие приемы, либо из-за того, что действия других приемов уведут задачу из области его применимости. Обучающий материал для решателей дают задачники, а не учебники.

Таким образом, чтобы научиться создавать решатели, прежде всего нужно овладеть техникой разбиения траектории решения задачи на последовательность применений приемов. Отсюда возникают исходные неформальные версии описаний приемов, которые затем уже записываются на ГЕНОЛОГе или на ЛОСе. Разумеется, для одной и той же задачи могут быть предложены сильно отличающиеся друг от друга способы решения, а для одного и того же решения - различные версии объясняющих его приемов. Наиболее простые и эффективные варианты не всегда удается "угадать" по единственному примеру, так что оптимизация решателя предполагает более или менее регулярные откаты для модернизации уже созданных групп приемов. Обычно эти откаты имеют локальный характер, а использование компилятора ГЕНОЛОГа делает их в техническом отношении не слишком дорогостоящими.

Рассмотрим несколько примеров анализа траекторий решения задач, в которых не будем предполагать наличия каких-либо ранее созданных приемов. Все эти задачи взяты из задачника решателя, и в качестве упражнения можно рекомендовать проследить по шагам его действия. Приводимые ниже рассуждения дают лишь первое, весьма приблизительное представление о разбиении решений на приемы. Реальное обучение решателя требует учета гораздо большего числа технических подробностей.

Начнем с простейшего примера - решения уравнения

$$\frac{6a+7b}{6a} - \frac{3bx}{2a^2} = 1 - \frac{bx}{a^2-ab}.$$

Первый шаг, который представляется естественным в этой задаче - группировка в левой части всех членов с неизвестной x, а в правой - всех известных членов. Этот шаг сразу подсказывает прием: если обе части числового уравнения имеют неизвестные либо известные слагаемые, то выполняется указанная выше перегруппировка членов. Уровень срабатывания приема можно взять совсем маленьким, например, равным 1. Для усиления стандартизации вводим еще один прием, переводящий неизвестную часть равенства влево, а известную - вправо. Уровень его пусть тоже будет равен 1. Каких-либо соображений об ограничении применения данных приемов не возникает. Однако, следует учитывать, что теперь на уровнях выше первого все

уравнения будут иметь известные слагаемые только в правой части. Поэтому, например, шаблон для усмотрения квадратных уравнений должен иметь вид $ax^2 + bx = c$, вместо привычного $ax^2 + bx + c = 0$.

Итак, получаем уравнение:

$$\frac{bx}{a^2 - ab} - \frac{3bx}{2a^2} = 1 - \frac{6a + 7b}{6a}.$$

Следующий шаг - сложить дробные слагаемые в левой части уравнения. Формулируем соответствующий прием: "если левая часть уравнения имеет дробное слагаемое, то обращаемся к вспомогательной задаче на преобразование ее к виду дроби, после чего выполняем замену". Каких-либо оснований откладывать это действие не видно, так что уровень срабатывания приема снова выбираем равным 1. Целесообразность применения данного приема уже не столь очевидна, как в предыдущем случае. Сложение дробных выражений может привести к очень громоздкому результату, и задача будет заведена в тупик. С другой стороны, сразу привести условия, отделяющие допустимые применения от недопустимых, мы не можем. Чтобы возникли какие-то подсказки на этот счет, нужно все-таки ввести прием таким, как есть, и подождать появления задачи, в которой он будет мешать. Тогда и начнется накопление серии дополнительных эвристических ограничений, которые обеспечат должное управление приемом.

В нашем случае эта работа уже проделана - можно заглянуть в решатель и посмотреть, какие ограничения возникли. Прежде всего, оказалось, что для систем уравнений уровень срабатывания данного приема лучше положить равным не 1, а 3. Если нужно не складывать неизвестные дробные выражения, а обозначить их новыми неизвестными и решить полученную вспомогательную систему, то прием, выполняющий эти действия, успеет сработать. Если уравнение содержит неизвестную, явно выраженную с помощью еще одного уравнения через другие неизвестные, то прием блокируется - лучше (вообще говоря) сначала подставить найденное значение. Блокировка происходит также при наличии в уравнении неизвестного логарифма от суммы с дробным слагаемым. Она заставляет решатель сначала преобразовать указанный логарифм. Наконец, в случае единственной неизвестной применение приема откладывается (как и в случае систем) до уровня 3 при наличии неизвестных логарифмов по разным основаниям. Перечисленные условия относятся к сравнительно редким ситуациям, и таким образом применение рассматриваемого приема оказалось "почти всегда" оправданным.

Введенный нами прием обратился к вспомогательной задаче на сложение дробных выражений. Поэтому временно прерываем анализ цепочки преобразований основной задачи и переходим к рассмотрению выражения

$$\frac{bx}{a^2 - ab} - \frac{3bx}{2a^2}.$$

План наших действий очевиден - сначала нужно разложить на множители знаменатели, а затем выполнить сложение дробей. Оформим эти действия в виде приемов.

Разложение на множители числителей и знаменателей можно считать преобразованием общей стандартизации, предшествующим применению других приемов, относящихся к дробям. Однако, на этапе завершающего редактирования упрощаемого выражения может понадобиться обратное преобразование - иногда раскрывание скобок

приводит к получению более компактной записи. Момент перехода к завершающему редактированию ответа можно помечать вводом специального комментария задачи, и тогда первый прием (разложение на множители) будет срабатывать при отсутствии данного комментария, а второй (попытка упрощающего раскрывания скобок) - при его наличии. Точкой применения приема обращения к разложению на множители можно считать сумму - основание степени, являющейся сомножителем знаменателя (соответственно, числителя) дроби, допуская вырожденный случай степени с показателем единица. В нашем примере единственной такой точкой является выражение $a^2 - ab$.

Прием, применяемый здесь для разложения на множители, состоит в вынесении за скобку общего множителя всех слагаемых. Для его программирования понадобится вспомогательная процедура, находящая наибольший общий делитель двух одночленов.

После разложения на множители знаменателя получаем выражение

$$\frac{bx}{a(a-b)} - \frac{3bx}{2a^2}.$$

Прием, выполняющий сложение дробных выражений, сначала находит общий множитель числителей и общий множитель знаменателей. Они сразу выносятся за скобки, после чего применяется обычное преобразование: числители и знаменатели перемножаются "крест накрест" и складываются. Перед тем, как составить результирующую дробь, предпринимается попытка разложить эту сумму на множители. Последнее действие может показаться излишним - ведь уже имеется прием, который пытается разложить на множители числитель. Однако, тогда понадобятся два цикла сканирования задачи вместо одного - система как бы "забудет" о том, что только что сложила дроби, и лишь после нового цикла рассмотрения задачи натолкнется на указанный числитель. Циклы сканирования задачи в решателе обычно составляют главную часть вычислительного времени. Поэтому лучше объединять в одном и том же приеме как основное действие, так и все сопровождающие его дополнительные - это дает весьма ощутимое ускорение.

В нашем случае задача решается с целью сложить дроби, и каких-либо особых эвристических решающих правил не требуется. Достаточно ввести в прием проверку наличия данной цели. Впрочем, можно ввести ограничение, требующее при сложении нескольких дробных выражений начинать с самых коротких. Иногда это упрощает выкладки.

Возвращаемся к цепочке преобразований уравнения, которое после сложения дробных выражений в левой части приобретает вид

$$\frac{bx(3b-a)}{2(a-b)a^2} = 1 - \frac{6a+7b}{6a}.$$

Это уравнение имеет в левой части дробь; для устранения ее естественно домножить обе части уравнения на знаменатель. Однако, если сформулировать прием подобным образом, то он иногда будет приводить к излишним вычислительным затратам. Целесообразно до исключения знаменателя левой части сложить дробные выражения в правой, известной части. Тогда можно будет сначала вынести за скобку общие множители числителей и знаменателей, и лишь затем избавляться от знаменателей. Результатом преобразований станет дизъюнкция, объединяющая в себе уравнение с

исключенными знаменателями и частные случаи равенства нулю общих множителей числителей:

$$b = 0 \vee 3x(3b - a) = -7a(a - b).$$

Как и прием сложения дробных выражений, данный прием исключения знаменателей объединяет сразу несколько независимых преобразований, ускоряя тем самым процесс вычислений.

Следующий шаг решения задачи - разбор случаев. Прием, используемый для этого, последовательно рассматривает уравнения, соответствующие подслучаям, и затем объединяет полученные ответы связкой "или". Ответ каждого подслучая упрощается отдельно, однако дизъюнкция ответов, вообще говоря, допускает дальнейшее упрощение - какие-то серии корней или целые промежутки могут склеиваться. Поэтому прием должен обращаться к вспомогательной задаче на упрощение полученной дизъюнкции. Более того, этот же прием должен сразу выдать ответ задачи, иначе дизъюнкция, которая им получена, снова вызовет разбор случаев, и система зациклится.

В нашем примере первый подслучай - условие $b = 0$. Оно не содержит неизвестных, и может быть сразу выдано в качестве ответа. Это действие, хотя и очень простое, тоже требует специального приема. Кроме усмотрения того, что условия задачи не содержат неизвестных, данный прием должен обратиться к вспомогательной задаче на упрощение их конъюнкции, и результат упрощения выдать в качестве ответа.

Второй подслучай - уравнение $3x(3b-a) = -7a(a-b)$. Согласно условиям на область допустимых значений (хотя мы и опустили их рассмотрение, но приведенные выше приемы должны были сопровождать все преобразования коррекцией таких условий), правая его часть отлична от нуля. Поэтому уравнение эквивалентно соотношению

$$x = -\frac{7a(a - b)}{3(3b - a)}.$$

Формулировка приема, выполняющего данный переход, очевидна.

Далее следует подстановка найденного значения неизвестной в сопровождающие условия на область допустимых значений и упрощение этих условий. Эти действия требуют специального приема, выполняющего обращение к задаче на редактирование ответа. Полученный ответ

$$a \neq 0, a - b \neq 0, 3b - a \neq 0, x = \frac{7a(b - a)}{3(3b - a)}$$

возвращается приему разбора случаев, объединяющему его с ответом $a \neq 0, b = 0$ первого подслучая и выдающему окончательный ответ.

Разобранный пример оказался совсем простым с точки зрения управления преобразованиями - каждое действие было практически однозначным. Рассмотрим несколько более сложный случай - решение системы уравнений. Будем решать систему:

$$\begin{cases} x^2 + y^2 &= z^2 \\ xy + yz + xz &= 47 \\ (z - x)(z - y) &= 2. \end{cases}$$

Поначалу каких-то соображений однозначного характера о ее преобразованиях не возникает. Однако, есть соображения о том, с чего начинать анализ ситуации. Например, можно попробовать раскрыть скобки в третьем уравнении и посмотреть,

не станут ли очевидными следующие шаги. Это - тоже преобразование, однако не основной задачи на описание, а сопровождающей ее задачи на исследование, в которой накапливаются общие следствия посылок и условий. Здесь мы имеем сразу два приема: первый принимает решение о переходе к выводу следствий в задаче на исследование, второй - выводит из третьего уравнения следствие, получаемое раскрыванием скобок. Впрочем, в действительности эти приемы в решателе переставлены местами - раскрывание скобок выполняется сразу же. Случаи, когда оно может повредить, отслеживаются приемами, срабатывающими на меньших уровнях. Создавая прием для раскрывания неизвестных скобок в уравнениях, мы не должны сразу же предусматривать какие-то ограничения на его применение. Если они необходимы, то проявятся позднее, при анализе других задач. Так или иначе, получаем в списке посылок задачи на исследование первые два уравнения, сопровождаемые уравнением $xy - xz - yz + z^2 = 2$. Теперь становится видно, что при сложении второго и третьего уравнений уничтожаются сразу два неизвестных слагаемых в левой части. Получается уравнение $2xy + z^2 = 49$ - следствие двух исходных уравнений. Прием, выполняющий это действие, можно несколько обобщить - чтобы он искал линейную комбинацию двух уравнений, позволяющую устранить сразу два неизвестных слагаемых. Следующий естественный шаг - сложить первое уравнение с полученным следствием, чтобы исключить неизвестную z. Формулировка соответствующего приема несложна. После сложения уравнений имеем следствие $x^2 + y^2 + 2xy = 49$. Очередной шаг очевиден - представить левую часть как полный квадрат: $(x + y)^2 = 49$. Прием, выполняющий это действие, обращается к вспомогательной задаче на разложение левой части уравнения на множители. Если такое разложение удается, то есть вероятность, что полученное уравнение будет полезно для дальнейшего (например, чтобы поделить два уравнения, сократив неизвестный множитель). Разумеется, нужно принять меры, чтобы прием не пытался разложить на множители левую часть уравнения, полученного после раскрывания скобок, и обратно. Для этого можно использовать специальные комментарии к уравнениям, блокирующие "обратный ход". Следующий шаг - решение простейшего степенного уравнения. После него появляется дизъюнкция $x + y = 7 \ \vee \ x + y = -7$. Обычно получение дизъюнкции в задаче на исследование означает, что следует вернуться в исходную задачу на описание и предпринять разбор случаев. В нашей ситуации нужно, кроме того, учесть, что найденная дизъюнкция эквивалентна, при наличии первых двух уравнений, третьему. Это позволяет при разборе случаев исключить третье уравнение. Таким образом, цикл вывода следствий из уравнений завершился, и далее решаем две независимых системы. Ограничимся рассмотрением первой из них -

$$\begin{cases} x + y &= -7 \\ x^2 + y^2 &= z^2 \\ xy + yz + xz &= 47. \end{cases}$$

Очередной шаг состоит в преобразовании первого уравнения к виду $x = -(y + 7)$. Прием, реализующий это шаг, можно было бы представить как обращение к вспомогательной задаче, разрешающей одно из уравнений системы для исключения неизвестной. На первый взгляд, такое обобщение нашего шага может показаться естественным. Однако, при рассмотрении последующих примеров выяснилось бы, что данный прием почти всегда вреден - если произвольно выбрать какое-то уравнение системы и разрешить (пусть даже успешно) относительно одной из неизвестных, то чаще всего возникает такое усложнение системы, после которого решить ее стано-

вится весьма затруднительно. Сравнивая случаи, где выражение одной неизвестной через другие полезно, со случаями, где оно вредно, можно было бы создать несколько приемов, применяемых в очень специальных ситуациях. Для нашей задачи вводим прием, выражающий неизвестную из уравнения, если оно линейно по всем своим неизвестным.

Далее подставляем найденное выражение для x через y и переходим к решению системы относительно y, z. Это делается отдельным приемом, который должен создать вспомогательную задачу для двух неизвестных и обратиться к ее решению. Завершающая обработка ответа допускает разные варианты. Можно получить ответ на вспомогательную задачу, объединить его с уравнением $x = -(y + 7)$ и упростить результат. Однако, если при решении вспомогательной задачи происходил разбор случаев и ответ на нее имеет вид дизъюнкции, то при упрощении снова понадобится разбор случаев. Поэтому в решателе реализован другой способ - уравнение $x = -(y + 7)$ передается вспомогательной задаче через ее технические структуры данных и извлекается оттуда при редактировании ответа для каждого подслучая. Тогда после решения вспомогательной задачи какой-либо дополнительной обработки ответа не потребуется.

После перехода к неизвестным y, z выполняются простые преобразования, связанные с общей стандартизацией вида уравнений и с раскрыванием скобок. Они реализуются простыми приемами, на которых можно сейчас не останавливаться. В результате возникает система $14y+2y^2-z^2 = -49, 7y+7z+y^2 = -47$. Легко заметить, что члены с неизвестной y в обоих уравнениях пропорциональны, и после вычитания из первого уравнения удвоенного второго остается уравнение с единственной неизвестной z. Прием, усматривающий возможность получить уравнение с единственной неизвестной за счет линейной комбинации уравнений, практически не требует каких-либо дополнительных ограничений на целесообразность срабатывания. В результате получаем систему $14y+2y^2-z^2 = -49, -14z-z^2 = 45$. Дальнейшие действия очевидны, и мы их разбирать не будем.

Разобранные примеры продемонстрировали два режима работы решателя - эквивалентные преобразования уравнения в первом случае и вывод следствий для получения дополнительной информации о неизвестных во втором. Первый режим, априори, требует большой осмотрительности при выборе каждого действия, так как неудачное преобразование может завести задачу в тупик. В действительности, однако, для многих областей наблюдается явление устойчивости: если в целом следовать некоторым несложным представлениям о том, какие преобразования упрощают задачу, то выбор конкретного порядка их выполнения несущественен - в любом случае за разумное число шагов получается один и тот же ответ. Это и позволяет в таких областях "избавляться от перебора". Второй режим позволяет исключать перебор в его классическом виде рассмотрения дерева задач, сводя поиск решения к рассмотрению расширяющегося списка посылок одной и той же задачи. Так как приемы, выводящие следствия, снабжены решающими правилами, отсекающими малоперспективные (например, чрезмерно громоздкие) с точки зрения эксперта следствия, то через определенное время наступает полное исчерпание разумных следствий, и процесс обрывается. Процедура решения задачи, вместо перебора, приобретает здесь характер "логического замыкания" исходных данных.

Рассмотрим пример, в котором возникает еще одна разновидность "логического ре-

жима". Будем решать задачу на доказательство неравенства

$$0 \leq a^4 - 2a^3b + 2a^2b^2 - 2ab^3 + b^4.$$

В таких задачах часто помогает прием, использующий неравенство для среднего арифметического и среднего геометрического. Удобнее переформулировать его в виде приема, выделяющего в оцениваемой сумме квадрат суммы или квадрат разности. В нашей задаче можно заметить, что слагаемые a^2b^2, b^4 и $-2ab^3$ представляют собой квадрат разности величин ab, b^2. Поэтому, если доказать неравенство $0 \leq a^2b^2 + a^4 - 2a^3b$, полученное отбрасыванием данных слагаемых, то задача будет решена. Применяя к последнему неравенству тот же прием, получаем искомое доказательство.

Заметим, что выбор группы слагаемых, образующих квадрат суммы или разности, выполняется неоднозначно. Например, на первом шаге можно было бы выделить группу $2a^2b^2, a^4, b^4$ и сразу завести задачу в тупик. По этой причине может показаться, что прием выделения оцениваемой группы слагаемых требует какого-то сильного управления, без чего возникнет трудоемкий перебор. Однако, данный прием обычно сильно упрощает правую часть неравенства, так что длина цепочек неравенств, возникающих при доказательстве, невелика. Кроме того, часто появляются тупиковые ситуации, в которых прием неприменим. Поэтому реальный перебор оказывается невелик, и особых проблем с принятием решения не возникает. Здесь возникает режим "ограниченного перебора", у которого ограничения обусловлены не какими-то специальными решающими правилами, а просто тем фактом, что сложность задачи при каждом шаге сильно уменьшается. Этот режим используется в решателе для разных разделов (например, вычисление пределов и интегрирование), и обычно дает очень быстрое получение ответа.

Перейдем к примерам из других разделов. Рассмотрим сначала простую геометрическую задачу на вычисление. В параллелограмме $ABCD$ из точки N пересечения диагоналей проведены перпендикуляры NF, NE к сторонам AB,AD. Длины этих перпендикуляров равны, соответственно, p, m. Угол BAD равен a. Найти длины x, y диагоналей AC, BD и площадь параллелограмма z.

Схема решения геометрических задач на вычисление напоминает схему решения систем уравнений - происходит вывод следствий из посылок и условий до тех пор, пока не возникают либо равенства для значений неизвестных, либо уравнения, из которых значения неизвестных извлекаются чисто алгебраическими методами. В начале процесса вывода порождаются совсем простые утверждения, которые можно рассматривать как логическое представление чертежа задачи. В нашем примере таковыми являются утверждения о параллельности прямых AB, CD и AD, BC. При решении геометрической задачи последовательность рассуждений допускает множество вариаций. Итоговая картина при этом обычно оказывается не очень чувствительной к ее конкретной версии. Приведем для нашего примера одну из таких возможных последовательностей.

Прежде всего, замечаем, что в задаче нужно найти площадь параллелограмма и что к стороне AD проведен перпендикуляр NE. Для получения высоты, длина которой участвует в формуле площади, продолжаем перпендикуляр до пересечения с противоположной стороной в точке G, и одновременно выписываем соотношение $z = l(AD)l(EG)$. Эти действия оформляем в виде отдельного приема. Основаниями для его применения служат указанные выше признаки - упоминание в задаче о

площади параллелограмма и "недоведенный" до конца перпендикуляр к одной из сторон.

Продолжая общий анализ чертежа, выводим из равенства $\angle(BAD) = a$ соотношения для четырех остальных углов параллелограмма. Основанием для применения этого приема является упоминанение в задаче хотя бы одного из углов параллелограмма.

Аналогичным образом, выводим равенство длин противоположных сторон параллелограмма $l(BC)$ и $l(AD)$. Основанием применения приема служит упоминание в задаче одной из этих длин.

Так как выделена точка E пересечения диагоналей параллелограмма, замечаем, что диагонали делятся в ней пополам: $l(CN) = l(AN)$, $l(DN) = l(BN)$. Еще один прием отмечает, что точка E не просто лежит на прямой BD, но является точкой отрезка BD. Основанием для его срабатывания служит наличие в посылках задачи равенства $l(DN) = l(BN)$. Этот же прием устанавливает, что точка N лежит на отрезке AC.

Так как точка N находится на отрезке AC, причем длина отрезка и расстояния от N до его концов упоминаются в задаче, то выводим соотношение равенства длины отрезка сумме длин подотрезков: $x = 2l(AN)$. Аналогично, выводим $2l(BN) = y$.

Далее выводится условие принадлежности точки N отрезку EG - проведенной высоте параллелограмма. Основаниями являются принадлежность точки N диагонали и параллельность сторон. Отсюда, аналогично предыдущему, получается следствие $l(EG) = 2m$ - прием о равенстве длины отрезка сумме длин подотрезков срабатывает уже в третий раз.

Так как угол BAD и высота $l(EG)$ теперь известны, можно выписать тригонометрическое соотношение $\sin(a)l(AB) = 2m$, в котором единственный неизвестный числовой параметр - длина отрезка AB. Выписывание таких соотношений (возможно, не на самых малых уровнях сканирования задачи) представляется разумным, так как постепенно доопределяются новые параметры чертежа.

После того, как предыдущее соотношение занесено в посылки задачи, оно преобразуется к виду $l(AB) = 2m/\sin(a)$. Здесь работает прием, разрешающий линейное соотношение относительно числового параметра, определенного через ссылки на точки.

Как только оказалась введена в рассмотрение длина стороны AB, срабатывает уже упоминавшийся выше прием, выписывающий равенство длин сторон AB,CD.

Далее вводится высота параллелограмма CH, проведенная к продолжению стороны AB. Основанием для этого действия служит то, что, во-первых, длины отрезков AN,NC пропорциональны, а длина p высоты FN - известна. Отсюда прием выводит следствие $l(CH) = 2p$. Во-вторых, основанием для срабатывания является наличие известного угла CBA, который мог бы позволить выразить впоследствии через $l(CH)$ какие-то новые параметры чертежа.

Условие перпендикулярности прямых CH и AB преобразуется в условие параллельности прямых CH и FN - чтобы в дальнейшем иметь дело с классами параллельных прямых, выбирая в каждом таком классе единственного представителя.

Снова применяется прием, выписывающий тригонометрическое соотношение

$$\sin(a)l(AD) = 2p.$$

Однако, это другой прием, срабатывающий на меньшем уровне, чем рассмотренный выше. Оснований для его срабатывания больше - параметр $l(AD)$ к этому моменту уже встречается в задаче. Введенное соотношение сразу преобразуется к виду $l(AD) = 2p/\sin(a)$.

Теперь начинает срабатывать прием, подставляющий найденное значение $l(AD)$ во все посылки, где этот параметр встречается. В частности, таким образом из посылки $z = 2ml(AD)$ получаем фрагмент ответа $z = 4mp/\sin(a)$. Накопление информации о параметрах чертежа "само собой" привело к нахождению искомой площади. Разумеется, существенную роль при этом сыграли приоритеты на получение соотношений, устанавливающих связь данных и искомых параметров. Вывод следствий имеет характер "встречного распространения" цепочек зависимостей - от известных и от неизвестных величин.

К текущему моменту определились длины сторон параллелограмма и его углы. Поэтому для нахождения неизвестных диагоналей и завершения решения остается воспользоваться теоремой косинусов. Основания для срабатывания приема - известны длины двух сторон треугольника и угол между ними, а длина третьей стороны связана с неизвестными задачи каким-либо соотношением.

Перечисленные выше действия в действительности совпадают с действиями решателя. Таким образом, видно, что даже после накопления большой базы приемов удается избежать чрезмерного количества выводимых следствий. Это достигается тенденцией вводить при обучении как можно более сильные ограничения на срабатывания, пропускающие лишь действительно разумные шаги.

Сразу заметим, что при обучении решателя геометрическим задачам было допущено одно отклонение от "человеческих" стандартов. Вместо того, чтобы в цикле предварительного анализа чертежа определять, какие его элементы можно выразить через другие, и лишь затем выписывать нужные соотношения, решатель вводит эти соотношения сразу. Однако, за исключением редких случаев, он никак их не преобразует, а использует лишь для установления самого факта взаимной выразимости параметров - как своего рода "граф" взаимосвязей. На быстродействии компьютерной системы это сказывается мало, но у пользователя, анализирующего ход решения по шагам, складывается впечатление об избыточной громоздкости накапливаемой информации. Впрочем, за счет дальнейшего усиления решающих правил, с данным явлением можно достаточно эффективно бороться. Иногда оно и вовсе незаметно.

Следующая задача - на качественное исследование поведения функции $f(x) = x^x$ с помощью пределов и производных. Она решается по схеме, аналогичной только что разобранному геометрическому примеру. Решение заключается в последовательном выводе следствий, характеризующих функцию f, и в отборе тех из них, которые целесообразно включить в итоговый список. Прежде всего применяется прием, находящий область определения функции. Он выписывает условия на область допустимых значений x и обращается к вспомогательной задаче на упрощение класса таких значений. Выводится следствие $\mathrm{Dom}(f) = (0, \infty)$. Следующий шаг - вычисление производной. Получаются следствия $g(x) = (1 + \ln x)x^x$; "Производная(f, g)". Снова применяется прием, определяющий область определения введенной в рассмотрение функции g. После того, как явно указаны области определения f, g, вводится посылка, указывающая множество точек, где производная не определена либо не вычислена: $\mathrm{Dom}(f) \setminus \mathrm{Dom}(g) = \emptyset$. Следующий шаг - срабатывание приема, обращающегося к вспомогательной задаче для отыскания корней производной и вводяще-

го следствие $\mathrm{roots}(g, \mathrm{Dom}(g)) = \{1/e\}$. Чтобы анализировать поведение функции в точках, где ее производная определена и отлична от нуля, эта область явно находится и разбивается на промежутки. Соответствующий прием обращается к вспомогательной задаче, осуществляющей такое разбиение, после чего появляется посылка "областьроста$(f, (0, 1/e) \cup (1/e, \infty))$", распадающаяся на утверждения "областьроста$(f, (0, 1/e))$", "областьроста$(f, (1/e, \infty))$". Используя информацию о промежутках монотонности и анализируя знак производной на их стыке (точка $1/e$), следующий прием выводит утверждения "возрастает$(f, (1/e, \infty))$", "убывает$(f, (0, 1/e))$". Далее применяется прием, анализирующий знаки функции f на интервале монотонности. Он выводит следствия о числе корней на интервале: $\mathrm{card}(\mathrm{roots}(f, (0, 1/e))) = 0$, $\mathrm{card}(\mathrm{roots}(f, (1/e, \infty))) = 0$, немедленно преобразуемые к виду $\mathrm{roots}(f, (0, 1/e)) = \emptyset$, $\mathrm{roots}(f, (1/e, \infty)) = \emptyset$. Следующий прием усматривает экстремум на стыке промежутков монотонности -

$$\mathrm{Extr}(f, \frac{1}{e}, \frac{1}{\exp \frac{1}{e}}, \min).$$

На этом поток вывода следствий исчерпывается, и выдается ответ, в который отбираются: соотношение, определяющее функцию f, информация об экстремуме, о промежутках монотонности, об области определения функции f и о числе ее корней на промежутках монотонности.

В этом примере управляющая компонента приемов была почти вырожденной - по существу, совокупность приемов составляла алгоритм исследования функции. Однако, разбиение такого алгоритма на отдельные почти не связанные друг с другом фрагменты - приемы - предоставляет несомненные удобства для последующего пополнения его все новыми и новыми элементами, ориентированными на различные специальные случаи.

Еще один пример, решаемый по схеме вывода следствий - из аналитической геометрии. Пусть стороны треугольника заданы уравнениями $7x + y - 2 = 0, 5x + 5y - 4 = 0, 2x - 2y + 5 = 0$. Нужно найти координаты точки внутри треугольника, равноудаленной от первых двух прямых и отстоящей от третьей на расстояние $\frac{3\sqrt{2}}{4}$. В аналитической геометрии, а в особенности в таких разделах, как теория вероятностей и физика, обычной математической символики для полной формулировки задачи оказывается уже недостаточно. Поэтому пошаговому анализу решения здесь предшествует анализ возможных вариантов логической формализации условия и отбор наиболее удобных вариантов. В нашем примере обозначения возьмем из логического языка решателя. Пусть вершины треугольника обозначены A, B, C, а прямоугольная система координат, относительно которой берутся уравнения - K. Сами уравнения можно записать тогда, например, в следующем виде: "коорд(прямая(AB),$K) = \mathrm{set}_{xy}(x -$ число $\&\ y -$ число $\&\ 7x + y - 2 = 0)$"; "коорд(прямая(AC),$K) = \mathrm{set}_{xy}(x -$число $\&\ y -$число $\&\ 5x + 5 - 4 = 0)$"; "коорд(прямая(BC),$K) = \mathrm{set}_{xy}(x -$ число $\&\ y -$ число $\&\ 2x - 2y + 5 = 0)$". Искомую точку обозначим D; условие ее принадлежности треугольнику запишем в виде $D \in$ фигура(ABC). Равноудаленность точки от первых двух прямых представляем записью "расстдопрямой$(D$,прямая$(AB))=$ расстдопрямой$(D$,прямая$(AC))$". Указываем расстояние до третьей прямой: "расстдопрямой$(D$,прямая$(BC)) = \frac{3\sqrt{2}}{4}$". Перечисленные утверждения образуют список посылок задачи, т.е. то, что дано. Условием ее служит равенство $z =$ коорд(D, K), причем переменная z является неизвестной задачи. По постановке задачи, в выражение для z не должны входить обозначения A, B, C, D, K, хотя они и появляются в посылках задачи, а следовательно, формально являются "известными". Это специально оговаривается в целевой установке задачи.

Процесс решения задачи выглядит как вывод следствий из объединенного списка посылок и условий. Приведем цепочку выводов, реализуемую решателем. Хотя она, быть может, и не оптимальна, однако для новичка вполне допустима.

Прежде всего, вводятся координаты точек A, B, C: "коорд(A,K)" $= (a, b)$; "коорд(B, K)" $= (c, d)$; "коорд(C,K)" $= (e, f)$. Основанием для такого действия служит то, что точки встречаются в обозначениях прямых, для которых известны уравнения. Следующий шаг - ввод координатного набора для неизвестной z: $z = (g, h)$. Основанием является то, что в задаче рассматривается расстояние от точки D до прямой, уравнение которой известно. Для всех введенных новых параметров $a, b, \ldots, h$ регистрируются посылки, указывающие, что параметры - числовые. Подставляя координаты точки A в уравнение прямой AB, получаем соотношение $b + 7a - 2 = 0$. Его преобразуем к виду $b = 2 - 7a$ и подставляем b во все остальные посылки задачи. Аналогичным образом, выводим соотношение $d + 7c - 2 = 0$ и преобразуем к виду $d = 2 - 7c$. Подставляя координаты точки A в уравнение прямой AC, находим $a = 1/5$. Все эти действия выполняются простыми приемами, срабатывающими без ограничений. Подставляя значение a в уравнение для b, находим $b = 3/5$. Для точек B, C выполняем аналогичные действия; в итоге получаем $c = -1/16$, $d = 2 + 7/16$, $e = -17/20$, $f = 33/20$.

Определив координаты точек A, B, C (быть может, они для дальнейшего и не понадобятся), решатель переходит к выводу соотношений для искомых координат g, h. Так как уравнение прямой AC известно, можно воспользоваться формулой для расстояния от точки до прямой и получить соотношение

$$50(\text{расстдопрямой}(D, \text{прямая}(AC)))^2 = 50gh + 25g^2 + 25h^2 - 40g - 40h + 16.$$

Основанием для этого действия служит то, что указанное расстояние уже упоминается в задаче, а уравнение прямой известно. Выражаем квадрат расстояния:

$$(\text{расстдопрямой}(D, \text{прямая}(AC)))^2 = \frac{50gh + 25g^2 + 25h^2 - 40g - 40h + 16}{50}.$$

Аналогичным образом, для расстояния от D до прямой AB получаем:

$$(\text{расстдопрямой}(D, \text{прямая}(AB)))^2 = \frac{14gh + 49g^2 + h^2 - 28g - 4h + 4}{50}.$$

С учетом посылки задачи, указывающей равенство данных расстояний, выводим соотношение

$$\frac{50gh + 25g^2 + 25h^2 - 40g - 40h + 16}{50} = \frac{14gh + 49g^2 + h^2 - 28g - 4h + 4}{50}.$$

Далее переходим к расстоянию от D до прямой BC:

$$(\text{расстдопрямой}(D, \text{прямая}(BC)))^2 = \frac{-8gh + 4g^2 + 4h^2 + 20g - 20h + 25}{8}.$$

В это равенство подставляем значение расстояния, данное в формулировке задачи:

$$\frac{9}{8} = \frac{-8gh + 4g^2 + 4h^2 + 20g - 20h + 25}{8}.$$

Начинается цепочка общей стандартизации последних уравнений. После устранения знаменателей и сокращения они преобразуются к виду: $5g - 2gh - 5h + g^2 + h^2 = -4$, $g + 3h + 2g^2 - 3gh - 2h^2 = 1$.

Теперь начинается учет условия принадлежности точки D треугольнику. Здесь-то и используются найденные ранее координаты вершин. Условие расположения точки D по ту же сторону от прямой AB, что и точка C, дает неравенство $h + 7g - 2 \leq 0$. В случае прямой AC получаем $0 \leq 5g + 5h - 4$. В случае прямой BC - $0 \leq 2g - 2h + 5$.

Далее решатель усматривает систему из двух приведенных выше уравнений относительно числовых параметров g, h и решает ее вне общего контекста, учитывая также неравенства для g, h. В результате получается $g = 0, h = 1$, $z = (0, 1)$, и задача решена.

Процесс вывода следствий похож на решение задачи по элементарной геометрии, однако управление приемами здесь существенно проще.

Глава 5

Алгоритмический язык ЛОС

5.1 Структуры данных и общая схема организации программы на языке ЛОС

Программы приемов решателя записываются на языке ЛОС (Логический Описатель Ситуаций) и реализуются интерпретатором этого языка. Язык ЛОС по своему уровню сопоставим с ПРОЛОГом, однако специально адаптирован к изложенной выше схеме сканирования задач и имеет много принципиальных отличий от ПРОЛОГа. Для логической системы он является языком низкого уровня, и запись на нем приемов "вручную"предпринимается крайне редко. Фактически приемы записываются на языке более высокого уровня, ГЕНОЛОГе, и далее компилятор ГЕНОЛОГа создает на основе этой записи исполняемую ЛОС-программу приема. Вместе с тем, основу языка ЛОС составляют около 200 предикатов и операций, которые могут рассматриваться не только как алгоритмический, но и как вполне развитый логический язык для описаний ситуаций в логико-сетевых структурах данных. Поэтому в тех случаях, когда речь идет не о работе с объектами предметной области, а о работе с объектами структурного уровня (иными словами, когда предметной областью являются сами структуры данных), логические уровни ЛОСа и ГЕНОЛОГа, по существу, совпадают, и программирование приемов непосредственно на ЛОСе становится вполне оправданным.

В языке ЛОС выделяются следующие основные типы объектов, обрабатываемых программой:

а) Символы переменных и логические символы. Максимально возможное число переменных и логических символов определяется конкретной версией интерпретатора; используемая в настоящее время версия допускает применение $2^{19} - 1$ логических символов и стольких же переменных. Можно вводить, удалять либо изменять название ранее введенных логических символов. Каждое название логического символа представляет собой произвольную последовательность символов расширенного алфавита (допускаются русские и латинские буквы,цифры и пр.), длина которой не более 24. Собственно в программных файлах ЛОСа логические символы представлены при помощи своих номеров, а названия их хранятся в отдельном файле, что позволяет изменять эти названия, не изменяя программ.

б) Термы, построенные из переменных и логических символов (заголовок операции - произвольный логический символ, число операндов - любое). Терм длины 1 отож-

дествляется с набором длины 1 (см.пункт в) и отличается от входящего в него логического символа.

в) Наборы $(a_1, \ldots, a_n)$ ранее определенных допустимых объектов.

г) Вхождения символов в термы и разрядов в наборы.

Заметим, что для скобок не предусмотрены специальные логические символы, а терм $\varphi(a_1 \ldots a_n)$ не является набором входящих в него логических символов и скобок - в структурах данных интерпретатора с ЛОСа скобки представлены неявным образом (указаны длины подтермов и ссылки на внешние операции). Тем не менее, во многих случаях терм можно рассматривать как последовательность составляющих его логических символов и символов переменных с опущенными скобками, применяя при этом операторы, предназначенные для работы с наборами. Для обозначения пустого набора выделен логический символ "пустоеслово", воспринимаемый рядом операторов языка как набор длины 0.

Все логические символы пронумерованы натуральными числами. При использовании их в качестве заголовков операторов языка ЛОС выделяются непосредственно реализуемые логические символы (номера от 1 до 240) и программно реализуемые логические символы. С непосредственно реализуемыми символами связаны соответствующие процедуры интерпретатора; выполнение операторов с программно реализуемыми заголовками происходит при помощи программ языка ЛОС.

Представление чисел в решателе осуществляется четырьмя различными способами. С одной стороны, для быстрых выкладок с целыми числами диапазона от -259 до 65535 используется кодирование их логическими символами (логический символ "0" имеет номер 260). С другой стороны, отличное от цифры десятичное число $a_1 \ldots a_n$, $a_{n+1} \ldots a_{n+m}$ кодируется набором $(a_1 a_2 \ldots a_n, a_{n+1} \ldots a_{n+m})$, где запятая является логическим символом (изменение знака числа происходит путем добавления либо удаления логического символа "минус"в начале набора), а цифра отождествляется с обозначающим ее логическим символом. При работе с термами применяется представление указанного числа в виде "величина $(a_1 \ldots a_n, a_{n+1} \ldots a_{n+m})$"(отрицательного числа - в виде "минус (величина $(a_1 \ldots a_n, a_{n+1} \ldots a_{n+m})$)"; цифры - в виде однобуквенного терма). Наконец, для приближенных вычислений с математическим сопроцессором и для вычислений с применением обычной 32-битной машинной арифметики используются форматы "вещественное с плавающей запятой", "компалексное с плавающей запятой" (64-битное представление как для вещественной, так и для мнимой части), "целое со знаком" (32 бита) и "длинное целое со знаком" (массив 32-битных значений, имеющий переменную длину). Числа в указанных форматах кодируются специальным образом: первое - набором длины 2, второе - набором длины 4, третье - набором длины 1, четвертое - набором переменной длины. Эти наборы несколько отличаются от обычных наборов ЛОСа, и работать с ними нужно только при помощи операторов, специально предназначенных для машинной арифметики.

Программе доступна совокупность данных, образованная, с одной стороны, значениями ее переменных (входные данные и вспомогательные объекты, формируемые программой), а с другой стороны, некоторая совокупность внешних данных, описывающих цепь задач, сложившуюся в процессе решения предложенной системе задачи. Как уже отмечалось в главе 3, задача Z представляет собой набор $(p_0, p_1, \ldots, p_n)$, где p_0 - логический символ, определяющий тип задачи; p_1 - набор посылок задачи; p_2 - набор весов посылок (логических символов от "0"до "20"); p_3 - набор наборов объектов, представляющих собой комментарии к посылкам, и т.д. Цепь задач представляет

собой упорядоченную последовательность задач $(Z_0, Z_1, \ldots, Z_N)$, где Z_0 - исходная задача; Z_N - текущая решаемая задача; Z_{i+1} - подзадача задачи Z_i $(i = 1, \ldots, N-1)$. Эта последовательность сама не образует допустимого объекта (набора задач) и обращение к ней осуществляется через специальные операторы языка. Такие операторы, в частности, позволяют определить по каждому элементу Z_i цепи задач его текущий m_i и максимальный M_i уровни (логические символы от "0"до "20"). Исходная задача Z_0 имеет фиктивный характер и создается автоматически при включении системы. Она представляет собой задачу на исследование с единственной фиктивной однобуквенной посылкой "вход"; при сканировании этой посылки запускается написанная на языке ЛОС программа интерфейса, позволяющая сформировать фактически решаемую задачу Z_1. Список общих комментариев к посылкам задачи Z_0 используется в качестве "доски объявлений", доступной любой процедуре системы. Главным образом, эта доска объявлений используется процедурами интерфейса.

Помимо указанных здесь основных типов данных, предусмотрены различные типы данных внешнего характера (файлы с древовидными информационными конструкциями, файлы программ ЛОСа и др.). Более подробно эти типы данных описываются ниже, при рассмотрении операторов языка ЛОС. Для работы с внешними данными используются ссылки, представленные посредством данных основных типов (как правило, наборы логических символов). Перечень внешних типов данных системы является, по существу, открытым, так как организация интерпретатора языка позволяет сравнительно легко подключать дополнительные непосредственно реализуемые операторы, работающие с данными нового типа.

Программа решателя организована по энциклопедическому принципу и распадается на программы отдельных логических символов. Программа логического символа φ имеет древовидную структуру и состоит из совокупности пронумерованных натуральными числами текстов, называемых фрагментами этой программы. Каждый фрагмент программы логического символа φ представляет собой последовательность $P_1, \ldots, P_n$ термов специального вида, называемых операторами языка. В большинстве случаев эти термы представляют собой просто утверждения логического языка, значениями переменных которых служат данные указанных выше типов.

Два специальных типа операторов, а именно "ветвь(N)" и "иначе(N)", где N - натуральное число, называются операторами перехода (для краткости скобки в операторах перехода опускаются, т.е. применяется запись "ветвь N", "иначе N"). Если в i - м фрагменте встречается оператор "ветвь j" либо "иначе j", то говорим, что этот фрагмент имеет ссылку на j - й фрагмент. Граф ссылок между фрагментами программы является деревом с корнем в 1-м фрагменте; 1-й фрагмент называется также началом программы символа φ. Оператор "иначе N"располагается в фрагменте только после некоторого другого оператора, не являющегося оператором перехода. Во избежание путаницы сразу заметим, что в интерфейсе программного редактора ЛОСа применяется не глобальная, как сказано выше, а локальная нумерация фрагментов программы: ссылки из каждого фрагмента на подфрагменты пронумерованы там независимым образом последовательными натуральными числами (по мере появления их в тексте фрагмента), начинающимися с 1. Указанная выше глобальная нумерация фрагментов введена здесь лишь для удобства описания языка; фактически в файле программы ЛОСа ссылка из фрагмента на подфрагмент задается смещением этого подфрагмента в файле.

В операторах используется специальный класс подтермов, называемых операторными выражениями. В большинстве случаев эти подтермы суть некоторые выраже-

ния логического языка. Входящие в операторные выражения переменные обозначаются в программе ЛОСа посредством буквы "x"с номером: x1, x2, ... По определению, каждое однобуквенное выражение является операторным. Синтаксис операторных выражений $\varphi(t_1 \dots t_n)$ определяется независимо для каждого символа φ. Непосредственно реализуемые символы φ рассматриваются в следующем разделе данной главы; в случае программно реализуемого символа все операнды $t_1, \dots, t_n$ сами должны быть операторными выражениями (число n может варьироваться, при условии, что программа способна самостоятельно определить его по заведомо доступным ей данным). Аналогичным образом, синтаксис операторов $\varphi(t_1 \dots t_n)$, $n \geq 0$, определяется независимо для каждого символа φ, причем в случае программно реализуемого символа φ все $t_1, \dots, t_n$ суть операторные выражения. При определении оператора вида $\varphi(t_1 \dots t_n)$ указывается разбиение множества его свободных переменных на входные и выходные переменные; выходная переменная имеет (кроме случаев $\varphi \in \{$ "и", "или", "альтернатива"$\}$) ровно одно вхождение, являющееся вхождением некоторого операнда $t_i (i = 1, \dots, n)$. В некоторых случаях допускаются различные варианты такого разбиения, причем интерпретатор ЛОСа при реализации оператора выбирает то из них, для которого значения всех входных переменных уже определены программой, а всех выходных - не определены.

При обращении к программе логического символа φ, а также на каждом шаге выполнения этой программы выделяется некоторый набор переменных, значения которых считаются определенными , причем программа способна самостоятельно отличать их от прочих переменных.

Обращения к программе символа φ бывают следующих трех типов:

1) Обращение в процессе сканирования текущей решаемой задачи Z_N (см. описание процесса сканирования задачи в главе 4). Если φ - тип задачи и произошло обращение к процедуре, объединяющей приемы решения задач типа φ без привязки к конкретному вхождению логического символа в задачу (см. пункт 1 описания процедуры сканирования), то определены значения переменных $x1$ (задача Z_N) и $x5$ (текущий уровень m_N этой задачи). Если в задаче выделено конкретное вхождение символа φ (пункт 3 описания процедуры сканирования), то определены $x1 = Z_N$, $x2$ - данное вхождение символа φ в текущую посылку либо условие задачи, $x3$ - вхождение текущей посылки либо условия задачи в список посылок либо условий (при отсутствии такового - в саму задачу), $x4$ равно 0, если просматривается посылка задачи, и равно 1 в противном случае, $x5 = m_N$. Таким образом, тройка $(x2, x3, x4)$ определяет координаты вхождения в задачу выделенного символа φ. Если произошло обращение к процедуре символа, являющегося названием типа задачи (пункт 1 процедуры сканирования), то значения $x2, x3, x4$ равны логическому символу 0.

2) Обращение при выполнении программно реализуемого оператора либо нахождении значения операторного выражения $\varphi(t_1 \dots t_n)$. В этом случае определены все переменные $xi, 1 \leq i \leq n$, для которых t_i не является выходной переменной оператора. Остальные переменные $xi, i \in \{1, \dots, n\}$, считаются не определенными и должны оставаться таковыми вплоть до момента присвоения им специальными заключительными операторами результатов выполнения программы перед выходом из нее.

3) Приемы решения задач часто используют вспомогательные процедуры, распадающиеся на систему независимых подпроцедур, связанных с различными логическими символами. Такие процедуры обеспечивают получение информации справочного характера, связанной с соответствующими логическими символами (напри-

мер, определяют арность символа; находят область допустимых значений операции с заданным заголовком, и т.п.). Эти процедуры называем далее справочниками. Каждая из них обозначается специальным логическим символом, отличным от символа "0" - типом данного справочника. Тип справочника ψ рассматривается как характеристика обращения к программе символа φ при попытке получить информацию, связанную с этим символом. Чтобы определить, что имеет место именно такой тип обращения, используется вспомогательный оператор "обращение(ψ)", истинный лишь в этом случае. Значения переменных $x1, \ldots, xn$, определенных при обращениях к справочнику типа ψ, задаются осуществляющим запрос операторным выражением "справка($\psi\varphi t_1 \ldots t_n$)"($x_i$ получает значение операторного выражения t_i).

Операторы языка ЛОС делятся на проверочные, перечисляющие и смешанного проверочно-перечисляющего типа. Проверочный оператор, при указании значений его входных переменных, либо принимает значение "ложь", либо однозначным образом определяет значения выходных переменных и принимает значение "истина". Перечисляющий оператор, при определении значений его входных переменных, генерирует некоторую конечную последовательность (возможно,пустую) наборов значений выходных переменных, принимая после каждой выдачи очередного набора значение "истина", после чего принимает значение "ложь". Операторы смешанного типа функционируют либо в проверочном, либо в перечисляющем режиме, в зависимости от конкретных значений их входных переменных. Оператор "ветвь N"по определению считается перечисляющим. Для возвращения к перечисляющему оператору и получения от него очередного набора значений выходных переменных при реализации программы создается стэк перечисляющих операторов ($Q_1 \ldots Q_m$), которые относятся, вообще говоря, к различным фрагментам программы символа φ, находящимся на пути от начала программы к текущему фрагменту.

Реализация очередного оператора P_i текущего фрагмента $P_1, \ldots, P_n$ программы символа φ происходит следующим образом:

1) Если оператор имеет вид $\psi(t_1 \ldots t_n)$, где ψ - программно реализуемый символ, то последовательно определяются значения входных операторных выражений t_i и осуществляется обращение к программе символа ψ; действия в случае непосредственно реализуемого ψ описаны в следующем разделе. Если по окончании выполнения оператора он принимает значение "ложь", то переход к пункту 2. В противном случае оказываются определены значения всех выходных переменных этого оператора. Если при реализации оператора P_i имел место режим перечисления, то оператор заносится в конец стэка перечисляющих операторов. Если $i < n$, то переход к выполнению следующего оператора P_{i+1}, иначе - переход к пункту 3.

2) Если P_{i+1} имеет вид "иначе N", то осуществляется переход к первому оператору N - го фрагмента программы. Если стэк перечисляющих операторов пуст, то выполнение программы символа φ завершается. При этом, если происходило сканирование задачи, то делается очередной шаг сканирования; если вычислялось операторное выражение $\varphi(\theta_1 \ldots \theta_k)$, то выдается значение "0" (логический символ); если же обрабатывался оператор $\varphi(\theta_1 \ldots \theta_k)$, то он получает значение "ложь". Если стэк перечисляющих операторов ($Q_1 \ldots Q_m$) непуст, то осуществляется возвращение к последнему оператору Q_m, который удаляется из стэка. При этом значения всех переменных, номера которых больше номера последней определенной до обращения к Q_m переменной, становятся не определены. Оператор Q_m определяет очередной набор значений своих выходных переменных либо, при отсутствии такого набора, принимает значе-

ние "ложь", и далее повторяется выполнение программы начиная с оператора, следующего за Q_m. Особо выделяется случай оператора Q_m вида "ветвь N". В этом случае просто выполняется переход к первому оператору N - го фрагмента программы.

3) В языке ЛОС имеется несколько специальных операторов, размещаемых обычно в конце фрагментов программы и используемых для управления ходом выполнения программы. Если P_i представляет собой один из таких заключительных операторов, то выполняются следующие действия:

а) P_i = "выход". В этом случае программа символа φ осуществляет реализацию проверочного оператора, который получает значение "истина".

б) P_i = "стоп". В этом случае программа осуществляет реализацию оператора (не обязательно проверочного), который получает значение "ложь".

в) P_i = "результат($xk_1\, t_1\, xk_2\, t_2 \ldots xk_s\, t_s$)". Программа осуществляет реализацию перечисляющего оператора $\varphi(\tau_1 \ldots \tau_p)$; выходным переменным $\tau_{k_1}, \ldots, \tau_{k_s}$ этого оператора присваиваются значения операторных выражений $t_1, \ldots, t_s$. Заметим, что при реализации проверочного оператора $\varphi(\tau_1 \ldots \tau_p)$ присвоение значений выходным переменным происходит таким же образом, но оператор "результат(...)" располагается непосредственно перед завершаюшим оператором "выход".

г) P_i = "продолжение". Этот оператор является тождественно ложным, т.е. происходит возвращение к последнему оператору Q_m стэка перечисляющих операторов либо (при пустом стэке) выход из программы.

д) P_i = "обрыв" либо P_i = "сброс(k)"; k - натуральное число. Происходит возвращение к оператору Q_{m-k} (в первом случае $k = 1$) из стэка $(Q_1 \ldots Q_m)$ перечисляющих операторов; если такого оператора нет, то - выход из программы, как при пустом стэке.

е) P_i = "ответ(t)". В этом случае программа либо выполняла сканирование задачи, и тогда определяется ответ t на эту задачу, либо вычисляла операторное выражение (в частности, вида "справка($\psi\varphi \ldots$)"), получающее значение операторного выражения t.

ж) P_i = "пересмотр". В этом случае программа выполняет сканирование задачи, которое возобновляется повторно при нулевом текущем уровне задачи.

з) P_i = "контроль". Программа выполняет сканирование задачи. Если эта задача имеет хотя бы один терм (посылку либо условие) с нулевым весом (что бывает при изменении терма либо занесении нового терма), то начинается повторное сканирование задачи при нулевом текущем уровне. В противном случае - переход к последнему перечисляющему оператору Q_m (если его нет, то выход из программы).

Если P_n не является оператором одного из указанных видов, то либо имеет место сканирование задачи, и тогда осуществляется очередной шаг сканирования, либо происходит реализация проверочного оператора без выходных переменных, который получает значение "истина".

5.2 Основные операторы языка ЛОС

5.2.1 Общие операторы

Язык ЛОС имеет свыше 200 реализуемых интерпретатором операторов, ориентированных главным образом на обработку данных сетевого и логического типов. Часть этих операторов ("выход", "стоп", "результат($x1\ t_1 \ldots xn\ t_n$)", "ответ(t)", "обрыв", "сброс(n)", "продолжение", "контроль", "пересмотр") была рассмотрена в предыдущем разделе. Приведем еще ряд простых операторов управляющего характера.

Операторы "решить", "программа", "обращение(ψ)" принимают значение "истина", соответственно, если программа символа φ применяется при сканировании задачи, при реализации оператора и при вычислении операторного выражения t; в последнем случае $\psi \neq$ "0" означает, что t имело вид "справка($\psi\ \varphi \ldots$)", а $\psi =$ "0" - что t не имело такого вида. Оператор "определено(xi)" истинен, если определено значение переменной xi. Оператор "повторение" представляет собой вырожденный перечисляющий оператор, принимающий при каждом выходе на него значение "истина"; он аналогичен оператору repeat в ПРОЛОГе. Оператор "уровень($n_1 \ldots n_k$)" (n_1, $\ldots, n_k$ - логические символы от "0"до "20") истинен, если происходит сканирование задачи, текущий уровень которой равен одному из чисел $n_1, n_2, \ldots, n_k$; оператор "последнийуровень если текущий уровень решаемой задачи Z_N равен ее максимальному уровню. Оператор "новый"применяется при сканировании задачи; он принимает значение "истина"в том случае, когда рассматриваемый терм задачи впервые попал в поле зрения системы при данном текущем уровне m_N. Более точно, этот оператор становится ложнм лишь в ситуациях повторного просмотра при значениях текущего уровня $U = 0, 1, \ldots, m_N$ временно выпавшего из поля зрения терма, имеющего вес $m_N + 1$ (см. пункт 3 описания процедуры сканирования задачи в последнем разделе первой главы). Оператор "новый" позволяет блокировать повторную попытку рассмотрения приема, если изменение окрестности терма не влияет на результаты такого рассмотрения.

Перейдем к описанию серии общелогических операторов языка. Если P - произвольный проверочный оператор, не имеющий выходных переменных, то утверждение "не(P)" является проверочным оператором, истинным тогда и только тогда, когда ложен P. Если $P_1, \ldots, P_n$ - операторы, то утверждение "и($P_1 \ldots P_n$)" является оператором. Если хотя бы один из операторов $P_1, \ldots, P_n$ реализуется в режиме перечисления, то конъюнкция их также считается реализуемой в режиме перечисления. В стэк перечисляющих операторов при этом заносится не последний перечисляющий оператор P_i, а вся конъюнкция. Аналогичным образом формируется дизъюнкция "или($P_1 \ldots P_n$)" операторов $P_1, \ldots, P_n$. Если каждый из операторов $P_1, \ldots, P_n$ проверочный и не имеет выходных переменных, то дизъюнкция является проверочным оператором. Если каждый оператор $P_i, i = 1, \ldots, n$, имеет непустое множество выходных переменных, то дизъюнкция реализуется в режиме перечисления (даже если все P_i - проверочные). При этом сначала перечисляются значения выходных переменных, определяемые оператором P_1, затем - P_2, и т.д. Дизъюнктивные конструкции, у которых часть операторов P_i имеет пустое множество выходных переменных, а часть - непустое, не рекомендуются, так как тогда наличие либо отсутствие режима перечисления определяется неоднозначно.

Если $P_1, \ldots, P_n$ - операторы, $n > 0$,P_0 - проверочный оператор, $x_1, \ldots, x_k$ - переменные, содержащие все выходные переменные операторов $P_0, P_1, \ldots, P_n$, то утвер-

ждение "длялюбого($x_1 \dots x_k$ если $P_1 \dots P_n$ то P_0)" является проверочным оператором. К моменту его реализации последняя определенная переменная должна иметь номер, меньший номеров переменных $x_1, \dots, x_k$. Все свободные переменные оператора P являются его входными переменными. При фиксированных значениях этих переменных цепочка операторов $P_1, \dots, P_n$ (среди которых могут встречаться перечисляющие операторы) определяет процесс перечисления удовлетворяющих условиям $P_1, \dots, P_n$ наборов значений их выходных переменных, причем для каждого такого набора вычисляется истинностное значение оператора P_0. Как только здесь возникает значение "ложь", процесс обрывается и P получает значение "ложь". В противном случае, по завершении перечисления, оператор P получает значение "истина", а значения всех переменных $x_1, \dots, x_k$ снова оказываются не определены. Заметим, что управляющие ходом перечисления операторы "повторение", "обрыв", "сброс(n)" внутри сложных операторов (в частности, в кванторных операторах) не используются. Аналогичным образом определяется оператор P вида "существует($x_1 \dots x_k$ P_0)", где P_0 - некоторый оператор (возможно, перечисляющий), все выходные переменные которого содержатся среди $x_1, \dots, x_k$. Как и в случае квантора общности, к моменту реализации P последняя определенная переменная должна иметь номер, меньший номеров переменных $x_1, \dots, x_k$. (Это правило является общим для всех операторов и операторных выражений со связанными переменными и далее особо не оговаривается. Вообще, при программировании на ЛОСе предпочтительно вводить новые программные переменные так, чтобы очередной номер переменной был на единицу больше наибольшего из уже использованных номеров.) Если при фиксации значений входных переменных оператора P оператор P_0 оказывается истинным для некоторого набора значений своих выходных переменных, то определяемое им перечисление обрывается, P получает значение "истина", а значения переменных $x_1, \dots, x_k$ становятся снова не определены.

К числу основных логических операторов присоединен оператор P = "альтернатива($P_0P_1P_2$)", где P_0 - проверочный оператор без выходных переменных; P_1, P_2 - некоторые операторы (не обязательно проверочные). Если P_0 получает на некотором наборе значений своих переменных значение "истина", то далее выполняется оператор P_1, иначе - оператор P_2. Здесь возникает возможность появления смешанных проверочно- перечисляющих операторов, так как если один из операторов P_1,P_2 проверочный, а другой - перечисляющий, то выбор фактического режима работы оператора P происходит в зависимости от значений его входных переменных. По аналогии с оператором "альтернатива($P_0P_1P_2$)" устроено операторное выражение "вариант(Pt_1t_2)", где P - проверочный оператор без выходных переменных; t_1 и t_2 - операторные выражения. Значение этого выражения совпадает со значением t_1 при истинном P и со значением t_2 при ложном P.

Для формирования наборов объектов, накапливаемых в процессе просмотра тех или иных структур данных, служат операторные выражения со связанными переменными "перечисление($x_1 \dots x_k$ P t)"и "выписка($x_1 \dots x_k$ P t)"(аналоги setof и findall в ПРОЛОГе). Здесь P - некоторый (как правило, перечисляющий) оператор; t - операторное выражение; $x_1, \dots, x_k$ - переменные, включающие все выходные переменные оператора P. При определении значения этих выражений оператор P перечисляет некоторые наборы значений своих выходных переменных, и для каждого такого набора определяется значение выражения t. В первом случае (оператор "перечисление(...)") найденные указанным образом значения выражения t записываются в формируемый набор подряд, с исключением повторений; во втором случае

(оператор "выписка(...)") исключение повторений не выполняется. Операторное выражение "суммавсех($x_1 \dots x_k\, P\, t$)"осуществляет суммирование значений выражения t (эти значения должны быть числами, т.е. цифрами либо наборами, см. описание представления чисел в разделе 1), возникающих в процессе реализации перечисляющего оператора P так же, как для операторов "перечисление(...)", "выписка(...)". Это выражение используется , например, при принятии решений для интегральной оценки ситуации.

Вычисление значения уже упоминавшегося в первом разделе операторного выражения "справка($\psi\, \varphi\, t_1 \dots t_n$)", где ψ, φ - логические символы; $t_1, \dots, t_n$ - операторные выражения, осуществляется программой логического символа φ при "типе обращения"ψ к этой программе и исходных значениях $t_1, \dots, t_n$ переменных $x_1, \dots, x_n$. Это выражение используется в тех случаях, когда некоторая процедура, условно обозначаемая логическим символом ψ, естественным образом распадается на множество независимых фрагментов, связанных с различными логическими символами φ.

Оператор "равно($t_1 t_2$)"либо не имеет выходных переменных (если все входящие в него переменные имеют уже определенные на предыдущих этапах значения) и осуществляет проверку равенства значений выражений t_1 и t_2 (равенство наборов понимается как равенство определяемых ими древовидных конструкций), либо имеет выходную переменную t_1, которой присваивается значение выражения t_2.

Для изменения значения переменной x_N используется оператор "замена(x_N t)", где t - операторное выражение, определяющее заменяющий объект. Сразу заметим, что в фактически исполняемой программе здесь вместо переменной x_N хранится ее символьный номер (логический символ номер $260 + N$), который и сообщается интерпретатору ЛОСа в качестве первого входного данного оператора "замена(...)". Аналогичный оператор "изменение($t_1 t_2$)" позволяет изменять разряд набора, вхождение которого определяется выражением t_1, на объект, являющийся значением выражения t_2. Применение операторов "замена(...)"и "изменение(...)" требует определенной аккуратности, так как после их реализации значения переменных могут измениться таким образом, что истинность предшествующих операторов рассматриваемого фрагмента программы нарушится.Эти операторы оказываются полезны при реализации циклических процедур в тех достаточно редких случаях, когда цикл не удается оформить с помощью перечисляющих операторов.

Возможно обращение к оператору либо операторному выражению, заголовок которых определяется некоторым операторным выражением. Это осуществляется при помощи терма "процедура(t_1 t_2 t_3)", у которого значением t_1 является указанный заголовок, причем при t_3 равном "0"данный терм реализует обращение к оператору, а при t_3 равном "1 к операторному выражению. Значением t_2 является набор входных данных соответствующего оператора либо операторного выражения. В случае оператора с выходными переменными позиции набора t_2, соответствующие выходным переменным, заняты логическим символом "неопред". Эти позиции после обращения к оператору "процедура(...)"изменяются на найденные значения выходных переменных.

В тех случаях, когда предпринимается обращение к процедуре, выполнение которой может оказаться неприемлемо трудоемким, используется фиктивный перечисляющий оператор "лимит(t_1)". Этот оператор устанавливает ограничение в t_1 шагов работы интерпретатора (количество таких шагов автоматически фиксируется в специальном регистре интерпретатора и может рассматриваться как машинно-

независимые внутренние "часы"решателя) начиная с момента обращения к нему. По истечении этого числа шагов, если все еще не было выхода из следующей за данным оператором ветви программы, реализуется откат к оператору с присвоением ему истинностного значения "ложь". Если после обращения к подпрограмме, которое могло оказаться слишком трудоемким, нужно отменить откат к оператору "лимит(...)", то применяется оператор "Обрыв". Единственное его действие - исключение из стэка перечисляющих операторов последнего элемента (не обязательно установленного оператором "лимит(...)"), без каких-либо откатов.

В заключение подраздела упомянем операторное выражение "типданных(t)", позволяющее различать типы значения выражения t. Если A - логический символ либо символ переменной, то значение p выражения "типданных(t)"равно 0; если A - терм, то p равно 1; если A - вхождение в терм либо в набор, то p равно 2; если A - набор, не являющийся термом, то p есть 4. Значение $p = 3$ зарезервировано для тех случаев, когда A есть ссылка на недопустимый объект (сигнал о наличии ошибки).

5.2.2 Операторы просмотра и преобразования задачи

Для выделения отдельных элементов задачи служит ряд специальных операторных выражений. Так, "цели(t)"имеет своим заголовком список целей задачи, определяемой выражением t; "неизвестные(t)" - цель, перечисляющую неизвестные (если ее нет, то это выражение имеет значение "пустоеслово"); "комментарии(t_1)", "комментариипосылок(t_1)", "комментарииусловия(t_1t_2)", "комментариипосылки(t_1t_2)" - соответствующие списки комментариев (задача определяется здесь выражением t_1, а вхождение рассматриваемого условия либо посылки - выражением t_2); "списокусловий(t)" и "списокпосылок(t)" - списки условий и посылок. Значением выражения "переменные(t)" служит список всех переменных, встречающихся в посылках и условиях задачи, а также среди ее неизвестных. Выражение "максимальныйуровень(t)" позволяет определить максимальный уровень задачи. Для обращения к процедуре решения задачи служат операторные выражения "ответзадачи(t)" и "прямойответ (t)". Выражение t в обоих случаях имеет своим значением некоторую задачу (т.е. набор допустимого вида; см.описание структуры данных задачи в третьей главе), причем в первом случае эта задача начинает решаться с максимальным уровнем, равным текущему уровню решаемой задачи Z_N, а во втором случае - с максимальным уровнем 4. Если перед вычислением значения выражения "ответзадачи(t)"применить оператор "уровеньобращения(k)", то новая задача будет решаться с максимальным уровнем k (установка уровня обращения к задаче имеет одноразовый характер и после обращения к задаче пропадает). Ответ либо логический символ "отказ", найденный при решении задачи, становится значением соответствующего операторного выражения.

Операторы "исходнаязадача(x)", "текущаязадача(x)"присваивают переменной x исходную и текущую задачи цепи задач, а оператор "надзадача(x t)"перечисляет в обратном порядке (т.е. от текущей задачи к исходной) все элементы цепи задач, тип которых равен значению выражения t. Если t имеет значение 0, то последнее ограничение отменяется. Сама текущая задача из перечисления исключается. Оператор "текущийуровень(x)"присваивает переменной x значение текущего уровня текущей задачи. Операторы "посылка($t_1t_2t_3$)"и "условие($t_1t_2t_3$)"используются в следующих двух вариантах:

а) Значением выражения t_3 является задача Z; t_1, t_2 - выходные переменные. В этом случае перечисляются все посылки либо, соответственно, условия задачи Z (порядок перечисления - слева направо), причем значением t_1 становится рассматриваемый терм, а значением t_2 - его вхождение в список посылок либо список условий. Если Z не имеет списка условий, то при реализации оператора "условие(...)"режим перечисления не включается; значением t_1 становится единственное условие задачи, а значением t_2 - его вхождение в задачу.

б) t_1 и t_3 - входные выражения, значения которых суть терм θ и задача Z; t_2 - выходная переменная. Выполняется проверка того, что θ - посылка (условие) задачи Z, и если это так, то t_2 становится равно вхождению θ в соответствующий список либо в саму задачу Z.

В некоторых случаях бывает необходимо осуществить просмотр всех посылок либо условий задачи, имеющих заданный вес; как правило, таковыми оказываются вновь введенные за некоторый период посылки либо условия. Для этого применяются операторы "новаяпосылка($t_1t_2t_3t_4$)", "новоеусловие($t_1t_2t_3t_4$)", реализация которых отличается от случая а) реализации операторов "посылка($t_1t_2t_3$)", "условие($t_1t_2t_3$)" лишь тем, что отбираются термы с весом, равным значению выражения t_4. Операторы "цель(t_1t_2)", "неизвестная(t_1t_2)"допускают два режима реализации, причем в каждом из них значением выражения t_1 является задача Z, не имеющая тип "доказать":

а) t_2 - выходная переменная, и тогда оператор перечисляет значения переменной t_2, являющиеся целями (неизвестными) задачи Z.

б) t_2 определено, и тогда оператор проверяет, что значением выражения t_2 является цель (неизвестная) задачи Z.

Ряд операторов служит для преобразования задачи. Прежде всего, это операторы "заменаусловия($t_1t_2t_3$)"и "заменапосылки($t_1t_2t_3$)" (t_1 - задача, t_2 - вхождение заменяемого терма в соответствующий набор, t_3 - заменяющий терм); "занесениеусловия(t_1t_2)"и "занесениепосылки(t_1t_2)"(t_1 - терм, t_2 - задача, имеющая в первом случае тип "описать"); "удалениеусловия(t_1t_2)"и "удалениепосылки(t_1t_2)"(t_1 - задача, t_2 - вхождение удаляемого терма). Измененный либо новый терм получает в задаче вес 0; если оператор указанного типа является заключительным в фрагменте программы и обращение к этой программе произошло при сканировании задачи, то текущий уровень задачи заменяется на 0 и цикл сканирования повторяется сначала.

Для занесения в соответствующие списки новых комментариев используются операторы "примечание($t\ \theta_1 \dots \theta_n$)"(вводится общий комментарий ($\alpha_1 \dots \alpha_n$) либо, при $n = 1$, общий комментарий α_1 к посылкам задачи, определяемой выражением t; $\alpha_1, \dots, \alpha_n$ - значения выражений $\theta_1, \dots, \theta_n$; α_1 - логический символ), "замечание($t\ \theta_1 \dots \theta_n$)" (вводится общий комментарий к задаче, определяемой выражением t), "примечпосылки($t\ \tau\ \theta_1 \dots \theta_n$)" (вводится комментарий к посылке, определяемой своим вхождением τ в список посылок), "замечусловие($t\ \tau\ \theta_1 \dots \theta_n$)" ($t$ определяет задачу на описание; τ - вхождение в список условий того условия, к которому относится комментарий).

Проверка отсутствия заданного комментария выполняется аналогичными по своей структуре операторами "коммент($t\theta_1 \dots \theta_2$)"(отсутствие среди общих комментариев к задаче набора ($\alpha_1 \dots \alpha_n$) либо, при $n = 1$, символа α_1), "комментпосылок ($t\theta_1 \dots \theta_n$)", "комментусловия($t\ \tau\ \theta_1 \dots \theta_n$)", "комментпосылки($t\ \tau\ \theta_1 \dots \theta_n$)".

Завершают перечень операторов задач операторы "удалениепримечания($t\theta$)", "удалениезамечания($t\,\theta$)", "удалениепримечпосылки($t\,\tau\,\theta$)", "удалениезамечусловия ($t\,\tau\,\theta$)", выполняющие удаление определяемого выражением θ комментария (t определяет задачу, τ - вхождение рассматриваемой посылки либо условия.

5.2.3 Операторы логического представления данных

Начнем с перечисления операторных выражений, предназначенных для работы с логическими конструкциями. Выражения "списокпеременных(t)", "параметры(t)" определяют, соответственно, набор всех переменных терма, являющегося значением выражения t, и набор всех свободных переменных этого терма. В последнем случае значением выражения t может служить как один терм, так и набор термов. Для образования новых термов используется операторное выражение "запись($t\theta_1 \ldots \theta_n$)"; если выражения $t, \theta_1, \ldots, \theta_n$ определяют, соответственно, логический символ φ и термы либо символы (т.е. логические символы либо символы переменных) $\tau_1, \ldots, \tau_n$, то формируется новый терм $\varphi(\tau_1 \ldots \tau_n)$. В тех случаях, когда набор операндов имеет переменную длину, новые термы определяются операторным выражением "сборка($t\ \theta$)". Здесь t определяет логический символ φ; θ - набор операндов $\tau_1, \ldots, \tau_n$ - термов либо символов. Для выделения фрагментов уже построенного терма применяются операторные выражения "подтерм(t)" (t указывает вхождение корня переписываемого подтерма), "набороперандов(t)" (t указывает вхождение первого символа терма $\varphi(\theta_1 \ldots \theta_n)$, и значением выражения становится набор термов $\theta_1, \ldots, \theta_n$), а также операторные выражения "первыйтерм(t)", "второйтерм(t)", "предпоследтерм (t)", "последнийтерм(t)"для наиболее часто рассматриваемых операндов $\theta_1, \theta_2, \theta_{n-1}$, θ_n терма $\varphi(\theta_1 \ldots \theta_n)$, вхождение первого символа которого определяется выражением t. Операторные выражения "первыйоперанд(t)", "второйоперанд (t)", "предпоследоперанд(t)", "последнийоперанд(t)"позволяют определить вхождения заголовков указанных операндов $\theta_1, \theta_2, \theta_{n-1}, \theta_n$. Операторные выражения "следоперанд(t_1)"и "предоперанд(t_1)"позволяют находить вхождения операнда некоторой операции, соответственно, непосредственно следующего за вхождением t_1 операнда той же операции либо предшествующего ему. В концевых случаях, при отсутствии следующего либо предыдущего операнда, выдается само значение t_1. Операторное выражение "операндномер($t\ n$)"имеет своим значением вхождение n - го операнда операции, расположенной по вхождению t. Здесь n - символьный номер (логический символ с номером $260 + n$). Если операция имеет меньше чем n операндов, то выдается логический символ 0.

Операторное выражение "заменатермов ($t_1 t_2 t_3$)" определяет результат замены в терме θ набора вхождений ($\alpha_1 \ldots \alpha_n$) расположенных слева направо непересекающихся подтермов (при $n = 1$ рассматривается не набор (α_1), а само вхождение α_1) на набор термов либо символов ($\tau_1 \ldots \tau_n$). Здесь t_1 определяет θ, t_2 - вхождения, а t_3 - заменяющие термы. Выражение "исключениеоперанда($t_1 t_2$)"позволяет находить терм $\varphi(\theta_1 \ldots \theta_{i-1} \theta_{i+1} \ldots \theta_n)$, получающийся из терма $\varphi(\theta_1 \ldots \theta_n)$ исключением операнда θ_i (если $n = 2$, то результатом исключения становится оставшийся операнд $\theta_j, j \neq i$); здесь t_1 определяет вхождение первого символа терма $\varphi(\theta_1 \ldots \theta_n)$, а t_2 - вхождение операнда θ_i. Если выражение t определяет вхождение символа в некоторый терм θ, то операторное выражение "базавхождения(t)" имеет своим значением данный терм θ. Для определения переменной с заданным номером применяется выражение "икс(t)", где t определяет N - й после "0"логический символ; N - номер требуемой

переменной. Выражение "кортежпеременных(t_1t_2)"определяет набор $(y_1 \dots y_k)$ переменных, отличных от переменных набора - значения выражения t_2 и имеющий ту же длину, что и набор - значение выражения t_1. Аналогичной цели служит оператор "новаяпеременная(t_1t_2)", у которого t_1 определяет набор переменных, а выходной переменной t_2 присваивается переменная, не входящая в указанный набор.

Операторы "логсимвол(t)", "переменная(t)"позволяют распознавать логические символы и переменные; операторы "корень(t)", "стандарт(t) распознавать корневое вхождение в терм и вхождение символа, за которым идет открывающая скобка. Последний оператор бывает полезен, чтобы отличить обычное использование символа операции φ в конструкциях вида $\varphi(\theta_1 \dots \theta_n)$ от использования его в качестве синтаксической константы в описаниях вида термов. При обращении к оператору "символ (t_1t_2)" значением выражения t_1 является вхождение в терм либо в набор. Если t_2 - выходная переменная, то ей присваивается объект, имеющий указанное вхождение; в противном случае оператор проверяет совпадение этого объекта со значением выражения t_2. Аналогично, оператор "заголовок(t_1t_2)"либо присваивает выходной переменной t_2 первый символ терма, являющегося значением t_1, либо, если t_2 определено, проверяет совпадение этого заголовка со значением t_2. Операторы "первыйсимвол (t_1t_2)", "второйсимвол(t_1t_2)", "предпоследсимвол(t_1t_2)", "последнийсимвол(t_1t_2)" определяют, соответственно, заголовки (логические символы либо переменные) операндов $\theta_1, \theta_2, \theta_{n-1}, \theta_n$ подтерма $\varphi(\theta_1 \dots \theta_n)$, вхождение первого символа которого задается выражением t_1; если t_2 - выходная переменная, то ей присваивается найденный заголовок, иначе он сравнивается со значением t_2. Эти операторы часто используются при сканировании задачи для идентификации того или иного подтерма. Оператор "свобвхождение(t)"распознает свободное вхождение переменной в терм; оператор "лексикопредшствует(t_1t_2) проверяет лексикографическое предшествование термов. Последний оператор бывает полезен для стандартного упорядочения операндов коммутативных и ассоциативных операций.

Перечислим ряд операторов,используемых для перемещения по вхождениям в терм. Оператор "операнд(t_1t_2)" используется в следующих двух возможных режимах:

а) t_2 определено и имеет значением вхождение α в терм; t_1 - выходная переменная, которой присваивается вхождение той операции, чьим операндом является α. Если такой нет, то оператор ложен.

б) t_1 определено и имеет значением вхождение первого символа подтерма $\varphi(\theta_1 \dots \theta_n)$; $n > 0$. t_2 - выходная переменная, значения которой перечисляют вхождения операндов $\theta_i; i = 1, \dots, n$.

Оператор "новоперанд(t_1 t_2)"имеет своим входным данным t_1 вхождение операнда некоторой операции. Выходная переменная t_2 перечисляет вхождения операндов той же операции, следующие за t_1 (исключая само t_1).

Для просмотра всех внешних операций,а также всех подряд символов некоторого подтерма служит оператор "подчинено(t_1t_2). Существует три возможных режима его реализации:

а) t_1 определено и имеет своим значением вхождение α в некоторый терм. t_2 - выходная переменная, перечисляющая в направлении от α к корню терма все вхождения заголовков содержащих α подтермов (само вхождение α отбрасывается).

б) t_2 определено и имеет своим значением вхождение β заголовка некоторого подтерма; t_1 - выходная переменная, перечисляющая слева направо все вхождения символов в этот подтерм (начиная с заголовка).

в) t_1, t_2 определены и задают вхождения α, β заголовков подтермов τ_1, τ_2. Оператор истинен, если τ_1 лежит внутри τ_2 либо совпадает с τ_2.

Оператор "вхождениетерма($t_1t_2t_3$)"имеет входные данные t_1 (терм θ_1) и t_2 (терм θ_2); t_3 - выходная переменная, перечисляющая (слева направо) все вхождения терма θ_2 в терм θ_1. Для просмотра всех антецедентов $f_1, \ldots, f_n$ импликативной конструкции "длялюбого($x_1 \ldots x_k$ если $f_1 \ldots f_n$ то f_0)"применяется оператор "антецедент(t_1t_2)", у которого значением t_1 служит указанная конструкция либо вхождение ее заголовка, а выходная переменная t_2 перечисляет вхождения корней утверждений $f_1, \ldots, f_n$.

Для сравнения термов,определенных вхождениями своих заголовков, служит оператор "равныетермы(t_1t_2)".

Чтобы ускорить поиск в заданном наборе термов A терма, подобного терму t (получающегося из него переобозначением связанных переменных и перестановкой операндов коммутативных операций и симметричных отношений), можно использовать оператор "ключподобия($A\ t\ x$)". Его выходная переменная x перечисляет вхождения в список A тех термов, которые могут оказаться подобными терму t (сравниваются длины термов, их заголовки и суммы кодов входящих в терм логических символов и переменных; код любой переменной здесь равен 1).

Завершает перечень операторов логических структур данных оператор "результподст($t_1t_2t_3t_4$)". Он допускает следующие два возможных режима реализации:

а) t_1, t_2 определяют набор переменных $(x_1 \ldots x_k)$ и набор термов $(\tau_1 \ldots \tau_k)$ соответственно; t_3 определяет терм θ. t_4 - выходная переменная, которой присваивается терм - результат подстановки термов $\tau_1, \ldots, \tau_k$ вместо переменных $x_1, \ldots, x_k$ в терм θ.

б) t_1 определяет набор переменных $(x_1 \ldots x_k)$; t_3 и t_4 - термы θ и θ'. t_2 - выходная переменная, которой присваивается такой набор термов $(\tau_1 \ldots \tau_k)$, подстановка которого в терм θ вместо переменных $x_1, \ldots, x_k$ дает терм θ'; если x_i не входит в θ, то τ_i становится равно логическому символу "пустоеслово". При отсутствии указанного набора оператор ложен.

5.2.4 Операторы сетевого представления данных

Операторы сетевого представления данных связаны с рассмотрением наборов и вхождений в них (в сетевых конструкциях вершины,а иногда и ребра кодируются посредством наборов). Операторное выражение "набор($t_1 \ldots t_n$)"($n \geq 1$) позволяет формировать набор значений выражений $t_1, \ldots, t_n$. Если выражения $t_1, \ldots, t_n$ определяют наборы либо символы (т.е. логические символы либо символы переменных), то выражение "конкатенация($t_1 \ldots t_n$)" определяет конкатенацию этих наборов и символов; логический символ "пустоеслово"при этом рассматривается как набор длины 0. Выражения "суффикс($t\ \theta$)" и "префикс($t\ \theta$)" позволяют присоединять к концу (соответственно, к началу) набора, определяемого выражением t, значение выражения θ. Операторное выражение "окончание(t)"осуществляет удаление первого элемента набора (пустой набор этим выражением сохраняется); выражение "вставка($t_1t_2t_3$)"имеет своим значением набор $(\alpha_1 \ldots \alpha_{i-1}\beta\alpha_i \ldots \alpha_n)$, полученный из набора $(\alpha_1 \ldots \alpha_n)$, определяемого выражением t_1, заменой i - го разряда, вхождение которого определяется выражением t_2, на объект β , определяемый выражением t_3. Если значения выражений t_1, t_2 суть набор $(\alpha_1 \ldots \alpha_n)$ и набор вхождений в этот набор разрядов $\alpha_{i_1}, \ldots, \alpha_{i_s}$; $i_1 < \ldots < i_s$ (если $s = 1$, то берется не набор вхождений, а само

вхождение), то значением операторного выражения "исключение(t_1t_2)"служит результат удаления указанных разрядов из набора $(\alpha_1 \dots \alpha_n)$. Выражение "заменаразряда($t_1t_2t_3$)"определяет результат замены в заданном наборе (значение t_1) заданного вхождения разряда (значение t_2) на заданный объект (значение t_3). Если выражения t_1, t_2 определяют наборы α, β, то выражение "пересечениесписков(t_1t_2)"имеет своим значением набор, полученный из α сохранением всех разрядов, встречающихся в β, причем с учетом кратности: число сохраняемых экземпляров одного и того же объекта набора α не превосходит числа его экземпляров в наборе β. Выражение "объединениесписков(t_1t_2)" имеет своим значением набор $\alpha\beta'$, где β' получено из β удалением всех элементов, входящих в α, а выражение "вычеркивание(t_1t_2)" - набор, получающийся из α в результате непродолжаемой последовательности одновременных вычеркиваний у α и β пар одинаковых элементов (порядок просмотра α - слева направо). Выражение "бланк($t_1t_2t_3$)", где t_1 определяет набор длины k, $k \geq 0$, t_2 - некоторый объект α; t_3 - N -й после "0"логический символ ($N \geq 0$), имеет своим значением набор $(\alpha \dots \alpha)$ длины $N + k$. Наконец, последнее из серии операторных выражений, используемых для построения новых наборов, - выражение "копия(t)", формирующее копию набора (либо терма).

Выражение "буква(t)"имеет своим значением объект, вхождение которого в набор либо терм определяется выражением t. Если значением выражения t_1 является набор $(\alpha_1 \dots \alpha_n)$, а значением t_2 - N - й после "0"логический символ ($N \geq 0$), то значением выражения "левпозиция(t_1t_2)"служит объект α_{N+1}; значением выражения "правпозиция(t_1t_2)" - объект α_{n-N}; значением выражения "окрестность(t_1t_2)" - вхождение $N + 1$ -го разряда в набор $(\alpha_1 \dots \alpha_n)$. В этой же ситуации значением выражения "левыйкрай(t_1)" является вхождение разряда α_1; выражения "правыйкрай(t_1)" - вхождение α_n; выражения "начало(t_1)" - объект α_1 и выражения "конец(t_1)" - объект α_n. Если значением выражения t является вхождение разряда α_i в некоторый набор $(\alpha_1 \dots \alpha_n)$, то выражение "левсосед(t)"определяет вхождение α_{i-1}, а "правсосед(t)" - вхождение α_{i+1} (если разряд крайний и сдвиг невозможен, то сохраняется значение α_i). Оператор "позиция(t_1t_2)"имеет выходную переменную t_1, перечисляющую слева направо все вхождения разрядов в набор, определяемый выражением t_2. Оператор "входит(t_1t_2)", аналогичный предыдущему, имеет два возможных режима реализации:

а) t_2 определено и имеет своим значением набор $(\alpha_1 \dots \alpha_n)$; t_1 - выходная переменная, перечисляющая объекты $\alpha_i (i = 1, \dots, n)$.

б) t_1, t_2 определены и имеют значения α, β; β - набор. В этом случае оператор проверяет вхождение объекта α в набор β.

Для поиска в наборе используются: оператор "разряд($t_1t_2t_3$)", где t_1 определяет набор либо терм α; t_2 - логический символ либо переменную φ; t_3 - выходная переменная, перечисляющая все вхождения φ в α, а также операторы "ключ($t_1t_2\theta$)" и "биключ($t_1t_2t_3\theta$)". У последних двух операторов t_1 определяет набор α; t_2 - логический символ β;t_3 - некоторый объект γ. Выходная переменная θ перечисляет все элементы набора α, имеющие в первом случае вид $(\beta, \dots)$ либо $\beta(\dots)$ либо равные β, а во втором случае - вид $(\beta, \gamma, \dots)$ либо $\beta(\gamma, \dots)$, либо $\beta(\gamma(\dots), \dots)$. Эти операторы часто применяются при поиске комментариев заданного типа, а также при отборе условий либо посылок задачи, имеющих заданный вид. Если выражения t_2, t_3 определяют наборы $(\alpha_1 \dots \alpha_n)$ и $(\beta_1 \dots \beta_m)$, а выражение t_1 - вхождение разряда $\alpha_i, i \leq m$, то операторное выражение "соотвпозиция($t_1t_2t_3$)"определяет вхождение разряда β_i.

Аналогично, если t_1, t_2 определяют наборы $(\alpha_1, \ldots, \alpha_n)$ и $(\beta_1, \ldots, \beta_n)$, а t_3 - некоторый объект β, то выражение "таблзначение$(t_1 t_2 t_3)$"либо имеет своим значением объект β_j, такой, что $\beta = \alpha_j$ и $\beta \neq \alpha_1, \ldots, \alpha_{j-1}$, либо, при отсутствии указанного объекта, имеет значение β. Оператор "Таблзначение$(t_1 t_2 t_3 t_4)$"позволяет перечислять все элементы t_4 набора t_2, у которых на соответствующей позиции набора t_1 располагается элемент, равный t_3.

Приведем ряд специальных операторов, используемых для перечисляения вхождений в наборы. Если выражения $t_1, \ldots, t_n$ имеют своими значениями наборы $(\alpha_{11}, \ldots, \alpha_{1m_1}), \ldots, (\alpha_{n1}, \ldots, \alpha_{nm_n})$ соответственно; $\theta_1, \ldots, \theta_n$ - различные переменные, не встречающиеся в $t_1, \ldots, t_n$, то оператор "серия$(\theta_1 t_1 \ldots \theta_n t_n)$"имеет выходные переменные $\theta_1, \ldots, \theta_n$ и перечисляет такие наборы $(\beta_{1i}, \ldots, \beta_{ni})$ $(i = 1, \ldots, \min(m_1, \ldots, m_n))$ их значений, что β_{ji} - вхождение разряда α_{ji} в набор $(\alpha_{j1} \ldots \alpha_{jm_j})$; $j = 1, \ldots, n$. Просмотр наборов "соответствующих"вхождений справа налево выполняется оператором "контрсерия$(\theta_1\, t_1 \ldots \theta_n\, t_n)$", причем $t_1, \ldots, t_n$ имеют своими зачениями вхождения разрядов $\alpha_{1j_1}, \ldots, \alpha_{nj_n}$ в некоторые наборы $(\alpha_{11}, \ldots, \alpha_{1m_1}), \ldots,$ $(\alpha_{n1}, \ldots, \alpha_{nm_n})$ и оператор перечисляет наборы $(\beta_{1j_1-i}, \beta_{2j_2-i}, \ldots, \beta_{nj_n-i}); i = 0, 1, \ldots,$ $\min(j_1 - 1, \ldots, j_n - 1)$.

Для просмотра разрядов набора начиная с некоторой фиксированной позиции (слева направо) служит оператор "постпозиция$(t_1\, t_2)$". Он допускает два режима реализации:

а) t_1 - выходная переменная; t_2 определяет вхождение разряда α_i в набор $(\alpha_1 \ldots \alpha_n)$. Тогда t_1 перечисляет вхождения $\alpha_i, \alpha_{i+1}, \ldots, \alpha_n$.

б) t_1, t_2 определены и имеют значения α, β - вхождения в некоторый набор. Оператор проверяет, что вхождение α расположено не раньше, чем β.

Иногда более удобно применять похожий оператор "новпозиция$(t_1 t_2)$", у которого значением t_2 является набор $(\alpha_1 \ldots \alpha_n)$ либо вхождение разряда α_i в этот набор, а выходная переменная t_1 перечисляет: в первом случае - все вхождения $\alpha_1, \ldots, \alpha_n$, а во втором - $\alpha_{i+1}, \ldots, \alpha_n$.

В заключение перечислим ряд проверочных операторов для работы с наборами. Оператор "пересекаются$(t_1 t_2)$"проверяет наличие общего элемента в наборах либо термах (в последнем случае под элементом понимается логический символ либо символ переменной); операторы "равнойдлины$(t_1 t_2)$"и "короче$(t_1 t_2)$ равенство длин наборов и тот факт, что длина набора t_1 меньше длины набора t_2. Оператор "различны(t)"проверяет попарное отличие друг от друга всех элементов набора. Если значением выражения t_1 является набор α, а значением выражения t_2 - N-й после "0"логический символ, то оператор "длинатекста$(t_1 t_2)$"истинен, если α имеет длину N, а оператор "длинаменее$(t_1 t_2)$ если длина α меньше, чем N.

Наконец, оператор "включается$(t_1 t_2)$"проверяет, что набор - значение выражения t_1 получается перестановкой и исключением части разрядов набора - значения выражения t_2.

5.2.5 Арифметические операторы

Арифметические операторы ЛОСа делятся на две группы: для работы с символьным представлением чисел (число N кодируется N-м после "0"логическим символом) и с обычными десятичными представлениями . В последнем случае цифры $0, \ldots, 9$ кодируются логическими символами "0", ..., "9"; неодноразрядное положительное целое

число, имеющее десятичную запись $a_1 \dots a_n$, кодируется набором логических символов, соответствующих цифрам $a_1, \dots, a_n$; для представления десятичных дробей внутри этого набора на соответствующей позиции размещается логический символ ","(специальный символ для запятой); для получения отрицательных чисел в начале набора добавляется логический символ "минус". Заметим, что все десятичные числа, кроме цифр, кодируются наборами логических символов, а цифры кодируются самими логическими символами.

Для работы с символьным представлением чисел используются операторные выражения "плюссимв(t_1 t_2)", "вычитсимв(t_1 t_2)", "умножсимв(t_1 t_2)"(сложение, вычитание и умножение). Заметим, что в ЛОСе логический символ "0"имеет номер 260, и логические символы с номерами от 1 до 259 могут использоваться как символьные представления отрицательных целых чисел от -259 до -1. В интерпретаторе ЛОСа предусмотрена возможность для работы с 65535 логическими символами, т.е. наибольшее допустимое в символьном представлении целое число есть 65275. Обычно символьные представления чисел используются для таких технических целей, как нумерация разрядов наборов, представление весов посылок и условий задач, нумерация переменных, задание координат точек на экране, и т.п. Указанные выше диапазоны изменения символьных представлений чисел для этого вполне достаточны.

Если значениями t_1 и t_2 служат переменные, то значением выражения "вычитсимв ($t_1 t_2$)"служит символьное представление разности номеров этих переменных; если значением t_1 является переменная x, а значением t_2 - символьное представление числа n, то значением выражений "плюссимв($t_1 t_2$)", "вычитсимв($t_1 t_2$)"служит переменная, номер которой получается, соответственно, увеличением либо уменьшением на n номера переменной x.

Оператор "частноесимв(t_1 t_2 t_3 t_4)"позволяет находить неполное частное t_3 и остаток t_4 от деления числа t_1 на t_2 (все представления здесь - символьные). Оператор "менее($t_1 t_2$)"истинен, если число t_1 в символьном представлении меньше числа t_2. Для перехода от одного представления чисел к другому служат операторные выражения "симвномер(t)"(логический символ с десятичным номером, определяемым выражением t) и "номерсимв(t)"(десятичный номер логического символа, определяемого выражением t).

Для работы с десятичным представлением служат операторные выражения "минус(t)", "плюс($t_1 t_2$)", "вычитание($t_1 t_2$)", "умножение($t_1 t_2$)", и оператор "меньше (t_1 t_2)". Для реализации деления десятичных чисел применяются программы, написанные на ЛОСе; в этих программах можно использовать вспомогательные операторы "блокделения($t_1 t_2 t_3 t_4$)" (t_1, t_2 - делимое и делитель; выходным переменным t_3, t_4 присваиваются неполное частное и остаток, причем оператор рассчитан только на случай, когда t_1 и t_2 не превосходят 65535) и "шагделения($t_1 t_2 t_3$)" (t_1, t_2 - целые неотрицательные; $t_1 \leq 1000$; $1 < t_2 \leq 100$; выходной переменной t_3 присваивается целая часть от деления t_1 на $t_2 + 1$). Здесь приведены лишь базисные операторы ЛОСа, непосредственно реализуемые его интерпретатором; для вычислений в десятичных числах на ЛОСе запрограммирована целая серия дополнительных операторов, которые будут указаны в последующих разделах. Отметим, что вычисления в десятичных представлениях на ЛОСе выполняются с "плавающим"числом разрядов; это число ограничено сверху лишь общими требованиями интерпрератора к длине формируемых наборов и может достигать даже тысячи знаков. Приведенные выше базисные операторы сложения, вычитания,умножения и изменения знака выполняются без округления; программно реализованные на ЛОСе вычислительные

операторы выполняют округление с сохранением заданного числа знаков после запятой, причем таким образом, что гарантируется получение нижней либо верхней (по выбору) оценки для точного значения.

Для вычислений в машинных форматах "с плавающей запятой"и "целое со знаком"используются группа операторных выражений "выч(...)"и операторов "Выч(...)". Заметим, что кроме непосредственного представления чисел в этих форматах, можно создавать массивы таких чисел. Манипулирование с массивом осуществляется при помощи ссылки на него, которая формально представляет собой некоторый набор длины 2 (как и представление числа в машинном формате, такой набор несколько отличается от обычного набора ЛОСа, и для работы с ним следует применять только указанные выше операторные выражения и операторы "выч","Выч").

Операторное выражение "выч(число a)"позволяет перейти от обычного десятичного числа a (цифры либо набора цифр и знаков "минус","запятая") к формату "с плавающей запятой"; операторное выражение "выч(целое a) к формату "целое со знаком"; операторное выражение "выч(Целое a)" - к формату "длинное целое со знаком"; операторное выражение "выч(комплексное a b)" преобразует в формат "комплексное с плавающей запятой" комплексное число с вещественной частью a и мнимой частью b. Обратные преобразования осуществляются операторным выражением "выч(выход a)". Здесь a - число в произвольном формате из перечисленных выше. Если оно представлено в формате "с плавающей запятой", то значением выражения служит пара десятичных чисел (A_1A_2), такая, что рассматриваемое число равно произведению A_1 на 10 в степени A_2 (по мере возможности, A_2 выбирается равным 0). Для комплексного числа выдается пара таких пар. В прочих случаях значением выражения служит десятичное число.

Чтобы извлекать элементы массива, используется выражение "выч(значение M i)". Здесь M - ссылка на массив одного из машинных форматов (кроме длинного целого); i - номер элемента массива, представленный в формате "целое со знаком". Значением выражения является представление в машинном формате i-го элемента массива.

Операции над числами в одинаковых машинных форматах реализуются выражениями "выч(плюс a b)", "выч(минус a)"(изменение знака), "выч(вычитание a b)", "выч(умножение a b)", "выч(дробь a b)"(в случае целых чисел берется неполное частное), "выч(степень a b)"(в случае целых чисел b должно быть неотрицательным), "выч(модуль a)". Для вычислений в формате "с плавающей запятой"используются также выражения "выч(логарифм a b)" (a - основание логарифма), "выч(натурлог a)"(натуральный логарифм), "выч(синус a)", "выч(косинус a)", "выч(тангенс a)", "выч(котангенс a)", "выч(арксинус a)", "выч(арккосинус a)", "выч(арктангенс a)", "выч(арккотангенс a)", "выч(квадркорень a)" (квадратный корень), "выч(эксп a)" (экспонента), "выч(гипсинус a)" (гиперболический синус), "выч(гипкосинус a)", "выч(гиптангенс a)", "выч(гипкотангенс a)".

Операторные выражения "выч(веществennаячасть a)" и "выч(мнимаячасть a)" определяют вещественную и мнимую части комплексного числа, представляя их в формате "с плавающей запятой". Выражение "выч(Комплексное a)" переводит в комплексный формат число a, находящееся в формате "с плавающей запятой". Выражение "выч(Модуль a)" определяет представленный в формате "с плавающей запятой" модуль комплексного числа a.

Для получения псевдоконстант при построении графиков используются выражения "выч(плюсбеск)" ($2e99$ - плюс-бесконечность), "выч(минусбеск)" ($-2e99$), "выч(

нет)" ($1e99$) - указатель на пропуск точки при построении графика).

Копирование представления числа в машинном формате выполняется при помощи выражения "выч(копия a)".

Чтобы создать (неинициализированный) массив из n чисел в машинном формате, используется оператор "Выч(набор a n x)". Здесь a - указатель на тип чисел (логический символ "число с плавающей запятой; "целое целое со знаком). Переменной x присваивается ссылка на созданный массив. Для регистрации в массиве m в качестве элемента с номером n (нумерация начинается с 0) числа a в соответствующем машинном формате применяется оператор "Выч(запись m n a)".

Для деления с остатком чисел типа "целое со знаком"служит оператор "Выч(деление m n p q)". Здесь m, n - делимое и делитель; переменной p присваивается неполное частное, q - остаток.

Сравнение чисел осуществляется операторами "Выч(меньше a b)"и "Выч(меньшеилиравно a b)", "Выч(равно a b)". Чтобы проверять отличие числа от 0 , служит оператор "Выч(0 a)"(истинен, если a не равно 0).

Оператор "Выч(изменение a b)"изменяет старое представление числа a, перенося в него значение числа b.

Чтобы ускорить вычисления по цепочке действий, не требующих условных переходов, предусмотрено создание микропрограммы - набора P (как структуры данных ЛОСа) псевдокоманд; такая микропрограмма выполняется за одно обращение к оператору "Выч(программа P A B)". Здесь A - набор вход-выходных данных типа "с плавающей запятой"; B - набор вход-выходных данных типа "целое со знаком". Если вычисления выходят за рамки о.д.з., то оператор ложен. Каждая псевдокоманда предствляет собой набор $(Nn_1 \dots n_k)$, где N - номер операции, $n_1, \dots, n_k$ - номера операндов (входных и выходных). $N, n_1, \dots, n_k$ имеют символьный формат. Номера операндов берутся из набора A либо B, соответствующего типу операнда (такой тип однозначно определяется по номеру операции). Можно выходить за рамки наборов A, B, используя номера, большие их длины (но не большие 200). Такие номера рассматриваются как вспомогательные переменные, которым присваиваются значения, используемые в дальнейших вычислениях по микропрограмме. Нумерация элементов наборов A, B начинается с 1.

Рассматриваются следующие типы псевдокоманд (номер операции совпадает с номером в приводимом списке):

1. Сложение в формате "с плавающей запятой". Первые два операнда - слагаемые, третий - сумма (аналогичное размещение операндов используется и для других операций).

2. Сложение в формате "целое со знаком".

3. Изменение знака числа в формате "с плавающей запятой".

4. Изменение знака числа в формате "целое со знаком".

5. Умножение в формате "с плавающей запятой".

6. Умножение в формате "целое со знаком".

7. Деление в формате "с плавающей запятой".

8. Деление с остатком для формата "целое со знаком".

9. Возведение в степень в формате "с плавающей запятой".

10. Экспонента.

11. Возведение в натуральную степень числа типа "целое со знаком".

12. Модуль в формате "с плавающей запятой".

13. Модуль в формате "целое со знаком".

14. Натуральный логарифм.

15. Логарифм - общий случай.

16. Квадратный корень.

17. Синус.

18. Косинус.

19. Тангенс.

20. Котангенс.

21. Секанс.

22. Косеканс.

23. Арксинус.

24. Арккосинус.

25. Арктангенс.

26. Арккотангенс.

27. Гиперболический синус.

28. Гиперболический косинус.

29. Гиперболический тангенс.

30. Гиперболический котангенс.

31. Сигнум.

32. Тождественная операция (присвоение значения выходной переменной).

5.2.6 Операторы интерфейса

Клавиатура, мышь и меню

Для ввода символов с клавиатуры используется оператор "клавиатура(t)". Здесь t - выходная переменная, которой присваивается логический символ, кодирующий нажатую клавишу. Для того, чтобы определить в процессе написания программы на ЛОСе логический символ, соответствующий той или иной клавише, достаточно, находясь в текстовом редакторе решателя, нажать сначала клавишу F8, а затем - ту клавишу, код которой определяется; при этом с позиции курсора будет прорисовано название соответствующего логического символа. Драйвер клавиатуры, связанный с оператором "клавиатура", имеет следующие регистры:

а) Регистры больших и малых латинских букв;

б) Регистры больших и малых русских букв;

в) Регистры "левый Ctr"и "правый Ctr".

Переключение между латинскими и русскими буквами в этом драйвере выполняется при помощи клавиш F11,F12. Нажатие клавиши F11 переводит драйвер в режим русских букв; F12 - в режим латинских букв. Кроме того, имеется оператор "русшрифт(t_1)", позволяющий переключать драйвер клавиатуры: при t_1 = "1" - на русский шрифт; при t_1 = "0" - на латинский.

При нажатии левой либо правой клавиши мыши оператор "клавиатура(t)"получает логический символ "мышь". Для уточнения данных, введенных при помощи мыши, после получения указанным образом логического символа "мышь", используется оператор "мышь($t_1t_2t_3$)". Его выходные переменные t_1, t_2, t_3 получают в качестве значений, соответственно, указатель нажатой клавиши (логический символ "0" - нажата левая клавиша; "1" - нажата правая клавиша), номер столбца, на котором при нажатии клавиши находился курсор мыши, и номер строки, где он находился. Эти номера имеют символьное представление (т.е. суть логические символы с номерами $260+N$, где N - номер столбца либо строки).

Экранные операции

Выдача изображений на экран осуществляется при помощи оператора "видео(M t_1 $\ldots t_n$)", где M - логический символ, определяющий тип экранной операции; $t_1, \ldots, t_n$ - входные данные этой операции и ее выходные переменные. Параметр n меняется в зависимости от M.

Подготовка области экрана к прорисовке изображения выполняется при помощи операции "видео(прямоугольник $t_1t_2t_3t_4t_5$)". Здесь t_1, t_2 - номера столбца и строки для верхнего левого угла прямоугольника; t_3, t_4 - номера столбца и строки для правого нижнего угла (все номера здесь и в рассматриваемых ниже экранных операциях - в символьном представлении). t_5 - логический символ, определяющий цвет прямоугольника. Для указания цвета используются логические символы начиная с "1"до "шестнадцать": 1 - черный; 2 - синий; 3 - зеленый; 4 - зеленовато-голубой; 5 - красный; 6 - пурпурный; 7 - коричневый; 8 - светло-серый; 9 - темно-серый; 10 - светло-синий; 11 - светло-зеленый; 12 - светло-голубой; 13 - светло-красный; 14 - светло-пурпурный; 15 - желтый; 16 - белый. Если t_5 равно "0", то изображение на экране внутри указанного прямоугольника не изменяется. В любом случае после применения операции "видео(прямоугольник $\ldots$)"границы прямоугольника становятся границами той области, внутри которой перечисляемыми ниже текстовыми операциями выполняется выдача текста (в частности, автоматически осуществляется переход к новой строке по достижении правой границы).

Для прорисовки группы отрезков применяется операция "видео(отрезок t_1t_2)". Здесь t_1 - набор $a = (a_1 \ldots a_s); a_i = (a_{i1}a_{i2}a_{i3}a_{i4})$, где a_{i1}, a_{i2} - позиция (номера столбца и строки) начала i - го отрезка; a_{i3}, a_{i4} - позиция конца этого отрезка. t_2 - цвет всех выдаваемых на экран отрезков.

Дуга окружности изображается при помощи операции "видео(окружность $t_1 \ldots$ t_9)". Здесь t_1, t_2 - координаты центра окружности (номера столбца и строки); t_3 - радиус окружности (в символьном представлении); t_4, t_5 - координаты точки, через которую проходит исходный радиус дуги; t_6, t_7 - координаты точки, через которую проходит заключительный радиус дуги; t_8 - направление прорисовки (логический символ "0 против часовой стрелки; "1 по часовой стрелке); t_9 - цвет окружности.

Текстовые фрагменты выдаются на экран решателя при помощи его собственного шрифта, в котором каждый символ представляется матрицей размера 8×19. Интерпретатор ЛОСа нетрудно было бы расширить таким образом, чтобы стало доступным использование стандартных шрифтов, однако при этом потребуется существенная коррекция программ формульного редактора, планирующего компоновку формул с учетом геометрических параметров символьных элементов. В интерфейсе решателя предусмотрена возможность изменения шрифта; кроме того, операторы ЛОСа позволяют вводить и использовать различные наборы шрифтов указанного выше размера. При работе решателя в оперативной памяти создается специальная область (далее называем ее видеокассой), хранящая двоичные матрицы символьных элементов. Изменение содержимого видеокассы выполняется при помощи операции "видео(бланк $t_1 t_2$)", где t_1 - логический символ, определяющий прорисовываемую букву (соответствие - то же, что для драйвера клавиатуры); t_2 - набор $(a_1 \dots a_{19})$; $a_i = (a_{i1} \dots a_{i8})$ - набор из логических символов "0"и "1". Этот оператор заносит двоичную матрицу, определяемую набором t_2, на позицию видеокассы, соответствующую логическому символу t_1. Чтобы прочитать матрицу изображения буквы, хранящейся в видеокассе, применяется операция "видео(развертка $t_1 t_2$)". Здесь t_1 - логический символ, определяющий букву; выходной переменной t_2 присваивается набор длины 19, состоящий из восьмиэлементных наборов логических символов "0"и "1" - требуемая матрица изображения. При запуске решателя видеокасса загружается из вспомогательного файла; для сохранения ее измененной версии в этом файле служит специальный оператор (см. ниже операторы работы с файлами решателя).

При выдаче на экран фрагмента текста начальная позиция либо задается в операциях явным образом, либо используются координаты из указателя текущей позиции, который хранит некоторые значения номеров текущих столбца и строки. Для занесения в указатель текущей позиции новых значений t_1 и t_2 номеров столбца и строки используется операция "видео(позиция t_1 t_2)". Для получения хранящихся в указателе номеров используется операция "видео(точка $t_1 t_2$)". По окончании выдачи текстовой операцией фрагмента текста указатель текущей позиции автоматически устанавливается на первую свободную после фрагмента позицию. Как уже отмечалось выше, при достижении правой границы текущего прямоугольника (т.е. того, который был указан при последнем обращении к операции "видео(прямоугольник ...)") происходит автоматическая смена строки. Каких-либо автоматических вставок знаков переноса (по крайней мере на уровне базисных операторов) при этом не предусмотрено.

Выдача буквы с заданной позиции выполняется операцией "видео(буква t_1 t_2 t_3 t_4 t_5)". Здесь t_1, t_2 - номера столбца и строки позиции, с которой выдается буква; t_3 - логический символ, кодирующий выводимую на экран букву; t_4 - цвет фона; t_5 - цвет символа. Если буква выдается с текущей позиции, то можно положить в качестве t_1 логический символ "продолжение"; в качестве t_2 - логический символ "0".

Для работы с фрагментами текстов введен специальный накопитель, называемый далее буфером текстов. Он представляет собой массив из 2000 ячеек, каждая из которых хранит некоторый экранный символ (фактически - номер этого символа в видеокассе, т.е. число от 0 до 255), а также цвет фона и цвет символа. Буфер текстов используется для обменов с файлами, хранящими текстовую информацию, а также для временного хранения текстовых фрагментов (например, ранее выданных на экран и требующих в определенных ситуациях повторной выдачи, быть может,

с изменением цветовых атрибутов). Большинство из приводимых далее операций, выдающих на экран текстовые фрагменты, автоматически заносят эти фрагменты в заданную область буфера текстов. При ссылке на ячейки буфера текстов они нумеруются числами от 0 до 1999, причем эти номера имеют символьное представление.

Операция "видео (логсимвол $t_1\,t_2\,t_3\,t_4\,t_5\,t_6\,t_7$)"выводит на экран название логического символа t_3. При этом t_1, t_2 суть номера исходных столбца и строки (при выдаче с текущей позиции - "продолжение"и "0"); t_4, t_5 - цвет фона и символов; t_6 - номер первой ячейки буфера, с которой в него заносится название логического символа; t_7 - выходная переменная, которой присваивается номер первой ячейки буфера текстов, идущей после данного названия. Если на экране либо в буфере текстов не хватило места, то оператор ложен. В этом последнем случае прорисовка на экране начала названия логического символа не выполняется.

Для выдачи на экран подтерма заданного терма в скобочной записи используется операция "видео(терм $t_1\,t_2\,t_3\,t_4\,t_5\,t_6\,t_7\,t_8$)". Здесь t_1, t_2 - номера столбца и строки исходной позиции (либо логические символы "продолжение", "0"); t_3 - вхождение первого символа выдаваемого на экран подтерма; t_4 - указатель раскраски. Этот указатель представляет собой набор $(a_0, \ldots, a_s)$, где $a_0 = (a_{01}, a_{02})$ - цвет фона и символов, используемые при прорисовке подтерма; $a_i = (a_{i1}, a_{i2}, a_{i3}), i = 1, \ldots, s$ - указывает, что подтерм рассматриваемого подтерма, вхождение первого символа которого есть a_{i1}, следует раскрасить цветом фона a_{i2} и цветом символов a_{i3}. Вхождения $a_{11}, \ldots, a_{s1}$ упорядочены слева направо, причем соответствующие подтермы могут пересекаться (т.е. быть вложенными друг в друга). Возможность раскраски подтермов существенным образом используется в редакторе программ ЛОСа, так как позволяет создавать многоцветную "указку", выделяющую при просмотре сложных логических конструкций цепочку вложенных термов и облегчающую концентрацию внимания и декомпозицию этих конструкций при чтении программ. Значение t_5 используется в тех случаях, когда весь терм не поместился на выделенную область экрана. Тогда для продолжения выдачи оставшейся части терма (например, после нажатия соответствующей клавиши) входной параметр t_5 полагается равным тому вхождению символа, с которого следует продолжать выдачу подтерма. В исходной ситуации, с начала выдачи подтерма, значение t_5 должно быть равно t_3. t_6 - номер исходной ячейки в буфере текстов, начиная с которой в буфер заносятся выводимые на экран символы. t_7, t_8 - выходные переменные: t_7 - индикатор переноса: если выдача всего подтерма успешно завершена, то он получает значение "0"; иначе - его значением становится вхождение того символа, с которого следует продолжать выдачу на экран. t_8 становится равно номеру первой ячейки в буфере текстов, следующей после заключительного символа подтерма. Если прорисовка терма на экране нежелательна, а требуется лишь зарегистрировать последовательность выводимых на экран символов в буфере текстов и при этом получить некоторую дополнительную информацию о соответствии вхождений в терм позициям буфера текстов, то в указателе раскраски полагается $a_{01} = a_{02}$. В этом случае, если t_1 положить равным номеру некоторой позиции в буфере текстов, то t_7 становится равно вхождению в подтерм логического символа либо переменной, к прорисовке которого относится указанная позиция. Значение t_2 в указанной ситуации игнорируется. Если в буфере текстов не хватило места, то оператор ложен.

Для выдачи набора термов можно использовать операцию "видео(серия t_1 t_2 t_3 $t_4\,t_5\,t_6$)", которая реализуется быстрее, чем последовательное выполнение операций прорисовки отдельных термов. Здесь t_1, t_2 - номера столбца и строки исходной пози-

ции (либо "продолжение", "0");t_3, t_4 - цвет фона и символов; t_5 - набор термов. Термы этого набора последовательно прорисовываются на экране начиная с исходной позиции; буфер текстов при этом заполняется начиная с нулевой ячейки. Выходной переменной t_6 присваивается набор той же длины, что t_5. На позиции этого набора, соответствующей позиции некоторого терма в наборе t_5, располагается пара номеров исходной и заключительной ячеек буфера текстов, содержащих текст данного терма.

Для выдачи на экран содержимого заданного отрезка буфера текстов используется операция "видео(слово $t_1\, t_2\, t_3\, t_4\, t_5$)". Здесь t_1, t_2 - номера столбца и строки исходной позиции на экране (либо "продолжение", "0"); t_3 - номер ячейки буфера текстов, начиная с которой выдается текст; t_4 - номер ячейки, до которой включительно происходит выдача (если t_4 - логический символ "продолжение", то выдача происходит вплоть до первой ячейки, хранящей "машинный ноль" либо, если такой нет, до конца буфера текстов). Выходная переменная t_5 является индикатором переноса: "0" - нет переноса, иначе - номер ячейки буфера текстов, с которой следует продолжать выдачу.

Операция "видео(запись $t_1\, t_2\, t_3\, t_4$)"позволяет занести в ячейку буфера текстов, имеющую номер t_1, букву t_2 (т.е. t_2- кодирующий эту букву логический символ)с цветом фона t_3 и цветом символа t_4. В случае $t_1 = 0; t_2 = t_3 =$ "1"в указанную ячейку заносится "машинный ноль", который воспринимается рядом операций как указатель на конец используемой части буфера текстов. Операция "видео(значение $t_1\, t_2\, t_3\, t_4$)"присваивает выходной переменной t_2 логический символ, обозначающий букву, расположенную в t_1-й ячейке буфера текстов; при этом значения выходных переменных t_3, t_4 становятся равны, соответственно, цвету фона и цвету символа, хранящимся в ячейке. Операция "видео(0)"заполняет весь буфер текстов "машинными нулями". Операция "видео(пустоеслово $t_1 t_2$)" заполняет "машинными нулями" отрезок буфера текстов начиная с t_1-й ячейки и кончая t_2-й ячейкой. Для изменения цветовых параметров хранящихся в буфере текстов символов (без изменения их изображения на экране) служат операции "видео(актив $t_1\, t_2\, t_3\, t_4$)"и "видео(вхождение $t_1\, t_2\, t_3$)". У них t_1, t_2 - номера первой и последней ячеек изменяемого отрезка буфера текстов. В первом случае цвет фона всех букв этого отрезка заменяется на t_3, а цвет символов - на t_4; во втором случае изменяется на t_3 только цвет фона.

Для определения границ текущего прямоугольника служит операция "видео(лимит $t_1\, t_2\, t_3\, t_4$)", присваивающая своим выходным переменным t_1, t_2, t_3, t_4, соответственно, номера столбца и строки верхнего левого угла прямоугольника и номера столбца и строки правого нижнего угла. Сдвиг изображения внутри текущего прямоугольника выполняется операцией "видео(спуск $t_1\, t_2\, t_3$)". Здесь логический символ t_1 указывает направление сдвига: "префикс" - вниз; "суффикс" - вверх; "правсосед" - вправо; "левсосед" - влево. t_2 - число (в символьном представлении) пикселей, на которые сдвигается изображение; t_3 - цвет чистых полос, возникающих при сдвиге.

Для сохранения текущего экрана в буфере битмэпов используется операция "видео(минус)"; восстановление экрана по содержимому данного буфера выполняется операцией "видео(плюс)", причем буфер битмэпов после этого обнуливается. Для временного сохранения содержимого буфера текстов введены два дополнительных буфера текстов. Операция "видео(копия t_1)" при t_1 равном "0" копирует буфер текстов в первый дополнительный буфер; при "3" - во второй; при "2" - обменивает содержимое буфера текстов и первого дополнительного буфера; при "1" - копирует первый дополнительный буфер в буфер текстов; при "4" - копирует второй дополнительный буфер в буфер текстов.

В дополнение к буферу текстов, для сохранения текстовых заготовок в оперативной памяти с целью быстрой выдачи их на экран введен так называемый массив текстов. Главным образом, он используется в текст-формульном редакторе. Размеры массива текстов составляют 56000 ячеек; он позволяет сохранять извлекаемые из буфера текстов фрагменты и возвращать сохраненные фрагменты в буфер текстов для различных экранных операций.

Создание массива текстов осуществляется оператором "видео(начало)"; удаление его - оператором "видео(конец)". Для пересылки в конец массива текстов фрагмента буфера текстов начиная с позиции m и кончая позицией n, используется оператор "видео(образ m n i j)". Его выходным переменным i, j присваиваются, соответственно, символьные номера первой и последней ячейки массива текстов, отведенные для сохраненного фрагмента. Для обратной пересылки служит оператор "видео(прообраз m n k)". Здесь m, n - символьные номера первой и последней ячейки фрагмента массива текстов, k - номер ячейки буфера текстов, начиная с которой размещается фрагмент, извлеченный из массива текстов.

Чтобы расчищать массив текстов, служит оператор "видео(сброс m)", устанавливающий указатель номера первой неиспользованной ячейки этого массива на символьный номер m. Если нужно изменить ранее записанный в массив текстов фрагмент, то применяется оператор "видео(заменавхождения m n k p)". Символьные номера m, n здесь указывают начало и конец изменяемого фрагмента; заменяющий фрагмент берется из буфера текстов - начиная с ячейки k и кончая ячейкой p. Если $p < k$, то происходит просто исключение фрагмента массива текстов. Если длина заменяющего фрагмента не равна длине заменяемого, то выполняется соответствующий сдвиг всех фрагментов массива текстов, расположенных после измененного. Оператор "видео(вычеркивание m n)"исключает фрагмент массива текстов начиная с позиции m до позиции n.

Оператор "видео(См m n k p)"позволяет определять находящуюся в массиве текстов на позиции m букву n с цветом фона k и цветом символа p. Оператор "видео(посылка m n k p)", наоборот, заносит на позицию m массива текстов букву n с цветом фона k и цветом символов p.

Для организации поиска в массиве текстов служит оператор "видео(ключ m n k p)". По заданным начальной и концевой позициям m, n и букве k он перечисляет все номера позиций p, на которых располагается эта буква.

Для прорисовки графиков функций одной переменной используется оператор "видео(график A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9)". Здесь A_1 - ссылка на массив числовых данных в формате с плавающей запятой (ординаты точек графика); A_2 - длина этого массива (десятичное число); A_3, A_4 - нижняя и верхняя границы отображаемого на экране диапазона значений (формат с плавающей запятой); A_5, A_6 - верхняя и нижняя полосы для графика (символьные числа); A_7 - столбец, начиная с которого рисуется график (символьное число); A_8, A_9 - цвет фона и цвет линии. Масштабирование таково, чтобы нижняя и верхняя границы диапазона значений соответствовали нижней и верхней полосам графика. Прямоугольник, выделенный для графика, должен иметь количество столбцов, не меньшее A_2.

Для небольших изображений можно непосредственно на ЛОСе создавать битмэпы, прорисовывать их и сохранять в информационных блоках. Первоначально битмэп формируется в виде набора $(A_1, \ldots, A_m)$ наборов $A_i = (B_{i1}, \ldots, B_{in})$; каждое B_{ij} - логический символ "0"либо "1". Здесь $A_1, \ldots, A_m$ - строки, перечисляемые сверху

вниз; в каждой строке элементы перечисляются слева направо. Далее битмэп перекодируется в более компактную структуру данных - некоторый набор логических символов (двоичные знаки выписываются последовательно и делятся на отрезки длины 15; каждый тако отрезок образует номер очередного логического символа, уменьшенный на 1). Такие наборы называем кодами двоичных матриц. Перекодировка выполняется оператором "видео(матрица логсимвол a b)", у которого a - исходное представление битмэпа, b - его код. Обратный переход от кода b к явному представлению битмэпа a осуществляется оператором "видео(матрица набор b m n a)", у которого появляются дополнительные параметры m (число строк) и n (число столбцов). Наконец, собственно прорисовка битмэпа по его двоичному коду b осуществляется оператором "видео(матрица видео p q r s b m n)". Здесь p, q - столбец и строка, определяющие верхний левый край прямоугольника, в котором нужно прорисовать двоичную матрицу; r, s - цвет нулевых и единичных элементов; m, n - число строк и столбцов.

Названия логических символов

В структурах данных решателя логические символы представлены своими номерами (от 1 до 65535), или, точнее, четырехбайтными словами, содержащими эти номера. Для определения номера логического символа по его названию либо названия символа по его номеру используется специальная таблица, хранящаяся в файлах вида Vi.lsi; $i = 1, \ldots, 7$. Каждый такой файл рассчитан на хранение названий 10000 символов. Для хранения названия используется 24-байтное слово, так что предельно допустимая длина названия равна 24. В качестве названия допускаются произвольные последовательности русских либо латинских букв, цифр и спецзнаков. Большие и малые буквы в названии различаются. В начале файла размещаются двухбайтные номера всех логических символов, названия которых он хранит, причем эти номера расположены в лексикографическом для соответствующих названий порядке (эти номера занимают ровно 20000 байт. За ними располагаются сами названия, строго по порядку номеров символов. Если для символа еще не введено название, то соответствующие 24 байта заполнены нулями.

Для работы с названиями логических символов служит оператор "словарь(M t_1 $\ldots$ t_n)". Здесь M - логический символ, определяющий тип выполняемой операции; $t_1, \ldots, t_n$ - набор входных данных либо выходных переменных. Для ввода нового логического символа с заданным названием (точнее, присвоения этого названия некоторому логическому символу, еще не имеющему названия) служит операция "словарь(запись t_1)". Перед обращением к ней в начальном отрезке буфера текстов должно быть сформировано требуемое название, после которого размещается "машинный ноль". Если логический символ с таким названием уже имеется, то оператор ложен; в противном случае выбирается некоторый логический символ, не имевший названия, ему присваивается указанное в буфере текстов название, и выходной переменной t_1 присваивается номер выбранного логического символа. Если не удалось найти символа без названия, то t_1 присваивается "0". Для определения названия заданного логического символа служит операция "словарь(значение t_1)". Если t_1 - логический символ, то в начальный отрезок буфера текстов заносится слово, обозначающее этот символ. При отсутствии названия данного символа первая ячейка буфера текстов обнуливается. Изменение ранее введенного названия логического символа выполняется операцией "словарь(изменение)". Перед обращением к ней в буфер текстов заносится слово вида "A0B0 ...", где A - заменяемое название; B - заменяющее

название; 0 - "машинный ноль". Если B уже использовано либо A не использовано, то оператор ложен; в первом случае название A удаляется,но B не вводится. Проверка существования логического символа с заданным названием осуществляется операцией "словарь(проверка t_1)". Если существует логический символ, название которого хранится в начальном отрезке буфера текстов, то выходной переменной t_1 присваивается этот символ, иначе оператор ложен. Для удаления хранящегося в начальном отрезке буфера текстов названия логического символа служит операция "словарь(исключение)".

Наконец, упомянем оператор "кодтекста(t_1 t_2 t_3)", используемый для перевода текстов термов либо наборов термов в сами термы либо наборы термов. Указанный текст размещается в начальном отрезке буфера текстов. Если t_1 есть "0", то этот текст транслируется в терм; если t_1 есть "1", то текст транслируется в набор термов; если t_1 есть "2", то текст транслируется в набор логических символов и термов (т.е. каждый однобуквенный терм рассматривается как символ); если t_1 есть"3", то текст транслируется в единственный логический символ. Если трансляция выполнена успешно, то выходная переменная t_3 получает значение "0", а выходной переменной t_2 присваивается результат трансляции. Если же в тексте обнаружена ошибка, то t_3 указывает на ее тип:

1) t_3 = "2" - после номера переменной идет посторонний символ. Тогда t_2 - ссылка на начало переменной в буфере текстов.

2) t_3 = "3" - номер переменной больше 511. Тогда t_2 - начало переменной в буфере текстов.

3) t_3 = "4" - неправильно набран логический символ; t_2 - его начало в буфере текстов.

4) t_3 = "5" - нехватка места для формирования результата.

5) t_3 = "6" - избыточная правая скобка; t_2 - ссылка на начало текста символа, после которого идет эта скобка.

6) t_3 = "8" - незакрытая левая скобка.

Словарь текстового анализатора

Для работы с текстами естественного языка используется словарь текстового анализатора, хранящий 65535 фрагментов слов (корни, окончания, суффиксы, приставки). Длина одного словарного фрагмента не должна превышать 24 букв. Каждому словарному фрагменту этого словаря (далее называемого внешним словарем) сопоставлен кодирующий его логический символ. При чтении слова оно разбивается на фрагменты, и решателю передаются лишь коды фрагментов.

Для регистрации во внешнем словаре нового словарного фрагмента сначала происходит размещение этого фрагмента в некотором отрезке буфера текстов, и далее выполняется оператор "внешсловарь(запись s m n)", где m, n - номера начальной и последней ячеек отрезка; s - логический символ, который становится кодом фрагмента. Символ s мог уже являться кодом некоторого словарного фрагмента; в этом случае старая версия фрагмента изменяется во внешнем словаре на новую. Чтобы получить словарный фрагмент, закодированный некоторым логическим символом, применяется оператор "внешсловарь(значение s m p q n)". У него s - логический символ, кодирующий фрагмент; m - номер ячейки буфера текстов, начиная с которой размещается словарный фрагмент; p,q - цвет фона и символов для размещаемого в

буфере текстов фрагмента. Выходной переменной n передается номер первой ячейки буфера текстов после фрагмента. Если символ s не являлся кодом словарного фрагмента, то оператор ложен.

Чтобы исключить из внешнего словаря словарный фрагмент, кодируемый логическим символом s, применяется оператор "внешсловарь(исключение s)".

Для чтения слова и разбиения его на словарные фрагменты применяется оператор "поиксслова(m s n)". В качестве входного данного он получает номер m позиции в буфере текстов, начиная с которого размещается еще не прочитанная часть слова. Выходная переменная s перечисляет логические символы, являющиеся кодами словарных фрагментов, расположенных в буфере текстов начиная с позиции m. При этом переменная n перечисляет номера первых идущих после указанных словарных фрагментов позиций буфера текстов. Окончательный отбор разбиения слова на словарные фрагменты выполняется уже с помощью программы, реализованной на ЛОСе.

Имеется возможность просмотра всех словарных фрагментов внешнего словаря, начинающихся с заданной последовательности букв. Для этого служит оператор "текслово(m n s)". Он получает в качестве входных данных номера m, n первой и последней ячеек буфера текстов, между которыми хранится указанная последовательность букв. Выходная переменная s перечисляет коды словарных фрагментов, начинающихся с данной последовательности. Перечисление происходит в лексикографическом порядке.

Информационные блоки

Логическая и текстовая информация, используемая решателем, может быть размещена в специальным образом организованных файлах. Эти файлы разбиты на группы, называемые далее информационными блоками. Информационный блок хранит древовидную конструкцию, образованную объектами следующих типов:

а) Корневой указатель - каталог. Этот объект фактически представляет собой таблицу ссылок на другие объекты информационного блока, длина которой равна числу логических символов, т.е. 65535. Если на i - й позиции этой таблицы находится 0, то ссылка по логическому символу с номером i из данного указателя считается неопределенной, иначе - определен переход из указателя по i-му символу к некоторому объекту информационного блока.

б) Указатель - список. Этот объект представляет собой набор $((a_1, S_1), \ldots (a_n, S_n))$ пар, имеющих попарно различные первые элементы a_i; каждый такой элемент (называемой меткой перехода в данном указателе) представляет собой либо логический символ, либо неоднобуквенный терм. S_i есть набор ссылок на объекты информационного блока, к которым осуществляется переход по метке a_i.

в) Логический терминал. Этот объект представляет собой набор логических символов, переменных и неоднобуквенных термов (однобуквенные термы при записи их в логический терминал автоматически преобразуются в логические символы либо символы переменных).

г) Текстовый терминал. Этот объект представляет собой просто последовательность байт, кодирующих текст (в зависимости от способа хранения текста, между байтами,

кодирующими символы, могут вставляться байты, кодирующие их цветовые атрибуты).

В действительности структура ссылок в информационном блоке "почти"древовидная, так как разрешаются ссылки на один и тот же терминал из различных указателей.

Интерпретатор ЛОСа обеспечивает работу с 16 информационными блоками, имеющими номера от 1 до 16. За ними закреплена определенная функциональная нагрузка, которая будет уточняться далее по мере описания структуры и функционирования решателя. Корневым объектом блоков с номерами 1 и 4 является указатель - список; каждый из этих блоков представляет собой единственный файл: для блока 1 это файл I0000.lsi; для блока 4 - файл R0000.lsi. Остальные блоки имеют своим корневым объектом указатель - каталог и состоят из группы файлов. Все файлы одного и того же информационного блока имеют вид Xijkl.lsi, где X - буква, соответствующая номеру блока; ijkl - восьмеричный номер файла в блоке. Нумерация файлов сплошная и начинается с 0000. Указатель-каталог в информационном блоке может быть лишь корневым; он занимает полностью файл X0000.lsi. Буквенная кодировка информационных блоков (кроме уже указанных 1-го и 4-го) такова: E - 2; H - 3; Q - 5; M - 6; W - 7; P - 8; T - 9; A - 10; B - 11; C - 12; D - 13; F - 14; G - 15; J - 16. Каждый из файлов информационного блока имеет не более 4 Мб; по мере увеличения размеров этих файлов происходит автоматическое разрезание их на две приблизительно одинаковые по размеру части, со сдвигом нумерации последующих за разрезаемым файлов. Все объекты, достижимые из корневого каталога по заданному логическому символу (т.е. "ветвь"этого логического символа в информационном блоке), размещаются в одном и том же файле, так что переходы через указатель - список возможны только в рамках одного файла.

Для работы с информационными блоками используются операторы вида "файл($M\,t_1 \ldots t_n$)"и "указатель($M\ t_1 \ldots t_n$)". Здесь M - логический символ, определяющий тип выполняемой операции; $t_1, \ldots, t_n$ - набор входных данных либо выходных переменных. Операции применяются к заранее выделенному "активному"информационному блоку. Выделение информационного блока с номером t_1 (номер - в символьном представлении) вместо ранее выделенного осуществляется при помощи операции "файл(актив t_1)". Если информационного блока с таким номером еще не было, то он при этом будет введен (пока без корневого объекта). Проверка наличия информационного блока с номером t_1 выполняется оператором "файл(проверка t_1)". Операция "файл (число t_1)" присваивает выходной переменной t_1 номер активного информационного блока.

Для ссылок на объекты активного информационного блока в ЛОСе служат наборы (a_1, a_2), образованные двумя логическими символами a_1, a_2. Эти символы кодируют некоторым образом (подробнее см. в описании интерпретатора ЛОСа) номер файла информационного блока и смещение в данном файле начала рассматриваемого объекта. Ссылкой на корневой объект произвольного информационного блока служит пара ("0","3").

Для перехода по заданной метке в корневом указателе - каталоге служит операция "указатель(логсимвол $t_1\ t_2\ t_3$)", где t_1 - ссылка на каталог (т.е. набор ("0","3")); t_2 - логический символ, представляющий собой метку перехода. Если переход по данной метке в каталоге не определен, то оператор ложен. Иначе - выходной переменной t_3 присваивается ссылка на объект, к которому имеет место переход по метке t_2.

Для перехода по заданной метке в указателе - списке служит операция "указатель(список $t_1\ t_2\ t_3$)". Здесь t_1 - ссылка на рассматриваемый указатель-список; t_2 - метка перехода (логический символ либо неоднобуквенный терм). Если в данном указателе-списке не предусмотрен переход по метке t_2, то оператор ложен. Иначе выходной переменной t_3 присваивается набор ссылок на объекты, переход к которым осуществляется по данной метке. Заметим, что порядок ссылок в наборе - обратный, т.е. первой идет ссылка, которая была зарегистрирована (с помощью указываемых далее операций) последней.

Чтобы определить множество всех меток перехода, используемых в указателе- списке, применяется операция "указатель(область $t_1\ t_2$)". У нее t_1 - ссылка на указатель; выходной переменной t_2 присваивается набор всех меток перехода в данном указателе. Ввод нового указателя (каталога либо списка) осуществляется операцией "указатель(новый $t_1\ t_2\ t_3\ t_4\ t_5$)". Здесь t_1 - логический символ "логсимвол" (если вводится корневой каталог - в случае пустого информационного блока) либо "список"(если вводится указатель - список). Если вводится корневой указатель, то $t_2 = t_3 =$ "0"; иначе t_2 - ссылка на внешний указатель, а t_3 - метка перехода, по которой в нем регистрируется новый указатель. В случае ввода каталога t_4 игнорируется, а в случае указателя - списка t_4 есть длина в байтах поля, зарезервированного для нового пустого указателя. При превышении данной длины регистрация в указателе новых ссылок связана с переписыванием в файле всего указателя на новой позиции, а до этого регистрация выполняется без переписывания указателя. Для оценки величины t_4 уточним, что хранение n ссылок по одной метке M требует $3n + 3m + 2$ байт, где m - число логических символов и "самостоятельных"(не идущих непосредственно за логическим символом либо символом переменной) закрывающих скобок в M. Выходной переменной t_5 присваивается ссылка на новый указатель.

Исключение объекта, на который имеется ссылка из некоторого указателя (каталога либо списка), осуществляется операцией "указатель(исключение $t_1\ t_2\ t_3$)". Здесь t_1 - ссылка на данный указатель; t_2 - метка, по которой из него достижим исключаемый объект. Если указатель является каталогом, то t_3 несущественно, иначе t_3 - ссылка на исключаемый объект. Происходит удаление ссылки из указателя по метке t_2 на исключаемый объект. Если этот объект предствлял собой указатель-список, то вся его ветвь автоматически удаляется из информационного блока. Сохраняются лишь те ее терминалы, на которые имеются ссылки из указателей, не относящихся к данной ветви. Если исключаемый объект - терминал, то он удаляется из информационного блока лишь при условии, что на него не было других ссылок.

Ссылка на объект может быть извлечена из одного указателя и перенесена в другой указатель. Это выполняется операцией "указатель(перестановка $t_1\ t_2\ t_3\ t_4\ t_5$)". Здесь t_1 - ссылка на первый указатель (каталог либо список); t_2 - метка, по которой из него имеется ссылка на переносимый объект. Если первый указатель является каталогом, то t_3 несущественно, иначе t_3 - ссылка на переносимый объект. t_4 есть ссылка на второй указатель (заметим, что он может совпадать с первым); t_5 - метка, по которой переносимый объект требуется зарегистрировать во втором указателе. Если указатели и метки совпадают, то просто происходит изменение порядка ссылок по рассматриваемой метке (новая ссылка заносится в начало списка ссылок). Важно заметить, что оператор применим лишь тогда, когда оба указателя относятся к одному и тому же файлу информационного блока (это гарантируется для блоков с номерами, отличными от 1 и 4, лишь при условии, что оба они относятся к ветви одного и того же логического символа в корневом каталоге). Если это не так, то для перенесения

объекта (фактически - ветви информационного блока) требуется создать его копию и удалить оригинал.

Если некоторый терминал информационного блока уже создан и необходимо создать дополнительную ссылку на него, то применяется операция "указатель(терминал $t_1\,t_2\,t_3$)". Здесь t_1 - ссылка на указатель, из которого создается дополнительная ссылка; t_2 - метка этой ссылки; t_3 - ссылка на терминал. Как и в предыдущем случае, новый указатель должен быть расположен в том же файле информационного блока, что и терминал.

Для определения типа объекта информационного блока служит операция "указатель(тип $t_1\,t_2$)". Здесь t_1 - ссылка на объект. Выходной переменной t_2 присваивается логический символ, указывающий тип объекта: "логсимвол" - каталог; "список" - указатель-список; "терминал" - текстовый терминал; "терм" - логический терминал.

Если все метки некоторого указателя - списка суть логические символы (как правило, номера в символьном представлении), то для одновременного увеличения либо уменьшения на заданную величину всех его меток, больших или равных заданной, служит операция "указатель(плюссимв $t_1\,t_2\,t_3$)". Здесь t_1 - ссылка на указатель; t_2 - метка, начиная с которой происходит смещение номеров; t_3 - логический символ, определяющий константу сдвига. Если он больше "0"(т.е. его номер больше 260), то номера увеличиваются, иначе - уменьшаются. Эта операция позволяет модифицировать указатель без переписывания его в файле.

Перечислим операции, используемые для чтения хранящихся в файле терминалов. Операция "файл(терм $t_1\,t_2$)" по ссылке t_1 на логический терминал присваивает выходной переменной t_2 набор записанных в этом терминале логических символов, символов переменных и неоднобуквенных термов. Если t_1 - ссылка на текстовый терминал, в котором хранится текст с пропущенными цветовыми атрибутами, то операция "файл(набор $t_1\,t_2\,t_3\,t_4\,t_5$)" переносит этот текст в буфер текстов начиная с ячейки данного буфера, имеющей номер (в символьном представлении) t_4. При этом все символы получают одинаковые цвет фона t_2 и цвет символов t_3. Выходной переменной t_5 присваивается номер последней использованной при чтении ячейки буфера текстов. Если же текстовый терминал хранит текст с явно указанными цветовыми атрибутами, то для его чтения используется операция "файл(слово $t_1\,t_2\,t_3$)", где t_2 - номер начальной ячейки буфера текстов; t_3 - выходная переменная, которой присваивается номер последней использованной ячейки. Если текстовый терминал по ссылке t_1 хранит таблицу видеокассы, то для загрузки ее в видеокассу используется операция "файл(видео t_1)".

Ввод нового терминала осуществляется операцией "файл(запись $t_1\,t_2\,t_3\,t_4\,t_5$)". Здесь t_1 - логический символ, определяющий тип вводимого объекта: "терм" - логический терминал; "слово" - текстовый терминал с явно указанными цветовыми атрибутами; "набор" - текстовый терминал с пропущенными цветовыми атрибутами; "видео" - текстовый терминал, хранящий таблицу видеокассы. Если создается логический терминал, то t_2 - набор логических символов, символов переменных и термов (допускаются и однобуквенные термы, однако в терминал они записываются как символы, и при чтении из терминала восстанавливаются как символы). В остальных случаях значение t_2 несущественно. Содержимое нового текстового терминала извлекается из буфера текстов начиная с нулевой ячейки и кончая указателем конца текста ("машинный нуль") либо концом буфера. t_3 - ссылка на указатель (каталог либо список),

в котором регистрируется новый терминал; t_4 - метка, по которой он регистрируется. Выходной переменной t_5 присваивается ссылка на новый терминал.

Изменение содержимого терминала выполняется при помощи операции "файл (изменение $t_1\,t_2\,t_3\,t_4$)". Здесь t_1 - логический символ, определяющий тип терминала таким же образом, как в операции "файл(запись ...)"; если изменяется логический терминал, то t_2 задает новый набор символов и термов, иначе t_2 несущественно. Как и при вводе терминала, информация для изменения текстовых терминалов берется из начального отрезка буфера текстов. t_3 - ссылка на изменяемый терминал. Выходной переменной t_4 присваивается ссылка на измененный терминал. Так как изменение терминала связано с переписыванием его новой версии в конце файла (старая версия снабжается специальной пометкой, причем ее поле "портится" - используется для хранения ссылки на новую версию), то при последующей работе с терминалом следует использовать только ссылку t_4.

При изменении указателей и терминалов, хранящихся в информационном блоке, старые их версии (снабженные специальными пометками) накапливаются и приводят к дополнительному увеличению размеров файла. Кроме того, так как переход к новой версии объекта осуществляется по цепочке ссылок, хранящихся в старых его версиях, может несколько замедляться работа программ. Поэтому предусмотрена процедура "уплотнения"информационного блока, переписывающая измененные его файлы без старых версий объектов и выполняюшая необходимую коррекцию всех ссылок (смещений в фале) из указателей. Эта же процедура, при усмотрении превышающего 450 Кбайт файла информационного блока (кроме блоков 1 и 4), осуществляет разрезание его на две примерно равные части, с увеличением на 1 номеров последующих файлов данного блока. Обращение к указанной процедуре выполняется операцией "файл(уплотнение)". После выполнения ее информационный блок перестает быть активным. Если t_1 - ссылка на некоторый объект информационного блока (кроме корневого каталога), который был изменен, и таким образом содержит ссылку на новую его версию, та - ссылку на ее новую версию, и т.д., то ссылка на концевой объект данной цепочки (т.е. на "реальную"текущую версию объекта) присваивается операцией "файл(ссылка t_1 t_2)"выходной переменной t_2. Операция "файл(выписка t_1)"присваивает выходной переменной t_1 набор номеров всех информационных блоков, для которых, возможно, требуется уплотнение (т.е. в некотором файле блока имеется хотя бы одна "отключенная"версия объекта).

Блок программ ЛОСа

Программа решателя, записанная на ЛОСе, хранится в группе файлов, называемой далее блоком программ. Эти файлы имеют вид Lijkm, где i,j,k,m - восьмеричные цифры. Файл L0000 хранит каталог программ - в нем для каждого логического символа, имеющего свою программу, указана ссылка на корневой фрагмент данной программы. Такая ссылка состоит из номера файла Lijkm (отличного от L0000) и смещения в этом файле начала фрагмента (более подробно формат данных в файлах блока программ приведен в описании интерпретатора ЛОСа). Все фрагменты программы одного и того же логического символа хранятся в одном файле блока программ. Так как программы языка ЛОС реализуются интерпретатором, то существенно облегчено изменение программой своих собственных фрагментов непосредственно в процессе работы решателя. Разумеется, здесь должны соблюдаться определенные ограничения на изменение тех фрагментов программ, которые на текущий момент

"активизированы", т.е. начато их выполнение и в стэках интерпретатора хранятся значения программных переменных, связанных с этими фрагментами. Однако, эти фрагменты продублированы в оперативной памяти, так что даже их при определенных условиях можно изменить. Например, реализованный на ЛОСе программный редактор позволяет изменять все без исключения фрагменты программ, в том числе свои собственные (последние изменения требуют особой аккуратности, во избежание необратимой порчи блока программ). Для обеспечения возможности восстановления файлов решателя при аварийных ситуациях в его интерфейсе предусмотрена возможность копирования полного комплекта его файлов в резервную директорию. Перед копированием выполняется проверка корректности структур данных в этих файлах (как в блоке программ, так и в информационных блоках); при обнаружении нарушений происходит выход в операционную систему, а копирование не выполняется.

Для работы с фрагментами программ в ЛОСе используется оператор "прогфайл $(M\, t_1 \ldots t_n)$". Здесь M - логический символ, определяющий тип выполняемой операции; $t_1, \ldots, t_n$ - набор входных данных либо выходных переменных. Ссылка на фрагмент программы в блоке программ представляет собой пару логических символов. Более подробно структура таких ссылок будет указана ниже, при описании интерпретатора (она несущественна при использовании приводимых далее операций). Операция "прогфайл(логсимвол $t_1\, t_2\, t_3$)"позволяет по заданному логическому символу t_1 находить ссылку (t_2, t_3) на начало корневого фрагмента программы символа t_1; если такой программы нет, то оператор ложен. Операция "прогфайл(значение $t_1\, t_2\, t_3$)"по ссылке (t_1, t_2) на фрагмент в блоке программ присваивает выходной переменной t_3 набор операторов этого фрагмента (каждый оператор, в том числе односимвольный, представляется термом). Операторы "ветвь N"и "иначе N"в данном наборе представлены как "ветвь$(a_1 a_2)$"; "иначе$(a_1 a_2)$", где (a_1, a_2) - ссылка на тот фрагмент, переход к которому определяется оператором.

Изменение ранее введенного фрагмента программы либо ввод нового фрагмента выполняются операцией "прогфайл(запись $t_1\, t_2\, t_3\, t_4\, t_5\, t_6$)". Здесь t_4 - набор операторов (термов), образующих новый фрагмент. В нем могут встречаться в указанном выше формате ссылки на другие фрагменты программы (напомним, что повторные ссылки на один и тот же фрагмент программы не разрешаются, так что те фрагменты, на которые имеются ссылки из нового фрагмента, либо должны быть непосредственнымии подфрагментами заменяемого фрагмента ,либо должны быть заранее "отключены"при помощи операции "прогфайл(вычеркивание ...)"; см. ниже). Кроме того, допускаются термы вида "ветвь(m)"; "иначе(m)", у которых m - логический символ, представляющий собой метку недоопределенного перехода из фрагмента. Такие недоопределенные переходы, после записи их в блок программ, восстанавливаются в том же виде операцией "прогфайл(значение ...)". Если новый фрагмент заносится в качестве корня программы некоторого логического символа f (возможно, уже имевшейся до этого), то t_1 есть символ "0"; $t_2 = f$; значение t_3 несущественно. Иначе t_1 равно "1"; (t_2, t_3) есть ссылка на вхождение оператора "ветвь" либо "иначе" в тот внешний фрагмент, который через данный оператор должен ссылаться на новый фрагмент. Под ссылкой на вхождение оператора перехода "ветвь" либо "иначе" в некоторый фрагмент здесь понимается пара (A_1, A_2), у которой A_1 - ссылка на фрагмент (пара логических символов); A_2 - номер (начиная с 1) данного оператора перехода в фрагменте. Если новый фрагмент F заносится вместо некоторого другого фрагмента G, то G и все ветви фрагмента G, не упомянутые в F, удаляются. По

окончании регистрации в блоке программ нового фрагмента выходным переменным t_5, t_6 присваивается ссылка на этот фрагмент.

Если фрагмент F и всю его ветвь требуется временно отключить для последующего использования в другом месте программы, то применяется операция "прогфайл (вычеркивание $t_1\, t_2\, t_3\, t_4$)". Если указанный фрагмент F является корнем программы некоторого логического символа f, то t_1 есть "0"; $t_2 = f$; t_3 - несущественно. Иначе t_1 равно "1"; (t_2, t_3) - ссылка на вхождение оператора "ветвь"либо "иначе"во внешний фрагмент, по которому происходит переход к фрагменту F. Если t_1 есть "0", то из каталога программ исключается ссылка по символу f; иначе - ссылка из внешнего фрагмента на F заменяется меткой t_4 недоопределенного перехода (логическим символом). В обоих случаях исключаются только ссылки на фрагмент F, а сам он и вся его ветвь сохраняются в блоке программ без изменений.

Для удаления ветви ранее отключенного фрагмента программы используется операция "прогфайл(исключение $t_1\, t_2$)". Здесь (t_1, t_2) - ссылка на данный фрагмент. Этот фрагмент и все достижимые из него фрагменты снабжаются в блоке программ специальными пометками. Аналогичным образом, операция "прогфайл(сброс t_1)" уничтожает всю программу логического символа t_1. Операция "прогфайл(корень $t_1\, t_2\, t_3$)"подключает ранее отключенную ветвь фрагмента, ссылкой на который является пара (t_2, t_3), в качестве программы логического символа t_1 (ранее имевшаяся программа этого символа удаляется).

При замене фрагмента программы на новую его версию старая версия снабжается специальной пометкой, но сохраняется в файле блока программ. Для того, чтобы периодически "расчищать" блок программ от накапливающихся в нем неиспользуемых отрезков, используется операция "прогфайл(уплотнение)". Она инициирует перезапись фрагментов программ без сохранения указанных отрезков, с соответствующей коррекцией ссылок между фрагментами. Так как при этом полностью нарушается корректность "старых"ссылок из регистров интерпретатора и из хранящихся в оперативной памяти копий фрагментов программы на фрагменты в блоке программ, то после указанного уплотнения блока программ происходит автоматический перезапуск решателя (это выглядит как возвращение к исходному меню). Заметим, что процедуре уплотнения подвергаются не все файлы блока программ, а лишь те из них, которые были изменены после последнего уплотнения. Как и при уплотнении информационных блоков, происходит разрезание пополам тех файлов, которые превысили 450 Кбайт. Заметим, что процедура уплотнения сначала предпринимает проверку корректности блока программ, и лишь после этого начинает собственно уплотнение. При обнаружении ошибки эта процедура выдает значение "истина" (в отличие от процедуры уплотнения информационных блоков), а в случае успешного уплотнения - значение "ложь".

Если произошло изменение программы некоторого логического символа, причем перезапуск решателя нежелателен (например, в цикле логического вывода и автоматической генерации приемов), то для устранения рассогласования между копиями фрагментов программы, хранящимися в оперативной памяти, и измененными фрагментами в блоке программ, используется операция "прогфайл (компонента t_1)". Эта операция удаляет из оперативной памяти все копии фрагментов программы логического символа t_1 и восстанавливает в копии каталога программ, хранящейся в оперативной памяти, ссылку на корневой фрагмент этой программы по блоку программ. Операция "прогфайл(пересмотр)"уничтожает вообще все хранящиеся в оперативной памяти решателя копии фрагментов программ и реализует его перезапуск.

Приведем в заключение ряд достаточно редко используемых операций над блоком программ. Операция "прогфайл(позиция $t_1\,t_2\,t_3\,t_4\,t_5\,t_6$)"по ссылке (t_1, t_2) на фрагмент программы; набору t_3 операторов этого фрагмента и вхождению t_4 в этот набор оператора "ветвь" либо "иначе" присваивает выходным переменным t_5, t_6 ссылку на вхождение данного оператора в рассматриваемый фрагмент (см. выше формат таких ссылок при описании операции "прогфайл(запись ...)"). Эта операция сохранилась от старой версии интерпретатора, в которой формат ссылок был иным; требуемую ссылку теперь легко получить простым подсчетом количества операторов перехода, предшествующих данному. Операция "прогфайл(продолжение $t_1\,t_2\,t_3$)"используется для поиска недоопределенных переходов. У нее t_1 - логический символ, в программе которого ищутся такие переходы. Находится первый такой переход "ветвь A" либо "иначе A"; здесь A - логический символ, являющийся меткой недоопределенного перехода. Определяется путь $(F_1, \ldots, F_s)$ от корневого фрагмента F_s программы символа t_1 к тому фрагменту F_1, в котором найден указанный переход (т.е. каждый фрагмент F_{i+1} имеет ссылку на F_i; $i = s-1, \ldots, 1$). Выходной переменной t_2 присваивается метка A; переменной t_3 - набор $(a_1, \ldots, a_s)$ пар логических символов - ссылок на фрагменты $F_1, \ldots, F_s$. Операция "прогфайл(имя $t_1\,t_2\,t_3$)"позволяет по заданным логическому символу t_1 и оператору t_2 (терму) находить первый фрагмент программы символа t_1, содержащий данный оператор. При этом выходной переменной t_3 присваивается такой же набор ссылок, определяющих путь к найденному фрагменту, как и для предыдущей операции.

Операции трассировки

Для отладки программ языка ЛОС используется программа логического символа "прерывание", написанная на ЛОСе. Эта программа реализована как программа фиктивного операторного выражения (именно, однобуквенного терма "прерывание"), причем данное операторное выражение в явном виде в программах ЛОСа не встречается, а обращение к указанной программе выполняется интерпретатором автоматически - при обнаружении тех или иных ошибочных действий либо при выполнении условий обрыва, заданных используемым режимом трассировки. Программа символа "прерывание" реализует обычные операции отладчика - позволяет устанавливать требуемый режим трассировки, просматривать программу текущего и внешних операторов, определять значения программных переменных, и т.п. Более подробно об этой программе будет рассказано в специальном разделе книги. Для использования в отладчике был введен ряд базисных операций ЛОСа, объединенных в операторе "трассировка($M\,t_1 \ldots t_n$)". Эти операции существенным образом используют специфику применяемого интерпретатора ЛОСа, поэтому их описание будет приведено в разделе, посвященном программе отладчика ЛОСа.

Глава 6

Редактор программ ЛОСа

6.1 Перечень названий операторов ЛОСа

Для поиска нужного оператора ЛОСа создано оглавление, в котором базисные и основные реализованные на ЛОСе операторы и операторные выражения ЛОСа упорядочены по своему назначению. Это оглавление достижимо из главного меню ЛОСа при нажатии клавиши "о"(кир.); соответствующий пункт находится в правом нижнем углу. В него также можно войти из редактора программ ЛОСа (см. ниже). Через концевой пункт оглавления осуществляется выход (нажатием клавиши "курсор вправо") в справочную информацию о логическом символе, являющемся заголовком соответствующего оператора либо операторного выражения. Интерфейс работы с такой справочной информацией уже был описан выше. Заметим, что в данном случае появляется возможность перехода к просмотру корневого фрагмента программы текущего логического символа - для этого достаточно нажать клавишу "Home". Для возвращения из просмотра программы нажимается "End".

Оглавление операторов ЛОСа можно пополнять самостоятельно, используя для этого следующие действия. После создания программы для нового оператора либо операторного выражения следует найти подходящий раздел оглавления либо ввести новый такой раздел; в нем создать концевой пункт с краткой характеристикой действий, выполняемых новой программой. Далее - нажать клавишу "Enter". Текст пункта окажется перерисованным в верхней части экрана; под ним будет проведена горизонтальная линия. После повторного нажатия "Enter"под данной линией возникнет курсор текстового редактора, с помощью которого следует ввести название нового оператора (операторного выражения). После завершения ввода (еще одно "Enter") нажимается "курсор влево", и новый концевой пункт создан.

6.2 Интерфейс редактора программ

Как уже говорилось выше, программа на ЛОСе представляет собой дерево фрагментов, переходы между которыми выполняются с помощью операторов перехода "ветвь N", "иначе N". Каждый фрагмент является последовательностью операторов ЛОСа. Редактор программ ЛОСа осуществляет постраничный показ фрагментов программы, так что каждый фрагмент целиком умещается на экране. Никаких прокруток изображения на экране при этом не предусмотрено. Если фрагмент слишком

большой и целиком на экране не помещается, то он разрезается на части, между которыми имеются линейные переходы: в конце очередной части помещаются операторы "ветвь N", "продолжение", где N - номер перехода к следующей части. Операторы размещаются в программе, отделенные друг от друга пробелами; никаких других разделяющих знаков быть не должно. Их последовательность не форматируется; при автоматической перерисовке операторы идут друг за другом с промежутками ровно в один пробел. Если строка заканчивается, то текст продолжается с начала следующей строки без каких-либо пробелов и знаков переноса. Разумеется, такой стиль изображения программы отличается от общепринятого. Однако, плотный режим размещения операторов имеет свои достоинства: удается разместить больший объем информации на меньшем пространстве, что упрощает анализ контекста при написании программы и ее чтении. Решающим фактором, сделавшим плотный режим размещения операторов практически приемлемым и даже удобным, оказалась специальная многоцветная указка, которая, по существу, заменяет форматирование, позволяя концентрировать внимание на нужном участке текста и определяя архитектуру его включения в целый текст. Разумеется, возможно создать более традиционные способы изображения ЛОС-программы на экране - если они обеспечат большую эффективность программирования, чем настоящая версия. Впрочем, уже сейчас подавляющее большинство приемов записывается не на ЛОСе, а на языке значительно более высокого уровня - ГЕНОЛОГе. Поэтому более перспективным направлением, чем совершенствование интерфейса редактора программ ЛОСа, представляется (естественное для логической системы) развитие средств автоматического создания ЛОС - программ.

6.2.1 Вход в редактор программ ЛОСа

Вход в редактор программ ЛОСа через главное меню обычно осуществляется одним из двух способов - с помощью явного указания логического символа, программу которого нужно просматривать либо редактировать, либо через оглавление программ ЛОСа, в котором представлены основные блоки системы или большие вспомогательные процедуры общего назначения.

Для входа в корневой фрагмент программы логического символа A нужно либо нажать клавишу "п", либо нажать левую клавишу мыши, переведя ее курсор в окно главного меню "Просмотр программы логического символа". В обоих случаях в синей рамке, расположенной внутри этого окна, появится курсор текстового редактора. Далее нужно набрать название символа A и нажать "Enter". При наборе последовательности букв, не являющейся названием логического символа, синяя рамка снова становится пустой, и набор следует повторить либо нажать "Esc". При правильном наборе символа на экране появится изображение корневого фрагмента программы символа A. Если такой программы еще не было создано, то в верхней части экрана будет перерисовано название символа A, а остальная часть экрана будет пустой.

Для входа в оглавление основных программ ЛОСа из главного меню нужно либо нажать клавишу "л", либо переместить курсор мыши в окно "Оглавление программ"и нажать левую клавишу мыши. После этого на экране возникает подменю оглавления программ, которое рассматривалось перед последним выходом из этого оглавления. Неконцевые пункты оглавления программ соответствуют основным блокам либо вспомогательным процедурам системы, или подблокам таких блоков и процедур. Фактически ветвь оглавления программ является вынесенной в отдельную

структуру данных иерархической системой комментариев к программе. Концевые пункты этой ветви жестко связаны с конкретными точками в программе; выйдя на такой пункт и нажав клавишу "курсор вправо", можно оказаться в соответствующей контрольной точке программы. Эта точка будет обозначена в прорисованном на экране фрагменте программы выделенным синим цветом фиктивным оператором "прием(N)"; N - последовательность цифр, образующая номер контрольной точки. Нумерация контрольных точек - своя для каждого логического символа. Для возвращения в оглавление из просмотра фрагмента программы следует нажать "End". Для перехода к просмотру надфрагмента либо подфрагментов, либо для изменения фрагмента нужно выйти из режима просмотра подтермов операторов фрагмента, в котором мы оказываемся после нажатия "курсор вправо", нажав клавишу "o". Заметим, что возвращение в оглавление может быть осуществлено из любого фрагмента просматриваемой программы (причем в тот концевой пункт оглавления, из которого был сделан переход к рассмотрению программы). Однако, для такого возвращения нужно сначала восстановить режим просмотра подтермов операторов (снова нажатием клавиши "o"). Способ регистрации в оглавлении программ ЛОСа новых контрольных точек описывается ниже.

В принципе, возможен еще один способ входа в просмотр фрагментов программы - через оглавление операторов ЛОСа. Это оглавление (как уже говорилось выше) достижимо из главного меню по клавише "o"(выбор пункта в правом нижнем углу экрана). После перехода в концевой пункт такого оглавления и нажатия клавиши "курсор вправо"появляется справочная информация о логическом символе - заголовке оператора. При нажатии "Home"здесь осуществляется переход к просмотру корневого фрагмента программы данного логического символа (но уже не в режим просмотра подтермов операторов, а в обычный режим просмотра). После просмотра и редактирования этого либо других фрагментов той же программы возможно возвращение в оглавление операторов - по нажатии клавиши "End".

Наконец, имеется возможность входа в просмотр программы приема, заданного на ГЕНОЛО́ГЕ - об этом будет сказано в разделе, посвященном редактору ГЕНОЛОГа.

6.2.2 Просмотр фрагментов программы

Перемещение по дереву фрагментов программы

При просмотре фрагмента программы переходы из этого фрагмента в подфрагменты занумерованы числами $1, 2, \ldots$. Каждый такой переход представляет собой оператор "ветвь N"либо "иначе N", где N - номер перехода. Один из этих переходов ("текущий") выделен желтым цветом. Используя клавиши "курсор вправо" (к следующему переходу) и "курсор влево" (к предыдущему переходу), можно выбрать нужный текущий переход. После этого нажатие клавиши "курсор вниз" переводит в просмотр подфрагмента, к которому ведет данный переход. Последующее нажатие клавиши "курсор вверх" вызовет возвращение к исходному фрагменту.

Если клавиша "курсор вверх" нажимается в корневом фрагменте, то происходит возвращение в главное меню (именно, в режим ввода логического символа, программу которого требуется просмотреть). Нажатие клавиши "PageUp" из произвольного фрагмента программы логического символа также позволяет (причем за один шаг) попасть в указанный режим главного меню. Обычно эти возможности используются для перехода к просмотру программы другого логического символа.

Если вход в редактор программ ЛОСа имел место через указание логического символа в главном меню, а не из какого-либо оглавления или из редактора ГЕНОЛОГа, то нажатие клавиши "End"при просмотре любого фрагмента программы переводит в корневой фрагмент этой программы.

Для перемещения по дереву фрагментов программы с помощью мыши следует подвести курсор мыши к нужному оператору перехода в подфрагмент и нажать правую кнопку (нажатие левой кнопки переведет в режим просмотра подтермов). Чтобы вернуться из подфрагмента в надфрагмент, следует перевести курсор мыши в зону под текстом программы и также нажать правую кнопку.

Режим просмотра подтермов операторов фрагмента программы

Многие операторы ЛОСа представляют собой сложные многоэтажные логические конструкции, текст которых занимает иногда значительную часть экрана. Для того, чтобы облегчить концентрацию внимания на отдельном фрагменте такого оператора и, более того, сделать видимой его "архитектуру", используется специальная многоцветная указка - режим просмотра подтермов операторов. Вход в режим просмотра подтермов обеспечивается нажатием клавиши "о"(кириллица). При этом пропадает выделение желтым цветом текущего перехода к подфрагменту, но первый оператор фрагмента программы окрашивается в голубой цвет. Используя клавиши "курсор вправо"и "курсор влево", можно последовательно выделять все операторы текущего фрагмента.

После выбора отдельного, представляющего интерес, оператора, можно выделить новым (отличным от голубого) цветом один из его корневых операндов. Для этого достаточно нажать сначала на клавишу "курсор вниз", и далее клавишами "курсор вправо"и "курсор влево"обеспечить выбор требуемого корневого операнда. Повторяя эту операцию для уже выделенного на некотором уровне операнда (снова нажимая клавишу "курсор вниз", и т.д.), можно войти в просмотр подоперандов этого операнда. Чтобы избежать смешения цветов, они чередуются в соответствии с некоторым фиксированным списком цветов (по достижении последнего в списке цвета снова используется первый).

Нажатие клавиши "курсор вверх"возвращает из просмотра подоперандов к внешней операции. Если уже достигнут уровень просмотра всего оператора в целом, то нажатие данной клавиши игнорируется.

Для возвращения из режима просмотра подтермов операторов к обычному режиму просмотра фрагмента программы достаточно повторно нажать клавишу "о".

Можно войти в режим просмотра подтермов, переведя курсор мыши к корню представляющего интерес подтерма и нажав левую кнопку мыши. Если нужно перейти к выделению другого подтерма - операция повторяется. Для выхода из режим просмотра подтермов можно перевести курсор мыши в зону под текстом программы и нажать левую кнопку.

Если при выделенном текущем подтерме, заголовком которого служит логический символ, нажать клавишу F3, то на экране появится справочная информация об этом логическом символе (интерфейс работы с ней уже был описан). Выход из просмотра этой информации - по любой клавише, кроме используемых для перелистывания и редактирования справочных текстов (например, по нажатию пробела).

Если нужно перейти к программе логического символа, выделенного при просмотре подтермов в текущем фрагменте программы, то нажимается клавиша "c". После этого появляется корневой фрагмент программы данного логического символа. Из этого фрагмента, далее, можно повторно перейти указанным образом к новому фрагменту, и т.д. По этой цепочке переходов можно возвращаться, нажимая клавишу "End" (при каждом возвращении устанавливается корневой фрагмент программы, а не тот, из которого имел место переход; восстановлению исходного фрагмента для первого элемента цепочки здесь иногда помогает нажатие клавиши F8 - например, если переход к нему был из оглавления программ ЛОСа либо из редактора ГЕНОЛОГа).

Справочная информация о логических символах

Чтобы при просмотре программы получить информацию о том, что делает тот или иной оператор, или что означает некоторое понятие, с которым работает прием, можно пользоваться справочной информацией о логических символах. Она размещена в 3-м информационном блоке, и фактически представляет собой записную книжку, в которой собрана вся необходимая техническая информация о логических символах - описания операторов, смысл применяемых в задачах обозначений, пояснения к использованию целей и комментариев задач, заголовком которых является данный логический символ, и т.п. По мере ввода новых символов и новых операторов, эту записную книжку следует регулярно пополнять новыми данными. Интерфейс просмотра и редактирования справочной информации о логическом символе уже описан выше в подразделе "Разное"раздела "Общие операторы интерфейса". Перечислим возможные способы обращения к этой информации из просмотра фрагмента программы.

Если требуется получить информацию о логическом символе, являющемся заголовком данной программы, то нажимается клавиша F3. Если нужна информация о каком-либо другом логическом символе, то следует нажать клавишу F2. Далее возникает окно диалога, в котором следует набрать название нужного символа и нажать "Enter".

Для получения справочной информации о логическом символе, встречающемся в текущем фрагменте программы, можно подвести курсор мыши к этому символу и нажать левую кнопку мыши. Тогда произойдет переход в режим просмотра подтермов операторов, в котором текущим окажется подтерм с корнем в выбранном логическом символе. Далее нажимается правая кнопка мыши, и на экране возникает справочная информация о символе. Для выхода из ее просмотра достаточно нажать любую кнопку мыши (кроме нажатия этой кнопки на пунктах меню $\rightarrow, \leftarrow$, которое вызовет перелистывание справочной информации). После этого можно либо убрать режим просмотра подтермов, выведя курсор мыши за рамки текста программы и нажав левую кнопку, либо перевести курсор к другому представляющему интерес логическому символу и нажать левую кнопку, чтобы соответствующий подтерм стал текущим.

Кроме справочной информации о логических символах, при просмотре фрагмента программы можно войти в оглавление операторов ЛОСа. Для этого достаточно нажать "Ctr-л". Напомним, что из главного меню к этому же оглавлению переход происходил по нажатию клавиши "о".

Поиск заданного терма в ветви программы логического символа

Для того, чтобы найти все вхождения в операторы текущей ветви программы заданного терма либо логического символа, следует нажать клавишу F4 и далее набрать в окне диалога текстовым редактором требуемый терм либо логический символ. После нажатия "Enter"будет либо прорисовано многоцветной указкой первое найденное вхождение, либо (если таких вхождений в ветви вообще нет) возникнет пустой экран; по нажатию любой клавиши от него - возвращение в просмотр фрагмента. При поиске вхождений, кроме текущего фрагмента программы, просматриваются все достижимые из него фрагменты. Здесь используется принцип "сначала вглубь", причем непосредственные подфрагменты любого фрагмента упорядочиваются "от начала к концу" (т.е. по убыванию локальных номеров).

Для перехода к следующему вхождению искомого терма (символа) повторно нажимается F4. Если найдено представляющее интерес вхождение, то можно (как обычно, нажатием клавиши "о") выйти из режима просмотра подтермов, автоматически установленного при поиске, и выполнить необходимые операции (например, по редактированию программы). При этом установка на поиск заданного терма (символа) сохраняется, однако при повторном нажатии F4 будет выполняться поиск только в той ветви, в которой это нажатие (первое в возобновленном цикле поиска) было осуществлено. Поэтому перед возобновлением цикла поиска следует сначала выйти на корень необходимой ветви программы.

Если после очередного нажатия F4 возникает пустой экран, то поиск завершен. Только после этого установка на заданный терм (символ) сбрасывается. Далее при нажатии любой клавиши - переход в обычный режим просмотра фрагментов программы.

Последовательный просмотр программ логических символов

Можно последовательно просматривать программы всех логических символов, начиная с заданного - в соответствии с нумерацией логических символов. Для перехода к корню программы очередного логического символа (первого после текущего, для которого создана программа) нажимается клавиша "ш". Кроме этого простейшего средства полного просмотра программ "вручную", имеется возможность автоматического просмотра всех программ системы, связанная с использованием специального оператора "смпрог". При полном просмотре всех программ и входящих в них операторов, ему передается информация о каждом текущем подоператоре. В зависимости от того, как будет запрограммирован оператор "смпрог", при этом может происходить поиск всех мест в программе, удовлетворяющих заданному условию, и изменение их по заданному принципу. Кроме того, будет выполняться фоновый (не зависящий от оператора "смпрог") поиск типовых ошибок в программах. Наличие такого режима автоматического просмотра сделало пока излишним развитие других средств поиска и контроля в программах ЛОСа; оно позволило обращаться со всем многообразием программ системы как с единым целым и вводить в них при необходимости любые изменения (в том числе, связанные с развитием языка ЛОС). Подробнее о режиме автоматического просмотра программы будет сказано ниже, после изложения необходимой для изменения оператора "смпрог"техники редактирования программ.

6.2.3 Редактирование текущего фрагмента программы

Для начала редактирования текущего просматриваемого (в обычном режиме) фрагмента программы следует нажать клавишу "p"(кир.). При этом в правом верхнем углу экрана появляется красный прямоугольник, предупреждающий о том, что в случае нажатия клавиши "Enter"новая версия текста фрагмента будет немедленно записана в файлы блока программ, а старая пропадет (в этом случае, все же, при необходимости можно будет извлечь из резервной директории SLCOPY все файлы ранее сохраненной в ней версии решателя). Редактирование осуществляется текстовым редактором.

Для завершения редактирования нажимается клавиша "Enter"; для отмены изменений фрагмента - "Esc". Чтобы изменения фрагмента были сохранены, в тексте не должно иметься незавершенных или неправильно набранных операторов. Попытка сохранить такой некорректный текст приводит обычно к появлению сообщения об ошибке, прорисовываемого внутри красного прямоугольника в правом верхнем углу. Курсор текстового редактора при этом перемещается либо в начало текста, либо к той точке, около которой произошла ошибка. После устранения ошибки попытка сохранения текста фрагмента может быть повторена. Предусмотрены следующие типы сообщений об ошибках:

1) "Незакрытая левая скобка" - число левых скобок в тексте больше числа правых. Курсор перемещается в начало текста.

2) "Избыточная правая скобка" - число правых скобок в некоторой точке превысило число предшествующих левых. Курсор перемещается к этой точке.

3) "Ошибка в логическом символе" - при наборе логического символа допущена ошибка либо произошла склейка идущих подряд названий символов или переменных. Курсор перемещается к началу неправильно набранного символа.

4) "Одинаковые метки" - несколько различных операторов перехода из фрагмента имеют один и тот же номер перехода. Курсор перемещается в начало текста.

5) "Недопустимая переменная" - номер переменной больше 512. Курсор перемещается в начало записи этой переменной.

Ссылки из фрагмента на подфрагменты нумеруются при редактировании произвольным образом натуральными числами. Впоследствии, при перерисовках фрагмента, они будут автоматически переобозначены на $1, 2, 3, \ldots$. Если изменяется старая версия фрагмента, то исключение оператора перехода, ссылающегося на подфрагмент, либо изменение номера его ссылки на новый, не использовавшийся ранее в фрагменте, приведут к полной потере всей ветви этого оператора.

Если в редактируемом фрагменте возникли новые номера перехода, то после нажатия клавиши "Enter" и регистрации в файлах блока программ новой версии фрагмента на экране появляется в верхнем левом углу один из новых номеров перехода (первый при движении от конца фрагмента к началу), а остальная часть экрана расчищается (остается только курсор текстового редактора). Это - приглашение к вводу подфрагмента с указанным номером. От него можно отказаться нажатием клавиши "Esc", а можно воспользоваться, введя новый подфрагмент. Этот подфрагмент, в свою очередь, может содержать ссылки на подфрагменты, и тогда снова появится приглашение к вводу последнего из них, и т.п. В результате, если не нажимать

"Esc", будут по принципу "сначала вглубь" перечислены все ссылки на новые подфрагменты. Здесь важно не использовать дважды одного и того же номера ссылки (в процессе перечисления новых подфрагментов нумерация ссылок - глобальная, но по завершении перечисления эти номера теряются и вводится локальная нумерация).

Если при наборе подфрагментов нажималась клавиша "Esc", то останутся недоопределенные переходы. Лучше всего доопределить их сразу, по завершении указанного выше перечисления подфрагментов. Для создания фрагмента по недоопределенному переходу, следует нажать на этом переходе клавишу "курсор вниз"(экран расчистится) и войти в его редактирование повторным нажатием "p". Вообще, предпочтительнее не нажимать при перечислении подфрагментов клавишу "Esc", а вводить какой-либо простой оператор для последующего возвращения к этому подфрагменту и фактического его набора (например, оператор "продолжение").

Исходный фрагмент программы любого оператора, справочника либо операторного выражения должен начинаться с оператора "метка(икс(N))", где N - символьный номер первой программной переменной, не определенной при обращении и не являющейся выходной переменной. Этот оператор должен размещаться сразу после операторов "программа"либо "обращение(t)", указывающих тип обращения к программе. В случае сканирования задачи (т.е. после оператора "решить") оператор "метка(икс(...))" не используется.

Чтобы получить справочную информацию о логическом символе непосредственно в процессе редактирования, нажимается клавиша F4 и в диалоговом окне вводится название этого символа. После просмотра информации восстанавливается изображение на момент прерывания редактирования. Если при наборе текста нужно ввести некоторый символьный номер N, больший двадцати (номера от 0 до 20 обозначены логическими символами "0", ..., "9", "десять", ..., "двадцать"), то вводится запись "логсимвол(N)". По окончании набора текста, после нажатия "Enter", эта запись будет автоматически переведена в требуемый символьный номер, и впоследствии в данном месте программы он будет прорисовываться в явном виде. Если нужно вставить в программу логический символ - код клавиатуры, то нажимается клавиша F8, после чего нажимается та клавиша, для которой нужен код. Этот код будет прорисован автоматически, начиная с текущей позиции курсора. Если нужен код для режима латинских букв, то предварительно нажимается F12; для возвращения к режиму кириллицы нажимается F11.

Для копирования части фрагмента программы в другие фрагменты можно использовать буфер текстового редактора: войти в режим редактирования фрагмента, содержащего нужный текст (клавиша "p"); занести этот текст в буфер; выйти из режима редактирования, не изменяя программы (клавиша "Esc"); войти в редактирование того фрагмента, в котором нужно вставить копию, и извлечь ее из буфера текстового редактора. При этом, однако, не будут скопированы те фрагменты, к которым из выделенного текста имелись переходы - их придется создавать повторно. Чтобы скопировать всю ветвь программы, предусмотрена специальная операция (см. ниже).

Если в процессе набора программы возникла необходимость получить информацию, недоступную из текстового редактора, то следует донабрать (произвольным образом, но со строгим соблюдением синтаксических правил для операторов со связанными переменными) текущий оператор, сохранить набранную часть фрагмента нажатием "Enter", выйти в просмотр необходимых данных, а затем вернуться в прерванное редактирование.

При редактировании программы в блоке программ обычно возникают "пустоты" - неиспользуемые остатки предыдущих версий фрагментов программ. Эти пустоты рекомендуется периодически устранять, выбирая в главном меню пункт "Уплотнение измененных файлов". Кроме исключения неиспользуемых фрагментов, операция уплотнения осуществляет контроль за корректностью общей структуры программ. Если по причине ошибки или машинного сбоя в блоке программ обнаруживается дефект, то при выборе указанного пункта либо пункта "Сохранение копий файлов" (тоже осуществляющего данную проверку) произойдет выход из программы в операционную систему. В этом случае следует воспользоваться сохраненной в резервной директории SLCOPY копией программы. Такую копию следует постоянно обновлять выбором в главном меню пункта "Сохранение копий файлов". Рекомендуемый режим работы - регулярный выбор пунктов "Уплотнение...", "Сохранение..." после любых изменений в программе и информационных блоках.

6.2.4 Сдвиг номеров программных переменных

При программировании на ЛОСе следует придерживаться принципа последовательной нумерации новых переменных - это связано в первую очередь с логикой откатов, при которых сбрасываются значения всех переменных, номера которых больше некоторого, а также с экономией объема глобального стэка интерпретатора ЛОСа. Однако, иногда возникает необходимость при доработке программы вставлять внутри нее операторы, вычисляющие те или иные дополнительные значения. Чтобы ввести для этих значений программные переменные, не нарушая принципа последовательной нумерации, приходится выполнять сдвиг (в данной ситуации - увеличение) номеров всех программных переменных, определяемых после точки вставки оператора, на заданную величину. Это осуществляется следующей последовательностью действий:

а) Войти в режим просмотра подтермов операторов и выделить оператор, после которого (включая его самого) требуется выполнить сдвиг нумерации переменных.

б) Нажать клавишу "п". Тогда в левом верхнем углу экрана возникает курсор текстового редактора. Если нужно увеличить номера переменных, большие или равные n, на величину m, то набирается терм "плюс(хn m)"(буква "х кир.). Если нужно уменьшить номера переменных, большие или равные n, на величину m, то набирается терм "минус(хn m)". В обоих случаях после нажатия "Enter"выполняется указанный сдвиг (он затрагивает не только данный фрагмент, но и все подфрагменты, достижимые из выбранного оператора).

6.2.5 Операции с ветвями программы

Чтобы скопировать всю ветвь некоторого фрагмента программы, из обычного просмотра этого фрагмента нажимается клавиша "Insert это приводит к регистрации ссылки на фрагмент в специальном буфере редактора программ. Затем предпринимается переход к тому фрагменту программы, всю ветвь которого следует заменить на всю ветвь исходного (учтенного в буфере) фрагмента. Эта ветвь НЕ ДОЛЖНА находиться внутри заменяющей ветви либо содержать ее. Как правило, такую ветвь сначала приходится специально создавать - состоящей из единственного фрагмента с единственным оператором "продолжение" - ответвляя в нужном месте переход к ней по "ветвь" либо "иначе". Если заменяемая ветвь относится к программе

другого логического символа, то выход в главное меню для перехода к программе этого символа должен происходить только по нажатии клавиши "PageUp" - иначе ссылка в буфере на заменяющую ветвь будет утеряна. После того, как на экране возник корневой фрагмент заменяемой ветви, последовательно нажимаются клавиши "o" (кир.) - для перехода в режим просмотра подтермов - и "к" (кир.) - собственно для копирования ветви.

Для удаления ветви программы, переход к которой из текущего фрагмента программы определяется выделенным желтым цветом оператором перехода, достаточно нажать клавишу "Ctr-Del". Эту операцию можно применять лишь однократно; для повторного удаления с ее помощью следует сначала выйти из просмотра программы текущего логического символа.

Для удаления всей программы логического символа следует перейти в корневой фрагмент этой программы и нажать "Ctr-F7".

Для удаления сразу нескольких ветвей программы, к которым из текущего фрагмента имеются переходы, можно войти в режим редактирования этого фрагмента и исключить операторы перехода, ссылающиеся на указанные ветви. По завершении редактирования (клавиша "Enter") эти ветви будут полностью удалены.

Если ветвь программы была удалена с помощью "Ctr-Del"либо "Ctr-F7", то ее можно восстановить в качестве корневой ветви всей программы данного логического символа - достаточно в корневом фрагменте программы нажать "Ctr-F6". Такое нажатие должно быть выполнено до выхода из программы текущего логического символа. Впрочем, для удаления части программы, находящейся выше некоторого фрагмента, более удобным оказалось использовать операцию слияния двух последовательных фрагментов (см. ниже).

Если фрагмент программы оказался чрезмерно большим (например, после вставки в него дополнительных операторов), то его можно разрезать на две части. Для этого следует войти в режим просмотра подтермов, выделить тот оператор, который должен стать первым оператором второй части, и нажать клавишу "р"(кир.). К концу первой части фрагмента будут добавлены оператор "ветвь N", ссылающийся на вторую часть, а также оператор "продолжение" (переход ко второй части будет выполнен после "отражения"от этого завершающего оператора).

Если два фрагмента программы выполняются последовательно - в том смысле, что первый из них завершается оператором "ветвь N", ссылающимся на второй, и оператором "продолжение", - то можно выполнить слияние их в один общий фрагмент. Эта операция в точности обратна описанной выше операции разрезания фрагмента. Разумеется, применять ее следует так, чтобы результат слияния умещался целиком на экране. В принципе, появление фрагментов, не помещающихся на экране целиком, допустимо (они иногда возникают при работе компилятора ГЕНОЛОГа). Однако, для удобства работы их лучше разрезать на части. Во всяком случае, это необходимо делать перед редактированием такого фрагмента.

6.2.6 Обнаружение ошибок в программе

Предусмотрена возможность автоматического контроля правильности программы логического символа, включающего обнаружение простейших ошибок - несоответствия числа операндов операторов и операторных выражений их арности, использова-

ние значения переменной, которое еще не было определено, попытка повторно определить уже определенное значение, и т.п. Для выполнения такого контроля следует войти в просмотр корневого фрагмента программы и нажать клавишу "п". При наличии ошибок будут выданы соответствующие сообщения (при этом произойдет переход в режим просмотра подтермов и будет выделена точка ошибки - для последующего ее исправления нужно будет сначала нажатием клавиши "о"(кир.) перейти в обычный просмотр фрагмента). При отсутствии ошибок, по окончании анализа программы, произойдет перерисовка корневого фрагмента.

Если контролируется программа нового оператора либо операторного выражения, причем после первого оператора ее корневого фрагмента, уточняющего тип обращения, не идет "иначе", то процедура контроля автоматически вводит программу справочника "арность", указывающую на число операндов этого оператора либо операторного выражения. Для перечисляющих операторов вводится также программа справочника "комментпосылок". При определении арности здесь используется оператор "метка(икс(N))", размещаемый в начале контролируемой программы.

Возможна серийная проверка программ логических символов, начиная с текущего логического символа. Для запуска этой проверки нажимается клавиша "П". Остановка серийной проверки осуществляется нажатием любой клавиши. Для просмотра всех программ системы следует начинать с логического символа "метка", имеющего номер 1. При просмотре перерисовки фрагмента программы на экране не происходит; меняются только заголовки просматриваемых программ в верхней части экрана.

При контроле программы одного логического символа (клавиша "п") либо при серийном контроле (клавиша "П") возможно выполнение дополнительных действий, определяемых программой оператора "смпрог(a b c d e)". К этому оператору происходят обращения для каждого текущего просматриваемого вхождения в операторы фрагментов программ. При этом значением переменной a становится логический символ, к которому относится фрагмент программы; значением переменной b - набор $(F_1, \ldots, F_n)$, определяющий путь от корня программы символа a к текущему просматриваемому фрагменту программы. Каждое F_i есть тройка (F_{i1}, F_{i2}, F_{i3}), у которой (F_{i1}, F_{i2}) - ссылка на фрагмент программы;F_{i3} - символьный номер перехода от этого фрагмента к подфрагменту, соответствующему тройке F_{i-1}. Последний элемент набора b определяет переход от корня программы символа a; первый элемент - переход к текущему фрагменту. Значением переменной c служит набор троек $(G_1, \ldots, G_{n+1})$; $G_i = (G_{i1}, G_{i2}, G_{i3})$. Эти тройки описывают фрагменты программы , расположенные на пути, определяемом набором b; G_{n+1} соответствует корню программы символа a, а G_1 - текущему фрагменту. G_{i1} есть сам фрагмент программы, представленный набором термов в зоне задач; G_{i2} - набор дополнительных логических условий на программные переменные, встречающиеся в операторах фрагмента G_{i1}. G_{i3} - набор информационных элементов, характеризующих фрагмент G_{i1}. Для указания корневого фрагмента здесь используется элемент "корень"; для указания вхождения v в фрагмент оператора перехода, ссылающегося на подфрагмент текущего пути - элемент (позиция v). Входной переменной d оператора "смпрог(...)"присваивается вхождение текущего оператора в список операторов текущего фрагмента; переменной e - вхождение текущего логического символа в текущем операторе.

Если оператор "смпрог"оказывается истинным в некоторой точке просмотра, то просмотр обрывается, и многоцветной указкой выделяется текущее вхождение в текущий оператор текущего фрагмента. Для продолжения просмотра после этого (кроме случаев сообщения о найденной ошибке) следует нажать клавишу "ш". Следует

учитывать, что при выходе из режима многоцветной указки цикл просмотра сбрасывается, и тогда нужно запускать его с текущего логического символа заново.

Оператор "смпрог"можно использовать для поиска точек в программе, удовлетворяющих заданным условиям. Эти условия, с помощью редактора программ, записываются в начале программы данного оператора (предварительно следует удалить тождественно ложный оператор, обычно вставляемый в начало программы "смпрог"для предотвращения ее срабатываний в режиме общей проверки; обычно таким оператором является "равно(0 1)"). В найденной точке можно изменить программу вручную и возобновить поиск, начиная с текущего логического символа. Возможна реализация цикла автоматического изменения всей программы системы - для этого в операторе "смпрог"нужно реализовать процедуру модификации фрагмента или целой ветви программы, и заменить старый фрагмент (ветвь) на новый, используя специальные операторы для работы с фрагментами программ, описанные в разделе, посвященном программной реализации редактора программ.

6.2.7 Сопровождение справочной информацией избранных точек в программе

Чтобы сопровождать избранные точки программ ЛОСа необходимыми пояснениями, а также чтобы быстро можно было находить нужную процедуру и находить в этой процедуре конкретную точку, создано специальное оглавление, в котором перечислены основные процедуры системы, реализованные на ЛОСе. Об этом оглавлении уже говорилось вкратце выше (при описании способов входа в просмотр программ); вход в него - через пункт "Оглавление программ" главного меню. Концевые пункты данного оглавления суть комментарии к избранным точкам программы. Для входа в просмотр такой точки достаточно нажать в концевом пункте клавишу "курсор вправо". После этого возникает текст фрагмента, в котором синим цветом выделен оператор "прием(i)" - он отмечает нужно место в программе. Здесь имеет место режим просмотра подтермов операторов. Далее можно произвольным образом перемещаться по фрагментам программы (не выходя в главное меню) и менять режим просмотра; если в произвольной точке восстановить режим просмотра подтермов и нажать "End", то будет восстановлен исходный концевой пункт оглавления программ.

Если при просмотре фрагментов программы после перехода к ним из оглавления программ (в обычном режиме просмотра) нужно вернуться к тому фрагменту, с которого был начат этот просмотр, то нажимается F8. Произойдет возвращение к просмотру исходного оператора "прием(i)", причем восстановится режим просмотра подтермов.

Если нужно связать комментарий с какой-либо точкой программы, то следует войти в режим просмотра подтермов, выделить тот оператор A, к которому относится этот комментарий, и нажать клавишу "PageDown". После этого на экране возникнет некоторая страница оглавления программ ЛОСа. Перемещаясь по этому оглавлению, а при необходимости - создавая новые его подразделы, следует перейти в то меню, которое связано с описанием рассматриваемой части программы. Далее нужно ввести новый концевой пункт этого меню (клавиша "к"либо "К"), и текстом этого пункта сделать требуемый комментарий. По окончании ввода текста (нажатие клавиши "Enter") перед оператором A будет вставлен фиктивный оператор "прием(N)", ссылающийся на введенный концевой пункт оглавления программ.

Если вход в программу происходил не из оглавления программ, то для получения информации оглавления программ, относящейся к выбранной ее точке, следует выделить в этой точке многоцветной указкой какой-либо оператор и нажать "End". Тогда будет найден первый предшествующий выделенной позиции оператор "прием(N)", и осуществится переход к его концевому пункту оглавления программ.

Чтобы удалить концевой пункт оглавления программ и одновременно удалить ссылку "прием(i)"в программе, соответствующую этому пункту, достаточно нажать "Ctr-Del"при выделенном концевом пункте оглавления программ.

6.2.8 Дополнительные возможности интерфейса редактора программ

Находясь в режиме просмотра фрагмента программы, можно выполнить ряд вспомогательных операций. Прежде всего, именно из этого режима осуществляется вход в создание и редактирование шаблонов диалоговых блоков. При нажатии клавиши "д" возникает оглавление диалоговых блоков, в котором можно либо выбрать для просмотра и редактирования ранее созданный шаблон блока диалога ("курсор вправо" на выбранном пункте; для возвращения в оглавление из редактирования шаблона - "Esc"), либо создать новый шаблон. Последнее осуществляется созданием нового концевого пункта и входом в него - так же, как и для ранее созданного пункта. Интерфейс редактирования шаблона блока диалога был описан выше. Для удаления шаблона достаточно удалить (Ctr-Del) соответствующий концевой пункт оглавления.

Кроме оглавления шаблонов блоков диалога, из режима просмотра фрагмента программы можно перейти в ряд других справочных оглавлений, которые бывает нужно использовать при редактировании программ. Подробнее об этих оглавлениях будет сказано в последующих разделах книги.

Глава 7

Отладчик ЛОСа

7.1 Семантическая трассировка решения задачи

Семантическая трассировка представляет собой пошаговый просмотр процесса решения задачи с отображением на экране сообщений о применении приемов, сопровождаемых пояснениями. Такая трассировка позволяет понять способ решения задачи "в целом"и локализовать моменты, требующие специальной доработки.

Способы запуска решения задачи уже были описаны выше в разделе "Общая схема функционирования решателя". Для запуска решения без трассировки нажимается клавиша "о"; для запуска решения с семантической трассировкой нажимается "р", причем предварительно уточняется режим трассировки. Это делается с помощью диалогового блока, активизируемого нажатием клавиши "т". Выбор левой кнопкой мыши пунктов этого блока включает либо выключает соответствующие установки на трассировку. Для обычного просмотра рекомендуется режим с выключенным пунктом "ручной выбор входа в подпроцесс"; включение этого пункта происходит в тех случаях, когда при решении задачи встречается обращение к чрезвычайно трудоемкой вспомогательной задаче, не выводимое на экран. Такое включение не обязательно выполнять с самого начала трассировки: обращение к диалоговому блоку выбора режима трассировки возможно на каждом шаге. Другие пункты этого диалогового блока таковы:

1) "Пропуск обращений к проверочным операторам" - обычно выключен. Его включение приведет к тому, что перестанут появляться сообщения об обращениях к процедурам проверки различных вспомогательных утверждений.

2) "Пропуск шагов удаления посылок и условий" - обычно включен. Его выключение приведет к тому, что будут явно указываться шаги удаления ненужных условий и посылок задачи.

3) "Пропуск изменений сопровождающих условий" - обычно включен. Его выключение приведет к тому, что будут отображаться изменения условий на область допустимых значений, сопровождающие основные шаги решения.

4) "Пропуск входа в любые подпроцессы" - обычно выключен. Сообщения об обращениях к наиболее существенным для решения задачи вспомогательным задачам и процедурам при его выключении автоматически выводятся на экран (хотя такие обращения означают не срабатывание приема, а только начало попытки его применения). Если этот пункт включен, то данные сообщения пропадут, а останутся лишь сообщения о фактически сработавших приемах.

5) "Просмотр шагов изменения комментариев" - обычно выключен. Изменения технических пометок - так называемых комментариев образуют достаточно интенсивный поток, вывод которого на экран существенно замедлит трассировку.

6) "Пропуск регистрации условий на о.д.з." - обычно включен. При его выключении более подробно будет отображен процесс первоначального ввода условий на область допустимых значений, сопровождающих условия и посылки задачи.

Чтобы перейти из режима семантической трассировки в просмотр текущей ситуации с помощью отладчика ЛОСа, достаточно нажать клавишу "ф". Тогда на экране появится текст текущего исполняемого фрагмента программы, вплоть до текущего оператора (пока не реализованного), который будет выделен малиновым цветом. Дальнейшие действия - согласно инструкции по отладчику ЛОСа (см. ниже). Для возвращения в просмотр текущего кадра семантической трассировки и продолжения ее достаточно нажать клавишу "р"либо клавишу "з".

7.2 Установка режимов технической трассировки перед запуском решения

Если нужно отладить прием, реализованный на ГЕНОЛО́Ге, то сначала следует создать либо найти в задачнике такую задачу, где он должен сработать. После этого можно создать установку на прерывание при попытке применения данного приема и запустить решение задачи до обнаружения такой попытки. Для этого (из просмотра задачи в задачнике) нажимается клавиша "г"или выбирается пункт "Генолог" меню выбора режима трассировки, расположенного в верхней части экрана. Тогда возникает некоторый пункт оглавления базы приемов ГЕНОЛОГа. В этом оглавлении находится нужный пункт, и внутри группы приемов этого пункта находится нужный прием (для этого служат клавиши "курсор вверх" - "курсор вниз"). Далее нажимается клавиша "Ctr-Enter", которая одновременно создает установку на прерывание и запускает решение задачи.

Если в процессе решения произойдет выход на начальный отрезок программы выбранного приема (именно, на используемый для указания этого начала фиктивный оператор "контрольприема(A_1 A_2 A_3)"; здесь A_1, A_2, A_3 - логические символы и термы, образующие стандартную техническую ссылку на прием), то включится отладчик ЛОСа, и на экране будет отображен фрагмент программы с выделенным малиновым цветом оператором, который должен будет выполняться непосредственно на следующем шаге (этот оператор следует за оператором "контрольприема($A_1A_2A_3$)"). При этом устанавливается пошаговый режим трассировки (см.ниже).

Если выбранный прием применяется при сканировании задачи либо при работе пакетного нормализатора (см. описание ГЕНОЛОГа), то можно установить прерывание при фактическом применении этого приема. Для этого вместо клавиши "Ctr-Enter"нажимается клавиша "Enter".

Если нужно проверить, применяется ли при решении задачи некоторый оператор либо операторное выражение, реализованные с помощью программы на ЛОСе, и проследить по шагам ход выполнения этой программы, то перед запуском решения задачи нажимается клавиша "Л". Тогда в левой нижней части экрана возникает надпись "Логический символ:" и появляется курсор текстового редактора. Выполняется набор необходимого логического символа и нажимается "Enter". Если было

набрано слово, не являющееся названием логического символа, то оно пропадает с экрана и курсор возвращается в исходное положение - для повторного набора. Иначе - одновременно создается установка на прерывание при обращении к программе указанного логического символа и запускается процесс решения задачи. По достижении первого оператора корневого фрагмента программы выбранного символа включается отладчик ЛОСа; сам этот оператор прорисован малиновым цветом и еще не выполнен. Для дальнейших действий автоматически устанавливается пошаговый режим трассировки.

Если нужно проследить моменты выхода к точкам программы, для которых созданы соответствующие концевые пункты оглавления программ, то перед запуском решения задачи нажимается клавиша "л". Тогда возникает некоторый пункт оглавления программ. Если выбрать нужный концевой пункт этого оглавления и нажать на нем клавишу "курсор вправо", то одновременно инициируется прерывание при выходе на соответствующую контрольную точку программы (напомним, что эта точка представляет собой фиктивный оператор "прием(N)"; N - номер контрольной точки внутри программы данного логического символа). Как и в предыдущих случаях, по достижении контрольной точки включается отладчик ЛОСа и устанавливается пошаговый режим трассировки.

При отладке бывает полезно применение счетчика шагов работы интерпретатора ЛОСа. Оно дает возможность при повторных запусках решения задачи идентифицировать (например, методом деления отрезка пополам) момент ошибочного действия. Это средство применяется, впрочем, достаточно редко, так как подавляющее большинство ошибок обнаруживаются по недопустимым типам входных данных операторов ЛОСа и немедленно выводятся на экран. Однако, в некоторых случаях появление заведомо ошибочных элементов структуры данных не выявляется средствами общего контроля, и анализ текущей информации не позволяет объяснить их происхождение. В этих случаях и используется счетчик шагов. При трассировке на уровне отладчика ЛОСа номер текущего шага работы интерпретатора выводится на экран в правом верхнем углу. Этот номер позволяет примерно определить диапазон, в котором следует искать момент появления ошибки. Если при повторном запуске решения задачи нажать клавишу "Ш"и ввести текстовым редактором в появившемся окне диалога (после надписи "Число операторов:") номер шага, соответствующего началу диапазона поиска ошибки, то по нажатии "Enter"будет одновременно установлено прерывание по достижении заданного шага и начато решение задачи. По достижении нужного шага включается отладчик ЛОСа и устанавливается пошаговый режим.

Кроме установок на автоматическое прерывание процесса решения, можно в любой момент прервать этот процесс вручную. Для этого достаточно нажать клавишу "Break". В результате появится текст текущего исполняемого фрагмента программы, в котором текущий (еще не выполненный) оператор выделен малиновым цветом. Для работы в отладчике ЛОСа устанавливается пошаговый режим. Заметим, что нажатие клавиши "Break"не позволяет входить в просмотр функционирования программ отладчика.

Наконец, самый простой (но сравнительно трудоемкий) способ вызвать прерывание работы программы и обращение к отладчику ЛОСа в нужной точке программы - вставить в этой точке оператор "трассировка(стоп 0)". Этот оператор вызывает немедленный вход в отладчик ЛОСа с пошаговым режимом, не выполняя никаких других действий (он всегда истинен). По завершении анализа текущего шага отладчиком и продолжении трассировки будет выполняться следующий за ним оператор.

Единственное необходимое условие его корректной работы - наличие после него в программе хоть какого-либо другого оператора. Заметим, что помещение данного оператора внутри процедур интерфейса и даже процедур самого отладчика приводит к тому же эффекту, так что он делает возможной отладку с помощью отладчика ЛОСа тех участков программы отладчика, в которых этот оператор не будет немедленно повторно выполняться при попытке обращения к отладчику.

7.3 Просмотр информации о моменте прерывания

Собственно отладчик ЛОСа представляет собой программу логического символа "прерывание", которая активизируется на тех шагах работы системы, для которых либо оказывается истинным заранее установленное условие прерывания, либо нажимается клавиша "Break", либо обнаруживается попытка реализации оператора ЛОСа для недопустимых входных данных. Эта программа позволяет просмотреть информацию о текущем состоянии системы и изменить режим трассировки, в том числе ввести новые установки на прерывание. При выходе из нее продолжается прерванное решение задачи. На момент входа в программу "прерывание"счетчик шагов интерпретатора ЛОСа отключается, так что работа с отладчиком никак не сказывается на соответствии номеров шагов действиям решателя. В процессе работы с отладчиком ЛОСа можно восстановить исходное изображение, возникшее на момент обращения к нему при прерывании, нажав клавишу "р"(кир.).

7.3.1 Просмотр программы

При входе в отладчик ЛОСа на экране прорисовывается некоторый фрагмент программы, в котором малиновым цветом выделен текущий (пока не выполненный) оператор. Этот фрагмент, вообще говоря, не является корневым; можно просмотреть все его надфрагменты в дереве программы текущего логического символа, используя клавиши "Home" (шаг по направлению к корню программы) и "End" (шаг от корня программы к фрагменту с текущим оператором). С указанными фрагментами связан общий набор значений программных переменных текущей ситуации, возникшей при выполнении текущей программы; эти значения сохраняются в некотором кадре глобального стэка интерпретатора ЛОСа. Заметим, что из режима трассировки возможен просмотр только линейной цепочки фрагментов, от корня программы до фрагмента с текущим оператором; выход на боковые ветви программы не предусмотрен.

При просмотре фрагментов программы выдается только начальный отрезок фрагмента до оператора, выделенного малиновым цветом (это либо текущий оператор, либо оператор перехода к подфрагменту). Чтобы просмотреть все операторы фрагмента, нажимается клавиша "ф". Чтобы убрать с экрана лишние (идущие после выделенного малиновым цветом) операторы, нажимается клавиша "о". В левой верхней части экрана размещен текст "Программа A", где A - заголовок программы (выделяется синим цветом). В правой верхней части экрана размещен текст "шаг № N", где N - номер текущего шага работы интерпретатора ЛОСа. Этот шаг не является абсолютным; он отсчитывается только с момента запуска решения извлеченной из задачника задачи, что делает его неизменным при повторных запусках и позволяет использовать для локализации событий, происходящих при решении.

Под текстом фрагмента программы проведена горизонтальная черта, отделяющая область экрана, в которой можно получать дополнительную информацию о текущем шаге.

Кадр глобального стэка интерпретатора ЛОСа сохраняет информацию о текущих значениях программных переменных, возникающих при реализации программы некоторого оператора либо операторного выражения, либо при сканировании некоторой задачи. При обращении от программы к подпрограмме создается новый стэковый кадр, для сохранения значений программных переменных этой подпрограммы. Таким образом, возникает цепочка стэковых кадров, в которых сохраняются значения программных переменных как для текущей программы, так и для всех ее надпрограмм. Самая "верхняя"программа в этой цепочке - сканирование исходной фиктивной задачи на исследование; от нее имело место обращение к сканированию задачи, извлеченной из задачника; далее - к некоторой вспомогательной задаче либо оператору, и т.д. Возможен просмотр всей цепочки "надпрограмм" текущей реализуемой программы с помощью отладчкиа ЛОСа. Для этого служат клавиши "PageUp" и "PageDown". Первая из них переводит от просмотра программы к ее надпрограмме; вторая - от программы к подпрограмме (все это - вдоль линейной цепочки обращений, сложившейся на текущий момент). При переходах прорисовываются отрезки соответствующих фрагментов программы вплоть до того оператора, от которого имело место обращение к подпрограмме. На каждом уровне цепочки обращений возможен просмотр значений программных переменных данного уровня. Так как значения этих переменных при выполнении программы на ЛОСе часто сохраняются неизменными (значение однажды определенной переменной обычно меняется только при откатах), то становится возможен анализ значительной части "предыстории" текущей ситуации.

Использование пар клавиш "Page Up - Page Down"и "Home - End"делает просмотр текущей ситуации двумерным: можно варьировать фрагмент программы по цепочке переходов внутри программы одного логического символа, а также варьировать саму рассматриваемую программу по цепочке обращений от одной программы к другой.

При просмотре программы отладчиком, как и при просмотре ее редактором программ, возможен вход в режим просмотра подтермов операторов (многоцветная указка). Этот режим упрощает чтение и понимание сложных логических операторов, а также используется при установке прерываний (см. ниже). Для входа в режим просмотра подтермов операторов следует нажать клавишу "курсор вниз" - первый из операторов текущего фрагмента выделяется синим цветом. Далее клавиши "курсор вправо" - "курсор влево" позволяют выбрать нужный оператор фрагмента; клавиша "курсор вниз" - войти в просмотр операндов выделенной операции; клавиши "курсор вправо" - "курсор влево" - выбрать нужный операнд. Для возвращения от операнда к внешней операции и выхода из режима просмотра подтермов нажимается клавиша "курсор вверх".

Для получения информации о логическом символе, выделенном при просмотре подтермов оператора, нажимается клавиша F3.

В режиме просмотра фрагмента программы (без выделения подтермов операторов) возможен обычный доступ к справочной информации о логических символах: нажатие клавиши F3 приводит к выдаче информации о логическом символе - заголовке текущей программы; нажатие клавиши F2 - к появлению курсора текстового редактора, с помощью которого набирается название логического символа, о котором

требуется получить информацию.

7.3.2 Просмотр цепи задач

Для входа в просмотр цепи задач (текущей задачи и всех ее надзадач) нажимается клавиша "з". При просмотре применяются клавиши "курсор вверх" - "курсор вниз"; "PageUp" - "PageDown"; "Ctr-PageUp" (начало цепи задач); "Ctr - PageDown" (конец цепи задач, т.е. текущая задача); "Ctr - курсор вверх" (к началу предыдущей задачи); "Ctr - курсор вниз" (к началу следующей задачи). Заметим, что хотя внешне просмотр цепи задач из отладчика ЛОСа ничем не отличается от просмотра ее при трассировке по шагам решения задачи, нажатие клавиши "Enter" в первом случае не приводит к продолжению процесса решения задачи.

Для возвращения в основной интерфейс отладчика ЛОСа из интерфейса просмотра цепи задач нажимается клавиша "ф".

7.3.3 Просмотр значений программных переменных

Для просмотра различной информации о текущей ситуации используется область экрана, расположенная под горизонтальной чертой, отделяющей текст фрагмента программы. Если эта область недостаточна, то можно вообще убрать с экрана текст фрагмента программы, нажав клавишу "Delete" (этот текст восстанавливается нажатием клавиш "о" - до текущего оператора, либо "ф" - полный текст фрагмента). Если нужно убрать текст программы, сохранив информацию, размещенную в нижней части экрана, то вместо "Delete" нажимается F5. В нижней части экрана данные выдаются в режиме автоматической прокрутки вверх (обратная прокрутка невозможна, и для повторного просмотра удаленных с экрана объектов их следует востребовать заново). Возможна ручная прокрутка вверх содержимого нижней части экрана (при сохраняющемся в верхней части тексте фрагмента программы) с помощью клавиши "курсор вниз".

Для просмотра значения программной переменной "хi"следует нажать клавишу "х" (кириллица) - появятся буква "х" и курсор текстового редактора, и далее набрать номер i в обычной десятичной записи.

Если значением программной переменной "хi"является терм, то он будет выдан в стандартной либо в скобочной записи в зависимости от ранее установленного режима просмотра термов. Клавиша "с" переводит в скобочный режим, клавиша "м" (кир.) - в режим стандартной математической записи.

Если значением переменной "хi" служит набор, не являющийся термом, то будут последовательно выданы все его разряды. Здесь логические символы и символы переменных указываются явно (последние - в виде, согласованном либо со скобочной записью, т.е как буква "х"с номером, либо в виде, согласованном со стандартной математической записью - как латинская буква, быть может, с индексом). Элемент набора, представляющий собой терм, обозначается посредством записи "тj", где j - номер, закрепляемый за данным термом на период просмотра элементов текущей структуры данных (эти номера сбрасываются при смене текущего просматриваемого фрагмента клавишами PageUp,PageDown, Home,End). Элемент набора, представляющий собой набор, отличный от терма, обозначается посредством записи "нj", где j

- номер, закрепляемый за указанным набором на период просмотра структуры данных. Элемент набора, представляющий собой вхождение в некоторый набор либо в терм, обозначается посредством "вj", где j - номер, закрепляемый за данным вхождением на период просмотра. Нумерация объектов всех указанных типов - общая.

Если значением переменной "хi"служит вхождение в терм либо в набор, то будет прорисован, соответственно, этот терм либо набор, в котором подтерм либо разряд, находящийся по рассматриваемому вхождению, выделен синим цветом.

Объекты "нj", "тj", "вj", введенные при просмотре набора, сами могут быть отображены на экране в одном из указанных выше видов. Для этого следует лишь нажать, соответственно, клавишу "н", "т" либо "в" и после этого набрать номер j объекта (набор номера завершается нажатием клавиши "Enter").

Для большей наглядности можно изображать набор в виде последовательности строк, каждая из которых в указанном выше формате представляет один разряд набора. При этом предусмотрена выдача лишь тех элементов набора, которые не являются логическими символами либо переменными (символьные элементы доступны при обычном просмотре набора, и здесь они пропускаются). Если рассматриваемый набор является значением переменной "хi", то нажимается клавиша "Х" (кир.) и вводится номер i; если же набор обозначен как "нi", то нажимается клавиша "Н" (кир.) и вводится номер i. Возможно отображение в указанном виде лишь тех элементов набора, которые начинаются с заданного логического символа S. Для этого после набора номера i в окне текстового редактора помещается название символа S (отделенное от номера пробелом).

Если требуется одновременно выдать на экран в скобочной записи все элементы набора термов "нi", то нажимается клавиша "а"(кир.), после чего вводится текстовым редактором номер i.

По-видимому, для просмотра сложных структур данных наиболее удобным является использование "сквозного режима". В этом режиме используется все пространство экрана (текст фрагмента программы временно удаляется); каждый разряд набора прорисовывается, после указания его номера, на отдельной строке либо группе строк, причем возможна обычная прокрутка (клавиши "курсор вверх-вниз" и "PageUp - PageDown"). Синим цветом выделяется номер текущего разряда набора (смена этого номера - клавишами "курсор вверх - вниз"). Если текущий разряд сам является набором, то для его просмотра нажимается клавиша "курсор вправо", после чего он прорисовывается на экране в указанном поразрядном виде. Таким образом можно дойти до атомарных объектов, не являющихся наборами. Возвращение от элемента к внешнему набору, а также выход из указанного режима обеспечиваются нажатиями клавиши "курсор влево". Номера атомарных (не являющихся наборами) разрядов завершаются круглой скобкой; номера разрядов, являющихся наборами - точкой. При отображении разряда, являющегося набором, используются обычные сокращения: "т" означает терм, "н" - набор, "в" - вхождение. Эти сокращения, однако, не сопровождаются номерами, излишними при возможности выбора объекта для просмотра с помощью курсора. Чтобы просмотреть в сквозном режиме набор, являющийся значением переменной "хi", нажимается клавиша "К"(кир.) и вводится номер i. Если просматривается набор "нi", то нажимается "к"(кир.) и вводится i.

Если задача сопровождается чертежом, то возможна его прорисовка в отладчике ЛОСа. Нажатие клавиши "ч"приводит к его прорисовке в верхней части экрана; клавиши "Ч к прорисовке чертежа под ранее выведенной в нижней части экрана

информацией. После вывода чертежа возможен обычный просмотр элементов текущей структуры данных, размещаемых под чертежом.

Если используются как стандартная математическая, так и скобочная записи термов, то для установления связи между обозачениями переменных в этих способах записи используется переходная таблица (она перечисляет пары "обозначение переменной задачи в стандартной записи - обозначение этой переменной в скобочной записи"). Для вызова на экран переходной таблицы нажимается клавиша "п".

7.3.4 Сообщение об ошибке в программе

Если интерпретатор ЛОСа обнаруживает попытку реализации некоторого оператора с недопустимыми для него типами входных данных, то он инициирует вход в отладчик ЛОСа и выдачу сообщения об ошибке: в верхней левой части пустого экрана возникает сообщение "Некорректное обращение к оператору". При входе в просмотр текущего фрагмента программы (нажатие клавиши "ф") будет прорисована часть этого фрагмента, предшествующая текущему оператору (выделенному малиновым цветом). Анализ окрестности этого оператора обычно позволяет выявить описку в программе. Возможны также обращения к отладчику ЛОСа при обнаружении интерпретатором переполнения различных ресурсов. В этих случаях выдаются следующие сообщения об ошибке:

1) "Переполнение глобального стэка" - означает, что имеется чрезмерно длинная цепочка обращений от программы к подпрограмме, так что суммарный объем кадров глобального стэка, описывающих значения программных переменных всех этих программ, превысил предельно допустимую величину. Часто это происходит при "зацикливании" в рекурсивных обращениях.

2) "Переполнение буфера текстов" - означает, что предпринята попытка создать настолько большой терм или набор, что он превысил предельно допустимую величину.

3) "Переполнение зоны задач" - означает, что вся зона задач заполнена объектами, введенными решателем, и места для дальнейшей его работы не хватает. Так как величина зоны задач достаточно велика, и обычный (нормальный) режим работы системы - при практически пустой зоне задач, появление этого сообщения обычно связано с каким-либо зацикливанием программы. Появляется это сообщение чрезвычайно редко.

4) "Исчерпание переменных либо переполнение при арифметическом действии" - обычно означает, что ведено слишком много символов переменных и запрос на ввод нового такого символа не может быть удовлетворен.

5) "Превышение допустимого сброса" - означает попытку применения оператора сброса, выводящую за рамки действия текущей программы (число указанных для сброса перечисляющих операторов в рамках программы больше фактически имеющегося).

Кроме перечисленных сообщений, может появиться сообщение "Нет программы логического символа S" - если предпринята попытка обратиться к выполнению отсутствующей программы этого символа.

7.3.5 Выход в базу приемов, реализованных на ГЕНОЛОГе

Если выход в отладчик ЛОСа произошел при семантической трассировке после срабатывания некоторого приема, то нажатие клавиши "б"приводит к просмотру описания примененного приема на ГЕНОЛОГе - если был применен прием, запрограммированный на ГЕНОЛОГе, либо к просмотру концевого пункта оглавления программ ЛОСа, - если прием запрограммирован непосредственно на ЛОСе, и данному концевому пункту соответствует последняя пройденная перед срабатыванием контрольная точка "прием(N)". При технической трассировке, если просматривается некоторый фрагмент программы (текущий либо некоторый его надфрагмент, достижимый из текущего с помощью клавиши "PageUp"), то нажатие клавиши "б" также приводит к просмотру описания соответствующего этому фрагменту приема ГЕНОЛОГа либо концевого пункта оглавления программ ЛОСа - однако, лишь при условии, что текущий оператор рассматриваемого фрагмента расположен после оператора "контрольприема(...)", указывающего на прием ГЕНОЛОГа, либо оператора "прием(N)", указывающего на концевой пункт оглавления ЛОСа.

Выйти в оглавление ГЕНОЛОГа из отладчика ЛОСа можно и безотносительно к срабатыванию текущего приема. Для это следует нажать клавишу "г", после чего появляется некоторый пункт оглавления базы приемов. По этом оглавлению можно перемещаться без каких-либо ограничений, входя в просмотр любых приемов ГЕНОЛОГа - так же, как и при просмотре этих приемов из главного меню. Заблокированы лишь те операции просмотра приемов, которые приводят к изменению приемов.

Если в оглавлении базы приемов был выбран некоторый концевой пункт и произошел вход в просмотр приема, закрепленного за данным пунктом (смена таких приемов, если их несколько, обеспечивается клавишами "курсор вверх - курсор вниз"), то возможна установка прерывания при попытке применить данный прием. Для ее ввода достаточно нажать клавишу "Ctr-Enter". Автоматически будет возобновлен процесс решения задачи, и выход в отладчик произойдет при обращении к оператору "контрольприема(...)", с которого начинается программа данного приема. Здесь устанавливается режим пошаговой трассировки и прорисовывается текущий фрагмент программы с выделенным малиновым цветом оператором, непосредственно следующим за оператором "контрольприема(...)". Для выявления собственно момента применения данного приема далее можно установить прерывание при выполнении завершающей части его программы (выбрав один из заключительных ее операторов, например, оператор "пересмотр").

7.3.6 Техническая трассировка

При отладке программы обычно используются такие режимы ее выполнения (называемые технической трассировкой), при которых процесс дробится на отдельные небольшие фрагменты, после каждого из которых осуществляется выход в отладчик ЛОСа. Принцип дробления можно либо задать один раз на весь период трассировки, переходя после этого к очередному шагу нажатием клавиши "Enter", либо каждый раз устанавливать очередное прерывание процесса специально.

Перечислим используемые в отладчике ЛОСа режимы технической трассировки и способы установки прерываний процесса. Сразу заметим, что для отмены любых установок на прерывания служит клавиша "0". После нажатия на нее, если не создать новых установок на прерывания и нажать "Enter", то решение задачи будет

продолжено без выдачи каких-либо пояснений на экране вплоть до завершения (ответа либо отказа).

1) Трассировка с самым мелким дроблением процесса - так называемая пошаговая трассировка. Для установки этого режима из отладчика ЛОСа достаточно нажать клавишу "1"(кроме того, пошаговый режим устанавливается автоматически после целого ряда прерываний, выводящих в отладчик). После этого каждое нажатие клавиши "Enter" будет вызывать выполнение единственного шага - реализацию оператора (корневого либо расположенного внутри некоторого составного оператора) либо вычисление невырожденного операторного выражения. При пошаговом режиме трассировки можно входить в просмотр шагов выполнения составных операторов (дизъюнкций, конъюнкций, кванторов и т.п.).

2) Следующий режим - пооператорная трассировка. В этом режиме за один шаг выполняется один корневой оператор программы - как простой, так и составной. Для установки режима следует выбрать, используя клавиши "PageUp - PageDown", ту программу, в которой предполагается выполнять трассировку, и нажать клавишу "2". Будут отслеживаться только корневые операторы, которые выполняются в "кадре" этой программы, без выхода в выполнение подоператоров и вспомогательных задач. По окончании выполнения программы трассировка должна быть переустановлена - иначе следующее прерывание произойдет в каком-то достаточно случайном месте. Эта трассировка при отладке является наиболее употребительной.

3) Иногда бывает нужно отслеживать моменты обращения к программе заданного логического символа A. Чтобы установить прерывание при обращении к этой программе, следует сначала нажать клавишу "л" - в результате появится курсор текстового редактора, - а затем набрать текстовым редактором название символа A. После этого, при нажатии "Enter", установка оказывается введенной, хотя продолжение процесса реализации программы еще не включается. При запуске программы (снова нажатием "Enter") отладчик обеспечит выход в просмотр первого оператора начального фрагмента программы символа A, как только произойдет обращение к этой программе. Установка на данное прерывание сохраняется до установки другого прерывания либо отмены прерываний, так что последовательные нажатия "Enter" позволят проследить цепочку указанных обращений.

4) При просмотре отладчиком некоторого фрагмента программы (текущего фрагмента либо его надфрагмента, достигнутого при помощи "Home" либо "PageUp") возможен ввод установки на прерывание при переходе к его подфрагменту. Для этого следует войти в режим просмотра операторов фрагмента (нажатием клавиши "курсор вниз"; выбрать требуемый оператор перехода к подфрагменту (клавиши "курсор вправо - курсор влево") и нажать клавишу "Ctr-Enter". Нажатие этой клавиши автоматически запускает процесс продолжения выполнения программы; при выходе на указанный подфрагмент произойдет прерывание. Заметим, что прерывание будет иметь место, даже если выход на подфрагмент произошел не из того "кадра" реализации программы, где была введена установка на прерывание, а из какого-либо совсем другого "кадра" реализации той же программы. Если требуется, чтобы прерывание произошло лишь при условии, что обращение к подфрагменту имело место в текущем "кадре"реализации программы, то после выбора оператора перехода нажимается "Enter". В этом случае завершение выполнения программы без выхода на выбранный ее подфрагмент приведет к прерыванию по ее завершении.

5) При просмотре отладчиком некоторого фрагмента программы (текущего фраг-

мента либо какого-либо его надфрагмента, достигнутого при помощи клавиш "Home" либо "PageUp") возможен ввод установки на прерывание перед выполнением какого-либо оператора этого фрагмента (не обязательно корневого; возможна установка прерывания перед выполнением подоператора составного оператора). Для этого следует войти в режим просмотра операторов фрагмента ("курсор вниз"); выбрать требуемый оператор (сначала клавишами "курсор вправо - курсор влево" выйти на нужный терм фрагмента, а при необходимости использовать клавишу "курсор вниз" и указанные выше клавиши для выделения подтерма) и нажать "Ctr-Enter". При этом автоматически запускается продолжение действий по программе, и при выходе на указанный оператор (непосредственно перед его реализацией) происходит прерывание. Заметим, что прерывание произойдет, даже если "кадр" реализации программы, в котором было установлено прерывание, отличается от "кадра" реализации этой же программы, в котором имело место обращение к оператору. Чтобы ввести установку на прерывания только в том же самом "кадре" реализации программы, следует нажать "Enter".

Использование установок на прерывания из пунктов 3,4,5 позволяет при отладке свободно перемещаться по программам различных логических символов и анализировать ситуации, возникающие на моменты выхода в различные их точки.

6) Чтобы использовать при отладке счетчик шагов работы интерпретатора ЛОСа, можно установить прерывание по достижении заданного числа шагов. Для этого нажимается клавиша "ш" - появляется курсор текстового редактора, - и набирается номер нужного шага. При наборе номера следует учитывать расположенный в правом верхнем углу экрана (если просматривается фрагмент программы) номер текущего шага. Как уже говорилось выше, при обращениях к отладчику счетчик шагов отключается, так что процесс отладки не изменяет выраженных в числе шагов интерпретатора "координат" событий, происходящих при решении задачи. После набора номера требуемого шага (как обычно, завершаемого нажатием "Enter") установка на прерывание введена; для продолжения цикла выполнения программы еще раз нажимается "Enter".

Если решается задача, извлеченная из задачника, то можо сохранить номер текущего шага и при повторных запусках ее решения выходить в отладчик по достижении шага с данным номером. Для сохранения номера следует нажать клавишу "щ". Чтобы повторно запустить задачу с прерыванием по достижении данного шага, следует нажать клавишу "а" при просмотре условия задачи в задачнике - это нажатие одновременно запускает решение и вводит установку на прерывание.

7) Если произошло прерывание с выходом в отладчик ЛОСа, отличное от обычного шага семантической трассировки, то для возвращения в режим семантической трассировки следует нажать клавишу "пробел", после чего нажать "Enter". Прерывание произойдет при первом же применении приема - в режиме сканирования задачи либо при реализации пакетного оператора ГЕНОЛОГа (только для особо крупных операторов, выделяемых специальной опцией в своем описании). Если вместо клавиши "пробел" нажать клавишу "й", то прерывание произойдет при первом же применении приема на уровне сканирования текущей задачи - выполнение пакетных операторов при этом игнорируется.

8) Если нужно ввести прерывание при выходе из текущего "кадра"реализации программы (выбираемого с помощью "PageUp, PageDown"), то нажимаются клавиши

"4" и "Enter". При ликвидации этого кадра и возвращении во внешнюю программу произойдет выход в отладчик и установится пошаговый режим трассировки.

9) Для ввода прерывания при изменении оператором "замена"в текущем кадре реализации программы значения переменной "хi", если этим значением служит терм, нажимается клавиша "7" - появляются буква "з" и курсор текстового редактора, - и далее набирается символьный (!) номер i. При каждом изменении указанной переменной будет возникать текст "значение переменной хi: A заменяется на B".

10) Если требуется установить прерывание при изменении разрядов набора, обозначенного при просмотре элементов текущей структуры данных посредством "нi", то нажимается клавиша "и" - появляется курсор текстового редактора, - и вводится номер i в обычном формате десятичного числа. Прерывание произойдет непосредственно перед выполнением оператора, изменяющего указанный набор. Если представляющий интерес набор является значением программной переменной "хi", для установки прерывания при изменении его разрядов нажимается клавиша "И" и вводится номер i.

11) Из отладчика можно выйти в оглавление программ и установить прерывание по достижении заданной в этом оглавлении контрольной точки. Для этого сначала нажимается "Ctr-г" - появляется некоторое меню оглавления программ. После выбора в оглавлении нужного концевого пункта, следует нажать на нем клавишу "курсор вправо". Это приведет к установке прерывания по выходу на соответствующую контрольную точку "прием(N)" и возобновлению процесса решения задачи.

12) Прерывание перед выполнением заданного оператора фрагмента программы (см. пункт 5) можно сопровождать списком дополнительных условий, а также обеспечивать автоматическую выдачу на экран при таком прерывании значений заданных программных переменных. Для входа в просмотр и редактирование списка дополнительных условий следует нажать при просмотре фрагмента программы клавишу "у" (кир.). При этом включается редактор списков, отображающий ранее введенные условия. Окно его располагается непосредственно под горизонтальной чертой, завершающей текст фрагмента программы. Список условий разбивается на группы; первым элементом каждой группы служит некоторый ключевой терм (см.ниже), после которого размещаются сопровождающие его информационные элементы. Используются ключевые термы следующих типов:

а) "терм(A)". A есть программное выражение, определяющее объект для сравнения. Следующий элемент списка есть терм, который должен совпадать с этим объектом.

б) "заголовок(хi A)". Значением программной переменной "хi" является логический символ A либо набор или терм, начинающийся с логического символа A.

в) "входит(A хi)". Значением переменной "хi"является терм либо набор, содержащий логический символ либо символ переменной A.

г) "корень". Текущий терм задачи должен совпадать с следующим элементом данного списка установок.

д) "См(A)". A - программная переменная "хi"либо выражение "буква(хi)". Если значением A служит терм либо логический символ, то при прерывании предпринимается выдача этого значения на экран.

В редакторе списков используется следующий интерфейс. Для набор очередного элемента списка формульным редактором служит клавиша "Enter"; для набора элемента текстовым редактором - клавиша "т"(кир.). Для изменения текущего элемента

формульным редактором нажимается "Ctr-ф"; для изменения его текстовым редактором - "Ctr-т". Завершается просмотр и редактирование списка нажатием клавиши "курсор влево". После этого, как обычно вводится установка на прерывание при обращении к некоторому оператору текущего фрагмента программы. Данное прерывание будет происходить всякий раз перед обращением к выбранному оператору, как только окажутся выполнены дополнительные ограничения согласно списку. Для сброса списка дополнительных установок на прерывание нажимается клавиша "Ctr-у"(кир.).

13) В заключение отметим еще одну исключительно важную возможность, предоставляемую отладчиком ЛОСа. При автоматическом решении серии задач из задачника используется так называемый режим "внутреннего перезапуска" - после каждой задачи интерпретатор повторно инициирует выполнение программы логической системы, обнулив все ее регистры и массивы в оперативной памяти. Решение серии задач невозможно отменить, даже используя предоставляемые операционной системой возможности по экстренному выходу из программы - при повторном запуске оно возобновится с той точки, где было прервано. Единственным способом отмены этого режима служит выход в отладчик ЛОСа с помощью клавиши "Break" и нажатие в отладчике клавиши "Ctr-з", которое, собственно, и отменяет серийное решение задач. Далее можно вернуться в главное меню, например, нажатием клавиши "Esc" (возвращение в главное меню без нажатия "Ctr-з" просто вызовет переход к следующей задаче серии).

7.4 Тестирование программы оператора, операторного выражения либо справочника

Для тестирования программы нового оператора, операторного выражения либо справочника лучше всего воспользоваться реальным контекстом, в котором она должна применяться - приемом решения задачи, либо приемом пакетного оператора, либо блоком интерфейса системы, и т.п. Однако, можно обратиться к этому оператору и непосредственно. Для этого служит программа логического символа "пуск", которую можно запускать из просмотра программы в редакторе программ ЛОСа нажатием клавиши "Ctr-Enter". Эта программа имеет единственную входную переменную x1, значением которой служит исходная задача. Войдя в редактирование программы символа "пуск", нужно набрать обращение к тестируемому оператору (программному выражению, справочнику). Сначала программируется построение входных объектов для тестируемой программе, затем размещается само обращение, после которого нужно поместить хотя бы один оператор - чтобы до выхода из программы "пуск" иметь возможность просмотра значений выходных переменных, полученных после обращения. Для удобства создания входных объектов и просмотра результатов обращения, если необходимо, в программу "пуск" можно включить упоминавшиеся выше вспомогательные операторы интерфейса. По завершении редактирования программы "пуск" следует нажать "Ctr-Enter". Тогда инициализируется работа только что набранной программы "пуск", причем сразу же включается отладчик ЛОСа с режимом пошаговой трассировки. Далее с помощью отладчика ЛОСа можно предпринять анализ выполнения тестируемой программы. По завершении программы "пуск" происходит возвращение в главное меню.

Глава 8

Примеры и упражения по программированию на ЛОСе

8.1 Примеры программ приемов, применяемых при сканировании задачи

ЛОС создавался для программирования приемов решения задач, и естественно продемонстрировать прежде всего несколько примеров таких программ. Хотя после появления ГЕНОЛОГа практически все приемы записываются на нем, эти приемы затем компилируются в программы на ЛОСе, которые и работают при решении задач. Поэтому представление о том, как устроена типичная программа приема на ЛОСе, весьма полезно при отладке.

Начнем с приема, выполняющего упрощение алгебраических выражений по формуле $a(\sin x)^2 + a(\cos x)^2 = a$. Первый шаг, который следует сделать при написании программы приема - выбрать логический символ, с рассмотрения которого при сканировании задачи начинается выполнение этой программы. При выборе такого символа следует учитывать, что не каждый символ, встречающийся в записи теоремы, на которой основан прием, обязательно будет встречаться в преобразуемом терме задачи. Прием должен иногда усматривать и "замаскированные" возможности срабатывания, используя те или иные простые преобразования для искусственного представления встретившегося в задаче терма в том виде, какой указан в теореме.

В нашем случае выбор, например, символа "умножение" нежелателен, так как это умножение (на a) при $a = 1$ либо $a = -1$ будет отсутствовать в задаче. С другой стороны, выбор символа "синус" является, по-видимому, вполне приемлемым. При выборе логического символа (будем называть его символом привязки) следует отдавать предпочтение из нескольких вариантов тому символу, который реже встречается в задачах - при этом меньшее время будет затрачиваться на попытки применения приема в ситуациях, когда его срабатывание заведомо невозможно.

Следующий важный шаг - выбор значения либо нескольких значений текущего уровня сканирования, при которых будет предприниматься попытка применения приема. Здесь следует исходить из соображений о приоритетности попыток применения приемов и учитывать ранее введенные уровни срабатывания конкурирующих приемов (находя эти приемы, например, по оглавлению базы приемов ГЕНОЛОГа либо анализируя процессы решения задач, в которых желательно либо нежелательно срабатывание нового приема).

Несколько различных уровней срабатывания приема используются в следующих случаях:

а) Выделяются несколько качественно различных ситуаций, с различными приоритетами срабатывания; для каждой из этих ситуаций (распознаваемых дополнительно внутри программы приема) предусматривается свой уровень срабатывания;

б) Вводится дополнительный контроль возможных изменений в задаче, при которых ранее невозможное либо нежелательное применение приема (заблокированное при выходе на программу с малым значением текущего уровня) впоследствии (при выходе на программу с бóльшим значением текущего уровня) становится возможным и целесообразным. Например, если для срабатывания приема необходимо наличие какой-либо дополнительной посылки задачи, отсутствовавшей изначально (при малом значении текущего уровня), но выведенной впоследствии, то дополнительная попытка применения приема на большем текущем уровне приведет к необходимому срабатыванию. Альтернативой такому средству контроля является уменьшение весов термов, содержащих символ привязки данного приема, как только текущие преобразования задачи (например, вывод следствий) делают вероятной целесообразность его применения.

В нашем примере выберем уровень срабатывания приема равным 1 - на этом уровне обычно применяются преобразования типа "общей стандартизации" термов задачи.

Следующий шаг при программировании на ЛОСе приема сканирования задачи - поиск в программе логического символа привязки той точки, в которой следует вставить оператор перехода ("ветвь N" либо "иначе N") для ветви, содержащей собственно новую программу. Здесь следует попытаться использовать как можно большее число операторов, уже введенных для программ других приемов, и выбрать точку ответвления как можно "глубже". Впрочем, выбор этой точки должен происходить таким образом, чтобы одна из программ не "портила" значений программных переменных, используемых другой программой - как правило, весьма необременительное ограничение, так как приемы на ЛОСе крайне редко прибегают к "переопределению" значений ранее определенных переменных. При вставке оператора "ветвь N"следует учитывать, что этот оператор является перечисляющим, и если после него в программе находились операторы сброса заданного числа внешних перечислений ("обрыв", "сброс(n)"), в "интервал перехода"которых попадает точка вставки, то необходимо увеличить на 1 константу сброса этих операторов. В приемах сканирования задач операторы сброса практически не используются, и здесь указанная ситуация не возникает.

Мы опустим здесь шаг поиска в уже существующей программе символа "синус" точки ответвления для нашего приема и приведем (в учебных целях) программу приема начиная с самого первого ее оператора.

Стандартное начало программы приема сканирования задачи имеет вид "решить стандарт(x2)". Оператор "решить"является истинным , если обращение к программе символа произошло из сканирования задачи; оператор "стандарт(x2)"проверяет, что вхождение x2 этого символа сопровождается открывающей скобкой (этот оператор используется только в тех случаях, когда рассматриваемый логический символ является символом операции либо предиката; вхождение такого символа без открывающей скобки может использоваться в построениях структурного уровня, описывающих вид термов).

Далее размещается фиксирующий уровень обращения к приему оператор "уровень(1)".

После выполнения указанных выше трех предварительных операторов начинается основная часть приема - идентификация ситуации, в которой возможно его применение. Мы имеем на текущий момент следующие определенные значения программных переменных: x1 - текущая задача; x2 - вхождение в некоторый ее терм логического символа "синус"; x3 - вхождение этого терма (называем его текущим) в соответствующий список задачи (в список условий либо список посылок либо в саму задачу); x4 - равно 0, если текущий терм является посылкой, и 1, если он является условием; x5 - значение текущего уровня сканирования (в нашем случае равно 1). Значение переменной x6 пока не определено. Начнем идентификацию указанной в теореме ситуации с того, что перейдем от x2 к вхождению x6 внешней для синуса операции: "операнд(x6 x2)". Эта операция должна быть степенью: "символ(x6 степень)". Показатель данной степени равен 2: "второйсимвол(x6 2)".

Далее возникает необходимость идентифицировать коэффициент a перед квадратом синуса. Снова рассматриваем внешнюю для x6 операцию x7: "операнд(x7 x6)". Эта операция, однако, не обязательно должна быть умножением. Приходится использовать выражение "вариант", определяющее значение x8 коэффициента a в зависимости от символа по вхождению x7: "равно(x8 вариант(символ(x7 умножение) исключениеоперанда(x7 x6)набор(1)))". Заметим, что если $a = 1$, то x8 имеет своим значением не сам символ 1, а одноэлементный набор из 1 (т.е. однобуквенный терм). Если же по вхождению x7 расположен символ "умножение", то a получается из данного произведения вычеркиванием сомножителя - квадрата синуса.

В действительности только что определенное значение x8 - это еще не коэффициент a, так как необходимо учесть знак слагаемого с квадратом синуса. Поэтому рассматриваем вхождение x9 внешней операции при умножении. Здесь, однако, удобно применить оператор "альтернатива": "альтернатива(символ(x7 умножение)операнд(x9 x7)равно(x9 x7))". Если по вхождению x7 не была расположена операция "умножение", то x7 уже является тем вхождением, на котором может быть расположен "минус". Наконец, определяем значение x10 коэффициента a с учетом знака: "равно(x10 вариант(символ(x9 минус)запись(минус x8)x8))".

Теперь переходим к внешней операции, которая должна оказаться суммой: "операнд(x11 x9) символ(x11 плюс)". Далее просматриваем слагаемые найденной суммы, отличные от слагаемого с квадратом синуса: "операнд(x11 x12) не(равно(x12 x9))". x12 - вхождение, которое следует идентифицировать с $a(\cos x)^2$. Прежде всего убеждаемся в том, что знак этого слагаемого совпадает со знаком исходного слагаемого: "альтернатива(символ(x9 минус)символ(x12 минус)не(символ(x12 минус)))". Здесь можно было бы использовать более экономное "символ(x12 буква(x9))", так как не просто знаки, но и заголовки двух слагаемых должны совпадать.

Далее определяем вхождение x13 второго слагаемого с отброшенным знаком "минус"(если он есть): "равно(x13 вариант(символ(x12 минус)первыйоперанд(x12)x12))". Определяем вхождение x14 во второе слагаемое, которое будет идентифицироваться с квадратом косинуса: "альтернатива(символ(x13 умножение)операнд(x13 x14)равно(x14 x13))". Проводим идентификацию для квадрата косинуса: "символ(x14 степень) первыйсимвол(x14 косинус) второйсимвол(x14 2) равныетермы(первыйоперанд (первыйоперанд(x14)) первыйоперанд(x2))". Последний оператор здесь проверяет совпадение аргументов синуса и косинуса. Проверяем совпадение коэффициентов (с

отброшенными знаками) при квадратах синуса и косинуса: "равно(х8 вариант(символ(х13 умножение)исключениеоперанда (х13 х14)набор(1)))".

Теперь переходим к программированию завершающей части приема, выполняющей необходимую замену. Прежде всего, заметим, что необходимо найти все слагаемые, отличные от двух выделенных, и учесть их в заменяющем терме, на который будет заменена сумма, расположенная по вхождению х11. Список этих слагаемых равен "выписка(х15 и(операнд(х11 х15)не(равно(х15 х9))не(равно(х15 х12)))подтерм (х15))". Соответственно, определяем заменяющий терм х15: "равно(х15 унисборка(плюс префикс(х10 выписка(х16 и(операнд(х11 х16)не(равно(х16 х9))не(равно(х16 х12)))подтерм(х16))))))". Операторное выражение "унисборка"применяется здесь из-за того, что число слагаемых в результирующем терме может оказаться равным 1, и тогда внешняя операция "плюс"будет опущена. Возможно альтернативное определение заменяющего терма при помощи операторного выражения "склейкаоперандов": "равно(х15 склейкаоперандов(х11 х9 х12 х10))".

Наконец, собственно оператор, выполняющий замену суммы по вхождению х11 на выражение х15: "заменавхождения(х11 х3 х4 х1 х15 пустоеслово)". Далее располагаем завершающий программу оператор "пересмотр", инициирующий повторное рассмотрение задачи с текущим уровнем, равным 0.

Основная часть приведенной программы выполняла усмотрение ситуации для применения заданного тождества - идентификацию его переменных с термами задачи. Программы такого типа, а их подавляющее большинство, создаются в решателе с помощью компилятора ГЕНОЛОГа на основе теорем предметной области, снабженных определенной технической разметкой. Непосредственно на ЛОСе обычно реализуются лишь приемы общелогического характера, не укладывающиеся в рамки стандартных алгоритмических построений ГЕНОЛОГа. В оставшейся части раздела мы рассмотрим несколько простых примеров таких программ, оставляя рассмотрение программ приемов, основанных на теоремах, до разделов, посвященных ГЕНОЛО́Гу.

Первый пример - программа приема, исключающего переменную связывающей приставки квантора общности, для которой в антецедентах этого квантора указано ее явное выражение. Заменяемое утверждение здесь имеет вид

$$\forall_{x_1 \ldots x_n}(x_i = t \ \& \ A_1(x_i) \& \ldots \& \ A_m(x_i) \Rightarrow A_0(x_i)),$$

где выражение t не содержит переменной x_i; заменяющее - вид

$$\forall_{x_1 \ldots x_{i-1} x_{i+1} \ldots x_n}(A_1(t) \& \ldots \& A_m(t) \Rightarrow A_0(t))$$

.

В качестве символа привязки выбираем "длялюбого" (равенство также могло бы претендовать на эту роль, но квантор общности встречается в задачах реже).

Так как прием выполняет вполне естественную общелогическую стандартизацию, его можно применять уже на нулевом уровне. Поэтому первые операторы программы имеют вид "решить стандарт(х2) уровень(0)". Так как возможность применения приема никак не связана с измененем его внешнего контекста, то далее размещается оператор "новый", который блокирует попытки повторного применения к терму задачи данного приема, если текущий уровень сканирования, после некоторой паузы, вновь поднялся до значения веса этого терма, возвратив его из теневой области в поле зрения.

Далее определяем список x6 переменных, относящихся к кванторной приставке квантора, и проверяем, что все они различны: "равно(x6 связприставка(x2)) различны(x6)". Находим список x7 антецедентов квантора (утверждения слева от стрелки $\Rightarrow$) и консеквент x8 этого квантора (утверждение справа от стрелки): "равно(x7 выписка(x8 антецедент(x2 x8)подтерм(x8))) равно(x8 последнийтерм(x2))".

Чтобы проверить, имеется ли среди антецедентов равенство, определяющее значение переменной кванторной приставки, просматриваем элементы x9 списка антецедентов: "входит(x9 x7)".

Начинаем идентификацию антецедента x9 с равенством требуемого вида: "заголовок(x9 равно) операнд(левыйкрай(x9)x10) символ(x10 x11) входит(x11 x6)". Здесь x10 - вхождение того операнда равенства, который оказался переменной x11, входящей в список x6. Эта переменная идентифицирована с x_i. Чтобы определить другой операнд равенства, используем операторы: "операнд(левыйкрай(x9) x12) не(равно (x10 x12)) равно(x13 подтерм(x12))". Здесь мы применяем перечисляющий оператор "операнд", выдающий последовательно вхождения каждого из двух операндов, и далее отбираем то вхождение x12, которое отлично от вхождения x10 уже рассмотренного операнда. x13 - переписанное в виде отдельного терма выражение, идентифицированное с t. Используя оператор "Операнды", можно было бы выполнить идентификацию равенства несколько короче: "заголовок(x9 равно) Операнды(левыйкрай(x9)x10 x11) символ(x10 x12) входит(x12 x6) равно(x13 подтерм(x11))"(здесь x_i идентифицируется с x12, t - с x13). Далее мы продолжаем программу для первого варианта идентификации.

После того, как найдены x_i и t, определяем набор утверждений $A_1(t), \ldots, A_m(t)$. Так как внутри выражения t могли встречаться связанные переменные, то после подстановки его вместо x_i в какое-либо $A_j(x_i)$, также имевшее связанные переменные, может оказаться, что некоторое вхождение переменной в $A(t)$ имеет два вложенных друг в друга внешних квантора либо описателя по этой переменной. Такое двойное связывание одной и той же переменной может повлечь за собой логические ошибки, так как для ускорения ряда процедур решателя делается допущение о невозможности двойного связывания либо одновременного использования в общем контексте переменной как свободной и как связанной. Это допущение обеспечивается своевременной нормализацией обозначений, и в данном случае лучше сразу же переобозначить связанные переменные в $A_j(x_i)$, сделав их отличными от переменных выражения t. Например, для этого можно использовать операторное выражение "новыесвязки(T S)", значением которого служит результат переобозначения в терме T связанных переменных на переменные, не встречающиеся в термах списка S. Таким образом, для нахождения списка x14 утверждений $A_1(t), \ldots, A_m(t)$ имеем оператор "равно(x14 выписка(x15 x16 и(входит(x15 x7)не(равно(x15 x9))результподст (набор(x11)набор(x13)новыесвязки(x15 набор(x13)) x16)) x16))". Он просматривает антецеденты x15, отличные от уже выделенного равенства x9, и применяет к результату указанного выше переобозначения связанных переменных в x15 подстановку x13 (т.е t) вместо x11 (т.е. x_i).

Аналогичную подстановку применяем к консеквенту, получая в качестве значения переменной x15 утверждение $A_0(t)$: "результподст(набор(x11)набор(x13)новыесвязки(x8 набор(x13))x15)".

Далее нужно найти остаток x16 связывающей приставки квантора общности после исключения из него переменной x_i: "равно(x16 вычеркивание(x6 набор(x11)))".

Теперь остается построить заменяющее утверждение из найденных списка его антецедентов x14, консеквента x15 и новой связывающей приставки x16. Однако, здесь следует учитывать вырожденный случай: если исходный квантор имел лишь одну переменную в связывающей приставке, то заменяющее утверждение нужно строить без использования квантора общности, как дизъюнкцию консеквента и отрицаний всех антецедентов. Учитывая это, получаем следующий оператор для формирования заменяющего утверждения x17: "равно(x17 вариант(равно(x16 пустоеслово) унисборка(или суффикс(выписка(x18 входит(x18 x14) отрицание(x18))x15))сборка (длялюбого конкатенация(x16 вариант(равно(x14 пустоеслово)набор(x15)конкатенация(если x14 то набор(x15)))))))". Если связывающая приставка x16 пустая, то для получения x17 соединяем связкой "или"отрицания утверждений набора x14, к которым в конце добавляем x15. Если она непустая, то в случае пустого списка новых антецедентов x14 сразу после связывающей приставки помещаем консеквент x15, иначе - сначала вставляется логический символ "если", затем идут антецеденты, затем логический символ "то", и в конце - консеквент x15. В конце программы приема размещаются операторы "заменавхождения(x2 x3 x4 x1 x17 пустоеслово)" (фактическая замена утверждений) и "пересмотр".

Второй пример простого общелогического приема, реализованного на ЛОСе, связан с решением задачи на доказательство Z, условие которой имеет вид кванторной импликации:

$$\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\&\, A_m \Rightarrow A_0).$$

Для решения такой задачи создается вспомогательная задача Z' на доказательство, посылки которой суть все посылки задачи Z, а также антецеденты $A_1, \ldots, A_m$. Условием новой задачи на доказательство служит консеквент A_0. Так как задачи Z и Z' эквивалентны, то после решения второй сразу выдается ответ (либо отказ) на первую. Попутно выполняется коррекция тех структур данных задачи Z, в которых регистрируется, какие из ее посылок в действительности были использованы при доказательстве.

В данном приеме символом привязки естественно выбрать "длялюбого"; уровень срабатывания будет равен 3 (чтобы успели сработать приемы общей стандартизации, которые могли бы существенно упростить кванторную импликацию или вовсе исключить ее квантор).

Первые операторы программы приема нам уже знакомы: "решить стандарт(x2) уровень(3) новый". Чтобы указать на то, что кванторная импликация размещается в условиях задачи и имеет корневое вхождение, далее добавляем операторы "равно(x4 1) корень(x2)". Следующий оператор усматривает задачу на доказательство: "тип(x1 доказать)".

Определяем список x6 антецедентов кванторной импликации и находим результат x7 добавления к нему всех посылок текущей задачи: "равно(x6 выписка(x7 антецедент(x2 x7)подтерм(x7))) равно(x7 конкатенация(x6 списокпосылок(x1)))".

Теперь можно формировать новую задачу Z'. Ее можно было бы создать без какого-либо использования комментариев задачи Z - с пустыми списками комментариев. Однако, комментарии задачи Z, возможно хранят какую-то ценную информацию, для повторного получения которой потребовались бы дополнительные затраты вычислительной трудоемкости. Перенесение всех ее комментариев в задачу Z' нецелесообразно, так как она все же отличается от Z, и какие-то комментарии могут в

ней утратить свою ценность либо оказаться ложными (хотя такая ложность обычно не сказывается на корректности действий с посылками и условиями задачи, она может существенно исказить логику принятия решений при управлении логическим процессом). Поэтому при создании задачи Z' следует проанализировать старые комментарии с помощью какой-то процедуры и отобрать либо модифицировать их для использования в Z'. Обычно здесь используются справочники, которые распадаются на множество мелких подпроцедур, каждая из которых анализирует и при необходимости модифицирует комментарии с заданным заголовком. В данном приеме введены два таких справочника, названия которых суть "новаяпосылка"и длялюбого". Первый из них обрабатывает комментарии к посылкам задачи, второй - комментарии к самой задаче. Справочник "новаяпосылка"имеет входные переменные x1 - задачу Z; x2 - либо 0 (если анализируется общий комментарий к посылкам), либо вхождение посылки, к которой относится комментарий, в список посылок; x3 - анализируемый комментарий. Если перенесение комментария в новую задачу нецелесообразно, то справочник выдает значение 1; если комментарий переносится без изменения, то выдается 0, в остальных случаях выдается модифицированный комментарий x3. Аналогично устроен справочник "длялюбого", но у него только две входных переменных: x1 - задача; x2 - комментарий. Фактическое заполнение двух указанных справочников процедурами обработки комментариев выходит за рамки нашего примера. Оно может быть продолжено по мере обучения решателя и появления комментариев новых типов. Так как случаи обязательного отбрасывания или коррекции комментариев довольно редки, число реально написанных процедур этих двух и аналогичных им справочников обработки комментариев, используемых в других приемах, невелико.

Оператор, формирующий новую задачу x8 (т.е. Z'), достаточно громоздок; в отсутствии многоцветной указки решателя, будем здесь и далее в таких случаях использовать разбиение записи оператора на блоки. Прежде всего, нам понадобится операторное подвыражение A_1, определяющее список списков комментариев к посылкам новой задачи. Так как новый список посылок получился присоединением к началу старого набора антецедентов x6, которые пока не сопровождаются комментариями, то A_1 представляем в виде конкатенации набора пустых слов, имеющего ту же длину, что и x6, и набора, составленного из модифицированных списков комментариев задачи Z: "конкатенация(бланк(x6 пустоеслово 0)выписка(x9 позиция(x9 левпозиция (x1 3))A_2))". Подвыражение A_2 здесь определяет модификацию текущего (расположенного по вхождению x9) списка комментариев к посылке (или посылкам) старой задачи. Эта модификация получается при просмотре элементов x10 данного списка: "выписка(x10 x11 и(входит(x10 буква(x9))равно(x11 A_3)не(равно(x11 1))вариант(равно(x11 0)x10 x11)))". x11 - результат обращения к справочнику "новаяпосылка"для обработки комментария x10; он определяется подвыражением A_3: "справка(новаяпосылка начало(x10)x1 вариант(равно(правсосед(x9)x9)0 соотвпозиция(x9 левпозиция (x1 3)списокпосылок(x1))) x10)". Условие "равно(правсосед(x9)x9)" здесь означает, что вхождение x9 списка комментариев - последнее, то есть он является списком общих комментариев к посылкам задачи Z. Выражение "соотвпозиция(x9 левпозиция(x1 3)списокпосылок(x1))" определяет вхождение той посылки, которой соответствует вхождение x9 списка комментариев.

Далее строим подвыражение A_4 для списка комментариев к задаче Z': "выписка (x10 x11 и(входит(x10 комментарии(x1)) равно(x11 A_5)не(равно(x11 1)))вариант(равно(x11 0)x10 x11))". Здесь A_5 - обращение к справочнику "длялюбого", анализирующему старый комментарий x10: "справка(длялюбого начало(x10)x1 x10)".

Используя блоки A_1, A_4 получаем оператор для построения задачи Z': "равно(х8 набор(доказать х7 бланк(х7 0 0)A_1 последнийтерм(х2) 0 A_4))".

Далее организуем обращение к решению задачи Z'. Прежде всего, применяем оператор "подуровень(х1)", который устанавливает максимальный уровень новой задачи равным максимальному уровню задачи Z (так как она целиком сводится к Z', последнюю следует решать с привлечением средств того же уровня, что и для Z). Далее размещается оператор "равно(ответзадачи(х8)истина)", который одновременно решает задачу Z' и проверяет, что ее ответом служит логический символ "истина". После него идет оператор перехода "иначе 1" (в каждом фрагменте переходы из него нумеруем начиная с 1), определяющий переход к фрагменту, состоящему из единственного оператора "ответ(отказ)". Этот отказ на задачу Z будет выдан, если на задачу Z' не был получен ответ "истина".

После оператора перехода "иначе 1" остается выдать ответ "истина" на задачу Z, предварительно скорректировав ее комментарий "выводимо A". В списке A перечисляются все посылки задачи Z, использованные при ее решении, причем они берутся в том виде, какой они имели на начальный момент решения задачи Z. Оператор, выполняющий коррекцию, имеет вид: "длялюбого(х9 если ключ(комментарии(х8)выводимо х9)то учетпосылок(х1 конец(х9)0 1))". Он извлекает из задачи Z' список "конец(х9)"ее исходных посылок, использованных при решении, и определяет, из каких именно исходных посылок задачи Z они получены. При этом используется несложный вспомогательный оператор "учетпосылок(t_1 t_2 t_3 t_4)". Программу приема завершает оператор "ответ(истина)".

В заключение рассмотрим прием решения задач на описание, в которых нужно подобрать какой-нибудь пример значений неизвестных, удовлетворяющих условиям. Прием предпринимает попытку усмотреть эти значения непосредственно из посылок задачи. Если удается найти такие посылку A и условие B, что A представимо (с точностью до изменения порядка операндов коммутативных операций и симметричных отношений) как результат подстановки в условие B вместо всех неизвестных $x_1, \ldots, x_n$, имеющихся в B, выражений $t_1, \ldots, t_n$, то эти выражения фиксируются в качестве значений неизвестных, причем найденные значения подставляются в оставшиеся условия. Если после подстановки в каком-либо условии не остается неизвестных, то его истинность проверяется с помощью решения вспомогательной задачи на доказательство; процесс подбора значений и проверки продолжается до полной реализации всех условий.

Данный прием, в отличие от трех предыдущих, не может быть закреплен за каким-либо определенным логическим символом, встречающимся в условиях и посылках задач. Поэтому он закрепляется за типом задачи - логическим символом "описать". Уровень его применения не должен быть ни чрезмерно большим - так как тогда слишком поздно будет происходить усмотрение очевидных наборов значений, ни чрезмерно маленьким - так как во многих случаях имеются более простые средства получения примера значений неизвестных, а действия приема сравнительно трудоемки. В решателе этот прием имеет уровень срабатывания 4.

На примере данного приема продемонстрируем, как происходит поиск точки ответвления к приему от уже имеющихся фрагментов программы логического символа. Если войти в корневой фрагмент программы символа "описать", то будут видны операторы "решить иначе 1 равно(х2 х3) ветвь 2 ...". Оператор "решить" означает, что обращение к программе символа "описать" происходит из сканирования задачи

- то есть программа нашего приема должна быть размещена после этого оператора. Переход "иначе 1"ведет к рассмотрению других возможных типов обращения к данной программе.

Обычно программа логического символа имеет главный ствол, на котором размещаются операторы, фиксирующие тип обращения; от каждого такого оператора к следующему имеется переход через "иначе 1".

После оператора "решить"расположен оператор "равно(x2 x3) его роль заключается в отсечении ситуаций, где обращение к программе символа "описать" произошло из-за появления этого символа в условии либо посылке задачи (например, если решается задача относительно структур данных самого решателя). В указанных ситуациях значением x2 было бы вхождение символа "описать" в терм задачи, а значением x3 - вхождение терма задачи, то есть эти значения различались бы. Совпадают они только при обращении к программе символа "описать" из рассмотрения типа текущей задачи - тогда x2 и x3 равны 0. При поиске точки ответвления к приему мы проходим данный оператор и перемещаемся вдоль программы дальше.

Затем идет оператор "ветвь 2", после которого располагаются операторы, никак не связанные с нашим приемом. Поэтому переходим к следующему фрагменту через "ветвь 2".

Новый фрагмент начинается с двух операторов перехода "ветвь 1", "ветвь 2", после которых идет оператор "уровень(0)". Он отсекает в нашем случае (т.е. при уровне 4) дальнейшее продвижение, так что нужно воспользоваться одним из указанных переходов. Сразу заметим, что первый из них группирует приемы, для которых уровень срабатывания либо очень велик, например, равен максимальному уровню задачи, либо почти не фиксирован (например, попытка применить прием выполняется для каждого уровня, начиная с некоторого). Во втором размещаются обычные приемы, для срабатывания которых выделены один или два-три уровня. Поэтому для дальнейшего продвижения выбираем переход "ветвь 2".

Очередной фрагмент начинается с операторов "ветвь 1 уровень(2 3)", так что однозначно переходим по оператору "ветвь 1". Далее идет фрагмент с началом вида "уровень(1 3)иначе 1", и снова однозначно переходим через "иначе 1".

Наконец, возникает фрагмент с началом "уровень(4)иначе 1 ветвь 2". Собственно говоря, этот фрагмент является как раз началом той программы, которую мы хотим получить в данном примере. Если бы это было не так, то можно было бы продолжить перемещение вдоль его операторов, пытаясь максимально использовать уже имевшиеся ранее операторы (они "бесплатные", так как все равно будут выполняться при сканировании). Встретив где-либо операторы, использование которых невозможно или нежелательно, можно было бы создать ветвь программы нового приема двумя способами - либо вставить оператор перехода, ведущий к этой ветви, либо сделать так, чтобы старая часть программы сама оказалась достижима через оператор "ветвь (...)"из новой программы. Последнее достигается стандартной операцией - переходя в режим просмотра подтермов операторов, выбираем тот оператор, который должен быть началом отдельной ветви "старой" программы, и нажимаем "p" (кир.). После этого конец фрагмента отрезается и попадает в отдельную ветвь. Начало фрагмента завершается операторами "ветвь(...)" и "продолжение". Войдя в режим редактирования, далее вместо оператора "продолжение" набираем текст новой программы.

Возвращаемся к программе нашего приема - приводим ее далее в точности такой, как она указана в фрагменте, до которого мы дошли указанным выше образом. Сле-

дующий ее оператор - "цель(x1 пример)" - означает, что текущая задача на описание должна иметь цель "пример". Расположенный за ним оператор перехода "иначе 3" ведет к программе некоторого другого приема, тоже относящегося к задачам на описание с целью "пример". Далее располагаются операторы "не(цель(x1 развертка)) не(цель(x1 редакция)) не(цель(x1 перечисление)) не(Входит(независит цели(x1)))". Все эти операторы отсекают ситуации, в которых применение данного приема не нужно (например, при редактировании уже найденного ответа) - они появились не при первоначальном создании программы приема, а вводились по мере надобности при проработке обучающего материала. Они являются эвристическими фильтрами, и при развитии системы могут оказаться измененными либо вообще удаленными. Для понимания того, зачем нужен тот или иной фильтр, необходимо предъявление конкретных контекстов, их создавших, так что сейчас мы не будем комментировать перечисленные оперторы.

Следующий оператор - "лимит(набор(3 0 0 0 0 0)) выполняет роль ограничителя времени, которое будет затрачено на попытку непосредственного подбора значений неизвестных. Эта попытка будет прервана ровно через 300000 шагов работы интерпретатора после своего начала, и программа вернется к данному оператору "лимит", рассматривая его уже как ложный. Константа 300000 тоже является эвристической (выбрана из рассмотрения тех задач, в которых данный прием должен был сработать), и при развитии решателя может быть изменена. Она не очень велика - трудоемкость решения простых и средних задач обычно составляет один-два десятка миллионов шагов.

Чтобы определить список x6 всех неизвестных задачи, встречающихся в ее условиях, используем оператор "равно(x6 пересечениесписков(окончание(неизвестные(x1))параметры (списокусловий(x1))))". Напомним, что выражение "неизвестные(x1)" дает набор (неизвестные $x_1 \dots x_n$), и для выделения собственно списка неизвестных $x_1 \dots x_n$, у этого набора нужно отбросить первый элемент, что и делается операторным выражением "окончание(...)". Подбор значений имеет смысл предпринимать лишь в случае непустого списка неизвестных, так что далее размещается оператор "не(равно(x6 пустоеслово))".

Прежде, чем переходить к подбору значений переменных списка x6, проверяем, что все не содержащие неизвестных условия задачи являются следствиями ее посылок, и одновременно создаем список x7 всех посылок, использованных при проверке истинности таких условий: "равно(x7 пустоеслово) длялюбого(x8 если входит(x8 списокусловий(x1)) известно(x8 x1)то существует(x9 и(извлекается(x8 списокпосылок(x1)x9) замена(x7 объединениесписков(x7 x9)))))".

Далее определяем список x8 всех условий задачи, содержащих неизвестные, и проверяем, что все они суть элементарные (бесквантворные, не содержащие связок "и","или") утверждения: "равно(x8 выписка(x9 и(входит(x9 списокусловий(x1)) не(известно(x9 x1)))x9)) длялюбого(x9 если входит(x9 x8)то элементарно(x9))".

Наконец, обращаемся к оператору "подборнеизвестных", уже упоминавшемуся в главе "Библиотека вспомогательных операторов ЛОСа" : "подборнеизвестных(списокпосылок(x1)x8 x6 x9 x10)". Затем регистрируем в комментарии "выводимо A" к задаче использованные на данном шаге исходные посылки (для этого имеем списки использованных утверждений x7 и x10): "учетпосылок(x1 объединениесписков(x7 x10)0 1)".

Выходная переменная x9 оператора "подборнеизвестных"имеет своим значением набор $t_1 \dots t_n$ выражений, определяющих значения переменных $x_1 \dots x_n$ списка x6 (расположенных в том же порядке). На основе списков x6 и x9 создаем ответ задачи - утверждение $x_1 = t_1 \,\&\dots\, \&\, x_n = t_n$: "равно(x11 унисборка(и выписка(x12 x13 серия(x12 x6 x13 x9)запись(равно буква(x12)буква(x13)))))"; в конце программы помещаем оператор выдачи ответа - "ответ(x11)".

8.2 Примеры программ вспомогательных операторов и операторных выражений

Рассмотрим пример вспомогательного оператора, осуществляющего поиск в некотором списке термов всех его элементов A, имеющих хотя бы одно вхождение заданного терма T, и выдающего в качестве результата список пар: (A - список всех вхождений в A терма T, упорядоченных слева направо). Входные данные этого оператора суть: x1 - набор термов; x2 - заданный терм T. Оператор имеет единственную выходную переменную x3 - ей присваивается указанный выше набор пар.

Первый шаг при программировании вспомогательного оператора - выбор названия для этого оператора. Через главное меню осуществляется ввод нового логического символа - названия оператора (в отдельных случаях можно использовать ранее введенный логический символ, если для него не была создана программа вспомогательного оператора; наличие такой программы распознается по оператору "программа"в начале фрагмента). В нашем примере введем для названия программы новый логический символ "вхождениятерма".

После выбора названия A оператора следует войти в просмотр корня программы логического симвла A и выбрать точку ответвления для записи программы оператора A. Если программы логического символа A вообще не было, то программирование начинается прямо с корня; если, например, корневой фрагмент имел вид "решить ...", то он изменяется на "решить иначе N...", где N - номер перехода к новой программе (N выбирается произвольно - лишь бы отличалось от номеров других, ранее введенных переходов из корневого фрагмента).

Первые операторы в нашем примере имеют вид: "программа метка(икс(4))". Здесь оператор "метка(икс(4))"указывает, что номер первой не определенной при обращении программной переменной равен 4. Это оператор является фиктивным и интерпретатор ЛОСа его игнорирует. Однако, он важен для процедур контроля ошибок в программах ЛОСа, так как позволяет распознать уже определенные при обращении переменные. Кроме того, при наличии оператора "метка(икс(...))", проверка новой программы оператора или операторного выражения приводит к автоматическому созданию приема справочника "арность", указывающему, сколько операндов должен иметь оператор. Этот справочник тоже нужен для контроля ошибок в программах. Наконец, система контроля ошибок в программах устроена так, что само отсутствие в начале программы оператора или операторного выражения фиктивного оператора "метка(икс(...))"воспринимается как ошибка.

Для получения требуемого списка пар - значения выходной переменной нашего оператора - организуем просмотр списка x1. Для текущего терма из x1 составим список всех вхождений в него терма T, и если этот список непуст, то зарегистрируем его в

накопителе результата. Все эти действия выполняются единственным оператором выдачи результата: "результат(х3 выписка(х4 х5 и(входит(х4 х1)равно(х5 выписка(х6 вхождениетерма(х4 х2 х6)х6))не(равно(х5 пустоеслово)))набор(х4 х5)))". Просмотр списка х1 здесь обеспечивается перечисляющим оператором "входит(х4 х1)"; список х5 всех вхождений в х4 терма х2 (т.е. T) составляется при помощи перечисляющего оператора "вхождениетерма".

После указанного выше оператора "результат(...)"необходимо поставить завершающий программу оператор "выход", так как иначе данный оператор "вхождениятерма" выполнялся бы в режиме перечисления (в данном случае по определению оператора может быть выдано лишь одно значение переменной х3, и режим перечисления не нужен).

Если программа вводилась для нового логического символа, то после ее набора можно выйти в корневой фрагмент и нажать клавишу "п". Тогда (как говорилось выше) будет автоматически введена программа справочника "арность", определяющая арность (число операндов) у нового оператора - эта программа будет использоваться при при контроле правильности программ. Одновременно будет выполнена проверка правильности программы (анализ арностей операторов; учет определенности либо неопределенности входных и выходных переменных операторов программы, и т.п.).

Следующий пример - программа оператора, работающего в режиме перечисления значений своих выходных переменных. Такие операторы полезны при программировании процедур, осуществляющих поиск в сложных структурах данных: собственно просмотр необходимых групп объектов осуществляется единственным обращением к оператору. Перечисляющие операторы избавляют от необходимости создавать специальные конструкции для организации циклов; особенно упрощает программирование их использование при перечислениях по рекурсии. Чтобы программа оператора заработала в режиме перечисления, достаточно после оператора "результат(...)", определяющего значения выходных переменных, не ставить оператор "выход". Тогда при откате произойдет возвращение в данную программу - в последний из ее перечисляющих операторов, который был выполнен перед выдачей результата. Хотя формально можно часть выходов из программы делать перечисляющими, а часть - неперечисляющими, и интерпретатор ЛОСа будет правильным образом обрабатывать такие выходы, этот режим программирования на ЛОСе оказался фактически невостребованным. Операторы со смешанным перечислением оказались неудобны из-за трудностей, возникающих при подсчете числа сбрасываемых внешних перечислений в явно указываемых откатах. Поэтому в операторах, программируемых на ЛОСе, режим смешанного перечисления вообще не используется. Процедура контроля программ выдает сообщение об ошибке, если часть выходов из программы оператора - перечисляющие, а часть - не перечисляющие. Среди базисных операторов ЛОСа наиболее часто порождает смешанный режим перечисления оператор "альтернатива".

Рассмотрим в качестве примера программирование следующего простого перечисляющего оператора, связанного с одночленами. Входными данными нашего оператора будут некоторый набор выражений х1, а также набор х2 натуральных чисел, имеющий ту же длину, что и х1. Выходная переменная х3 перечисляет произведения целых неотрицательных степеней выражений списка х1, показатели которых не превосходят соответствующих этим выражениям разрядов набора х2. Одновременно выходная переменная х4 приобретает в качестве значения произведение степеней выражений списка х1, показатели которых дополняют показатели произведений х3 до

значений набора x2. Этот пример уже реализован в системе как программа оператора "Делитель".

Первые операторы программы суть "программа метка(икс(5))". При вычислении произведений x3,x4 будем использовать рекурсию по длине наборов x1,x2. Поэтому следующие операторы программы суть "равно(x5 окончание(x1))"; "равно(x6 окончание(x2))" - заблаговременно определяем результаты x5,x6 отбрасывания первых элементов наборов x1,x2 для рекурсивного обращения к обработке укороченных наборов. Далее будем последовательно рассматривать возможные значения показателя степени при первом элементе A набора x1 - от 0 до первого элемента n набора x2. Вводим операторы: "равно(x7 0) равно(x8 начало(x2))". Здесь x7 - текущее значение показателя степени; x8 - значение, дополняющее его до n.

Далее будет организован цикл по x7 с применением оператора "повторение"(этот цикл можно было бы не создавать, а использовать вместо него перечисляющий оператор "Номера(0 начало(x2)x7)", который перечислял бы все целочисленные значения x7 от нулевого до максимального; однако, здесь полезно показать, как на ЛОСе происходит организация "явных" циклов). После оператора "повторение" разместим оператор перехода "ветвь 1"; нумерацию переходов в этом примере берем сквозную. По этому оператору будем переходить в ветвь программы, обеспечивающую очередное изменение x7,x8 и выход из цикла. После оператора "ветвь 1" размещаем оператор "равно(x5 пустоеслово) иначе 2 ", который обеспечивает раздельную обработку в случае завершения рекурсии (если список x5 пуст) и в случае рекурсивного обращения к укороченным наборам.

Если рекурсия завершена, то непосредственно выдается результат - здесь используется оператор "результат(x3 вариант(равно(x7 0)набор(1)вариант(равно(x7 1)начало(x1)запись(степень начало(x1)десзапись(x7))))x4 вариант(равно(x8 0)набор(1)вариант(равно(x8 1)начало(x1)запись(степень начало(x1)десзапись(x8)))))". Как видно, сначала формируется вырожденное произведение x3, с разбором случаев: $x7 = 0$ (выдается константа 1), либо $x7 = 1$ (выдается выражение A - начало набора x1), либо $x7 > 1$ (выдается степенное выражение). Затем аналогичным образом формируется произведение x4.

При рекурсивном обращении к укороченным наборам (переход к подфрагменту 2) программа имеет вид: "Делитель(x5 x6 x9 x10) результат(x3 вариант(равно(x7 0)x9 соединение(умножение вариант(равно(x7 1)начало(x1)запись(степень начало(x1)десзапись(x7)))x9)) x4 вариант(равно(x8 0)x10 соединение(умножение вариант(равно(x8 1)начало(x1)запись(степень начало(x1)десзапись(x8)))x10)))". Здесь x9,x10 - результат обработки пары укороченных наборов. Далее к x9,x10 добавляются необходимые множители, формируемые так же, как и выше. Добавление множителей происходит при помощи операторного выражения "соединение(...)", которое устраняет вложенные умножения.

Подфрагмент 1 имеет вид "равно(x8 0) иначе 3 стоп". Это - проверка окончания цикла при $x7 = n$. Наконец, подфрагмент 3 имеет вид "замена(x7 плюс(x7 1)) замена(x8 вычитание(x8 1)) продолжение" - он обеспечивает увеличение на единицу значения переменной x7 и уменьшение на единицу значения переменной x8. Напомним, что фактически в программе расположены операторы "замена(7 плюс(x7 1))", "замена(8 вычитание(x8 1))", а в указанным выше виде они выдаются на экран только для работы в редакторе программ. Отладчик ЛОСа, выдавая при трассировке на экран записи операторов программы, не выполняет такого преобразования для "замен".

В качестве примера программы операторного выражения рассмотрим процедуру, осуществляющую переход от предствления матрицы "по строкам"к ее представлению "по столбцам". Пусть x1 - набор $(A_1, \ldots, A_n)$ наборов $A_i = (a_{i1}, \ldots, a_{im})$, представляющих строки некоторой матрицы A. В качестве значения операторного выражения требуется получить набор $(B_1, \ldots, B_m)$ столбцов $B_j = (a_{1j}, \ldots, a_{nj})$.

Выберем прежде всего название для операторного выражения (например, "столбцы-матрицы") и введем его в качестве нового логического символа через главное меню.

Первые два оператора новой программы операторного выражения имеют вид "обращение(0) метка(икс(2))". Первый из них указывает, что происходит вычисление программного выражения; второй указывает на номер 2 первой не определенной при обращении программной переменной.

Собственно вычисление набора столбцов и выдача результата выполняются единственным оператором:

"ответ(выписка(x2 позиция(x2 начало(x1))выписка(x3 входит(x3 x1)буква(соотвпозиция(x2 начало(x1)x3)))))".

Здесь происходит просмотр всех вхождений x2 разрядов в первую строку матрицы, и для фиксированного такого x2 составляется список всех разрядов различных строк матрицы, расположенных соответственно разряду x2 первой строки. Значение операторного выражения, реализованного программой на ЛОСе, выдается при помощи оператора "ответ(...)". Если бы программа операторного выражения была завершена оператором "стоп"либо выход из нее произошел при откате, то в качестве значения был бы выдан логический символ 0.

Во многих случаях возникает необходимость в серии программ, обеспечивающих однотипные вычисления либо выдачу справочной информации для различных логических символов. Такая серия программ получает общее название - некоторый логический символ A - и называется справочником A. Обращение к справочнику A для обработки логического символа S имеет вид "справка(A S $x_1 \ldots x_n$)", где $x_1, \ldots, x_n$ - дополнительные входные данные (как уже говорилось, однотипные для всех программ справочника A). Это обращение представляет собой операторное выражение, так что справочник всегда возвращает в качестве результата некоторый объект (в справочной информации о логическом символе A этот объект обычно обозначается R). Если программа справочника A для логического символа S отсутствует, либо выход из нее осуществляется через оператор "стоп", либо при откате, то в качестве зачения возвращается логический символ 0. Одно из удобств при использовании справочников заключается в возможности независимого программирования его процедур, обслуживающих различные логические символы. Кроме того, время обращения к нужной процедуре справочника практически никак не зависит от числа ранее запрограммированных процедур этого справочника.

В качестве примера рассмотри справочник "одз", обеспечивающий получение условий на допустимые значения операндов некоторой операции либо некоторого предиката. Логическим символом S здесь служит указанная операция либо предикат; дополнительные входные данные таковы: x1 - задача, в которой встречается рассматриваемое вхождение S; (x2,x3,x4) - координата вхождения символа S в задачу x1 (x2 - вхождение S в терм задачи; x3 - вхождение этого терма в список задачи; x4 - указатель посылки (0) либо условия (1)).

Запрограммируем, например, процедуру справочника "одз", обслуживающую логический символ "тангенс". Первые два оператора программы (она вставляется в программу символа "тангенс") таковы: "обращение(одз) метка(икс(5))". Как обычно, здесь 5 - номер первой не определенной при обращении программной переменной. Первый оператор указывает название справочника.

Далее располагаются следующие операторы:

"равно(х5 первыйтерм(х2)) равно(х6 выписка(х7 или(и(не(существует(х6 и(равно (х6 справка(тип буква(первыйоперанд(х2))))не(равно(х6 0))входит(число х6))))равно (х7 запись(число х5)))равно(х7 запись(не запись(равно запись(косинус х5)0))))х7)) ответ(х6)".

Первый из них определяет терм х5 - операнд тангенса. Далее используется конструкция с оператором "выписка", в которой последовательно (с помощью оператора "или") формируются два элемента х7, регистрируемые в результирующем списке условий. Первый из этих термов х7 имеет вид "число(х5)"; второй - вид "не(равно(косинус(х5)0))". Чтобы избежать перечисления условий, очевидным образом уже выполненных в контексте рассматриваемого вхождения, регистрация условия "число (х5)"в списке предваряется проверкой того, что терм х5 не имеет своим заголовком операцию, принимающую только числовые значения. Для этой проверки используется справочник "тип".

Подробная информация о справочнике с названием A размещается в справочной информации логического символа A и при работе с редактором программ ЛОСа доступна, например, через F2. В поясняющих текстах информация о справочнике A обычно начинается со слов "A - индекс обращения.". При этом результат обращения к справочнику обозначается посредством R.

8.3 Упражнения по программированию на ЛОСе

8.3.1 Просмотр программ

1. Войти из главного меню в корневой фрагмент программы символа "синус".

2. Найти в программе символа "синус" фрагмент, содержащий оператор "уровень (1 3)". Найти фрагменты, содержащие операторы "уровень(2)" и "уровень(8)".

3. Найти все вхождения в программу символа "синус" оператора "символ(х20 косинус)". Сколько таких вхождений ?

4. Находясь в программе символа "синус", посмотреть справочную информацию для символа "синус". Находясь в той же программе, посмотреть справочную информацию для символа "степень".

5. Находясь в корневом фрагменте программы символа "синус" и используя только мышь, посмотреть справочную информацию символа "заменавхождения". Просмотреть все страницы этой информации.

6. Находясь в корневом фрагменте программы символа "синус" и не выходя в главное меню, перейти к просмотру корневого фрагмента программы символа "заменавхождения". После этого, опять же не выходя в главное меню, вернуться к корневому фрагменту программы символа "синус".

7. Находясь в корневом фрагменте программы символа "синус", перейти в оглавление операторов ЛОСа; найти в нем цепочку подразделов "Операторы для работы с логическими структурами данных" - "Списки термов" - "Лексикографическое упорядочение набора термов" и просмотреть справочную информацию для логического символа "лексупорядочение". Затем вернуться в программу символа "синус".

8. Войти в корневой фрагмент программы символа "заменавхождения"и перейти от него к подфрагменту вдоль цепочки переходов с номерами: 2, 1, 1, 2. Перейти к пункту оглавления программ, соответствующему ссылке "прием(5 9)", имеющейся в найденном фрагменте.

9. Войти в корневой фрагмент программы символа "синус" и, не выходя в главное меню, просмотреть корневые фрагменты программ нескольких следующих после синуса логических символов.

10. Войти в корневой фрагмент программы символа "биссектриса", перейти в режим просмотра подтермов операторов и выделить в предпоследнем операторе фрагмента многоцветной указкой подтерм "списокпосылок(x1)". Далее вернуться в режим обычного просмотра программы. Проделать эту операцию дважды - сначала только с помощью клавиатуры, затем - только с помощью мыши.

11. Войти в программу символа "синус"и, используя только клавиши курсора, найти фрагмент, содержащий оператор "обращение(тип)". Далее вернуться к корневому фрагменту программы "синус", используя единственное нажатие клавиши.

12. Войти из главного меню в оглавление операторов ЛОСа и найти в нем операторное выражение, определяющее номер вхождения в набор. Перейти в просмотр программы этого операторного выражения, после чего снова вернуться в оглавление операторов ЛОСа.

13. Найти в оглавлении операторов ЛОСа оператор, проверяющий отсутствие в терме повторных вхождений одной и той же переменной.

14. Найти в оглавлении операторов ЛОСа оператор, проверяющий, что один терм получается из другого вычеркиванием части операндов некоторой операции.

15. Найти в оглавлении операторов ЛОСа все операторы, обращающиеся к решению вспомогательной задачи на доказательство.

Указания

1. В главном меню нажать клавишу "п" либо нажать левую кнопку мыши в окне "Просмотр программы логического символа", затем набрать текстовым редактором слово "синус" и нажать "Enter".

2. Обычно операторы "уровень(...)"располагаются вдоль главного ствола ветви программы логического символа, содержащей приемы сканирования задачи. Эта ветвь располагается после оператора "решить". В корневом фрагменте видим операторы "уровень(0) иначе 2", так что для поиска операторов "уровень(1 3)", "уровень(2)" и "уровень(8)"переходим по указателю перехода "иначе 2".

Напомним, что клавиши "курсор влево - вправо" позволяют выбрать текущий указатель перехода, клавиша "курсор вниз" - переводит по этому указателю, "курсор вверх" - возвращает обратно. После перехода видим операторы "ветвь 1 уровень(1)". Наши операторы выбора уровня не могут размещаться после оператора "уровень(1)", так что перемещаемся по указателю "ветвь 1". Это перемещение приводит к фрагменту с оператором "уровень(1 3)" - один из искомых операторов. Продолжая далее перемещаться вдоль главного ствола (переходы до операторов "уровень" либо непосредственно после них), находим операторы "уровень(2)" и "уровень(8)". Заметим, что иногда сначала идет оператор "уровень($N_1 \ldots N_m$)", объединяющий несколько уровней, а после него размещается ветвь, в которой операторы "уровень" указывают различные сужения исходного списка $N_1, \ldots, N_m$.

3. Чтобы найти в некоторой ветви программы логического символа все вхождения заданного логического символа либо терма, следует сначала перейти в корень этой ветви, затем нажать клавишу F4 и набрать текстовым редактором искомый логический символ либо терм. После нажатия "Enter", завершающего набор, начинается поиск требуемых вхождений. Если вхождение найдено, то на экране изображается содержащий его фрагмент, в котором оно выделено многоцветной указкой. Каждое последующее нажатие F4 приводит к поиску очередного вхождения. По завершении поиска возникает пустой экран, и по нажатии любой клавиши далее восстанавливается изображение на момент начала поиска.

4. Для получения справочной информации о логическом символе, программа которого просматривается, достаточно нажать F3. Если нужно получить информацию о каком-либо другом символе, то нажимается F2 и набирается нужный символ.

5. Перевести курсор мыши на заголовок оператора "заменавхождения(...)" и нажать левую клавишу мыши, чтобы этот оператор выделить многоцветной указкой. Затем нажать правую клавишу мыши. Для перелистывания использовать либо "PageUp-PageDown", либо нажатия левой кнопкой мыши на стрелках в меню. Для возвращения в просмотр фрагмента программы нажать любую кнопку мыши внутри текста. Чтобы убрать мышью выделенный подтерм, нажать ее левую кнопку вне области операторов программы.

6. Выделить левой кнопкой мыши вхождение логического символа "заменавхождения" и нажать клавишу "c". Для возвращения в программу символа "синус" нажать "End".

7. Нажать клавишу "Ctr-л", переводящую в оглавление операторов ЛОСа. Для возвращения в просмотр фрагмента программы нажать "End".

8. Чтобы перейти в пункт оглавления программ ЛОСа, соответствующий указателю "прием(N)", найденному в некотором фрагменте программы, нужно выделить многоцветной указкой какой-либо оператор, расположенный после данного указателя, и нажать "Ctr-End". Возвращение в программу - нажатие "курсор вправо" на выбранном пункте оглавления. Данная операция работает только в случаях, когда просмотр программы был начат из главного меню.

9. Для перехода в корневой фрагмент программы следующего (по номеру - первого после текущего, для которого создана программа) логического символа достаточно нажать клавишу "ш".

10. Для входа в режим многоцветной указки (режим просмотра подтермов операторов) нажимается клавиша "о"(кир.). Далее клавишами "курсор влево - вправо" выбирается нужный оператор, клавишей "курсор вниз осуществляется вход в выделение его подоператоров, и т.д. вплоть до нужного подтерма. Этого же можно добиться, нажав левую кнопку мыши на заголовке выделяемого подтерма. При единичном выделении удобнее пользоваться мышью, при последовательном просмотре операторов и их подтермов удобнее пользоваться клавишами курсора.

11. Указатели типов обращений "решить", "программа", "обращение(A)" размещаются в виде линейной цепочки, начинающейся в корневом фрагменте программы. Начало корневого фрагмента программы символа "синус" имеет вид "решить иначе 1". Переходим по указетлю "иначе 1" - появляется фрагмент с началом "обращение(блокредактора) иначе 1". Снова переходим по указателю "иначе 1", и т.д., пока не найдем фрагмент с началом "обращение(тип)". Для возвращения в корневой фрагмент достаточно нажать "End". Последняя операция работает только в случае, если просмотр программы был начат из главного меню.

12. Для входа в оглавление операторов из главного меню нажимаем клавишу "о" (кир.). Находим в корневом меню оглавления раздел "Операторы и операторные выражения, реализованные на ЛОСе". Далее смотрим подраздел "Операторы для работы с наборами и вхождениями в наборы". Затем выбираем пункт "Вхождения в набор", и наконец - "Номер вхождения в набор". Остальные упражнения на поиск в данном оглавлении делаются аналогично.

8.3.2 Логические символы

1. Определить номер логического символа "синус".

2. Определить название логического символа с номером 1000.

3. Просмотреть названия логических символов с номерами от 290 до 294. Просмотреть справочную информацию об этих символах.

4. Найти логический символ, для которого имеется программа, но нет справочной информации.

5. Найти логический символ, для которого имеется справочная информация, но нет программы.

6. Найти номер логического символа, для которого не введено название.

7. Ввести новый логический символ с названием "рррр".

8. Изменить название логического символа "рррр" на "тттт".

9. Удалить логический символ "тттт".

Указания

1. Войти в пункт "Ресурсы и установки" главного меню, выбрать далее пункт "определение номера символа по его названию", нажав клавишу "c" либо используя левую кнопку мыши, и ввести текстовым редактором слово "синус". Затем нажать "Enter". Для возвращения в главное меню нажать любую клавишу.

2. Войти в пункт "Ресурсы и установки" и выбрать подпункт "Определение названия символа по его номеру". Затем набрать номер 1000 и нажать "Enter". Эту же операцию (и все остальные операции подменю "Ресурсы и установки", кроме описанной в предыдущем пункте) можно выполнять непосредственно из главного меню - начиная ее сразу с нажатия "Ctr - c").

3. Для просмотра списка упорядоченных по номерам логических символов войти из главного меню в пункт "Логические символы". Затем перелистывать страницы с помощью "PageDown - PageUp" до нахождения нужного диапазона номеров символов. Выделить нужный символ голубым цветом, используя клавиши "курсор вниз-вверх" в рамках одного столбца и "курсор влево - вправо" для смены столбца. Для просмотра информации о выделенном символе нажать "и". Для возвращения в просмотр символов нажать любую клавишу, не используемую при просмотре справочной информации.

4. Войти в просмотр списка упорядоченных по номерам логических символов и перелистывать страницы до обнаружения символа, после которого располагается знак "&" и нет знака "!".

5. Аналогично предыдущему, но искать символ, после которого идет "!", но нет "&".

6. Перелистывать список символов до номеров, после которых идут три знака "!", "?", "&".

7. Войти в пункт "Ресурсы и установки" главного меню. Выбрать подпункт "Ввод названия нового символа". Ввести текстовым редактором требуемое слово (если оно уже использовано в качестве названия, то об этом появляется сообщение) и нажать "Enter".

8. Войти в пункт "Ресурсы и установки" и выбрать подпункт "Изменение названия символа". Затем ввести изменяемое название (эту операцию требуется проводить осторожно, избегая вводить отсутствующее название) и нажать "Enter". После нажатия "Enter" старое название уже уничтожено; далее водится новое название. Если новое название уже было использовано, то будет выдано соответствующее сообщение, после чего по нажатии любой клавиши произойдет выход в главное меню. Чтобы в этом случае завершить операцию изменения названия, нужно дальше действовать так же, как при вводе нового логического символа - новое название будет сопоставлено первому свободному номеру символа, т.е. (в обычной ситуации) номеру, для которого только что было уничтожено старое название.

9. Войти в пункт "Ресурсы и установки" и выбрать подпункт "Удаление названия символа". Ввести название удаляемого символа (избегая вводить несуществующие названия) и нажать "Enter".

8.3.3 Редактирование программы

1. Ввести новый логический символ с названием "включсимв". Войти в просмотр программы этого символа. Войти в редактирование справочной информации для символа "включсимв" и набрать текст: "включсимв(x1 x2). x1,x2 - термы. Оператор проверяет, что каждый логический символ, встречающийся в терме x1, встречается также в терме x2". Затем вернуться в просмотр программы, войти в редактирование программы и набрать текст программы: "программа метка(икс(3)) равно(x3 перечисление(x4 и(входит(x4 x1) логсимвол(x4))x4)) равно(x4 перечисление(x5 и(входит(x5 x2)логсимвол(x5))x5)) включается(x3 x4) выход".

2. Разрезать программу символа "включсимв", введенную в предыдущем упражнении, на два последовательно идущих фрагмента, так, чтобы второй начинался с оператора "равно(x4 ...)".

3. Склеить два фрагмента, на которые была разрезана программа символа "включсимв", в один общий фрагмент.

4. Вставить в программу символа "включсимв"программу справочника "арность", состоящую из операторов "обращение(арность) ответ(2)". Затем удалить эту введенную вручную программу и осуществить ее автоматический синтез.

5. Увеличить на 2 номера всех программных переменных, начиная с переменной x4, в операторах программы символа "включсимв", идущих после оператора "равно(x3 ...)". Затем восстановить исходные номера этих переменных.

6. Войдя в редактирование программы символа "включсимв", выполнить следующие стандартные операции текстового редактора: а) переставить местами операторы "равно(x3 ...)" и "равно(x4 ...)", после чего восстановить исходное их размещение; б) Найти для каких-либо открывающих и закрывающих скобок операторов программы двойственные им скобки; в) Перед оператором "равно(x3 ...)" вставить оператор "равно(x3 0)", а затем удалить этот оператор; г) после оператора "равно(x3 ...)" разместить его копию, а затем удалить эту копию.

7. Войдя в редактирование программы символа "включсимв", просмотреть справочную информацию для логического символа "включается", после чего вернуться в редактирование.

8. Найти в программе логического символа "делит" ветвь фрагмента, начинающегося с оператора "обращение(программа)", и скопировать эту ветвь в программу логического символа "включсимв", после чего удалить добавленную ветвь.

9. Найти в программе логического символа "делит" фрагмент, начинающийся с оператора "обращение(выводсимвола)", и вставить копию этого фрагмента в программу оператора "включсимв" между операторами "равно(x3 ...)" и "равно(x4 ...)", отбросив идущий после "обращение(выводсимвола)" оператор "иначе 1". Затем удалить вставленные операторы.

10. Войти в редактирование программы "включсимв" и ввести несколько ошибок в программу - исказить название какого-либо оператора; добавить лишнюю

скобку либо удалить имевшуюся. Затем нажать "Enter" и исправить ошибку, на которую будет указано в верхней правой части экрана.

11. Войти в редактирование программы "включсимв" и внести в нее ошибку, изменив оператор "включается(x3 x4)" на "деление(x3 x4)". Затем выйти из редактирования и предпринять автоматическую проверку программы. После получения указания на ошибку исправить эту ошибку.

12. Аналогично предыдущему, но оператор "включается(x3 x4)" заменить на "включается(x3 x5)".

13. Удалить программу логического символа "включсимв", удалить справочную информацию для этого символа, после чего удалить сам символ.

Указания

1. Введя название нового символа, вернуться в главное меню и набрать название этого символа в окне "Просмотр программы логического символа" (набор завершается нажатием "Enter"). После появления пустого экрана нажать F3 и набрать требуемый текст справочной информации. После этого вернуться в просмотр программы (будет восстановлен пустой экран) и нажать клавишу "р". В правом верхнем углу экрана появится предупреждающий о режиме редактирования программы красный прямоугольник, а в левом верхнем углу - курсор текстового редактора. Далее выполняется набор текста программы, и по завершении набора нажимается "Enter". Если при наборе были допущены ошибки, то в правом верхнем углу появится сообщение об ошибке. При этом режим текстового редактора сохраняется, так что можно сразу же исправить ошибку и снова нажать "Enter". При отсутствии ошибок нажатие "Enter" завершает создание либо изменение фрагмента программы.

2. Для разрезания фрагмента программы на две части войти в режим просмотра подтермов операторов (клавиша "о", кир.), выделить тот оператор, с которого должна начинаться вторая половина разрезанного фрагмента, и нажать клавишу "р"(кир.).

3. Если фрагмент программы заканчивается операторами "ветвь N продолжение", то нажатие клавиши F6 приводит к тому, что текст фрагмента по ссылке "ветвь N" заносится в конец текущего фрагмента вместо двух его последних операторов, а сам этот фрагмент исключается.

4. Нужно войти в редактирование корневого фрагмента программы символа "включсимв" и вставить после оператора "программа" оператор "иначе 1". Затем нажать "Enter". Тогда появится пустой экран, в левом верхнем углу которого будет прорисована цифра 1 - номер ссылки к редактируемому подфрагменту, а под ней - курсор текстового редактора для набора текста этого подфрагмента. После набор текста нажимается "Enter", завершающее ввод ветви программы. Чтобы удалить ветвь программы, имеются два способа: а) в режиме обычного просмотра фрагмента программы выделить оператор перехода к удаляемой ветви и нажать "Ctr-Del"; б) войти в редактирование фрагмента программы и исключить оператор перехода к удаляемой ветви. Способ а) может быть применен лишь однократно; для повторного его применения необходимо сначала

выйти в главное меню через "Esc". Это объясняется тем, что удаленная ветвь временно сохраняется в буфере и может быть восстановлена (до указанного выхода в главное меню) в другом месте программы. Способ б) можно применять без каких-либо ограничений.

Для автоматического контроля программы и ввода подфрагментов, указывающих арность оператора либо операторного выражения, реализованного этой программой, нужно в корневом фрагменте программы нажать клавишу "п". Синтез программы справочника "арность" (для перечисляющего оператора - также справочника "комментпосылок") выполняется только при условии, что после начинающего корневой фрагмент оператора "программа" либо "обращение(0)" не идет оператор перехода "иначе".

5. Для увеличения либо уменьшения номеров программных переменных сначала следует войти в режим просмотра подтермов операторов, затем выделить тот оператор, начиная с которого должны быть изменены номера переменных, и нажать клавишу "п". Тогда в левом верхнем углу появляется курсор текстового редактора. Для увеличения на n номеров переменных, начиная с переменной хm (включая эту переменную), набирается текст "плюс(хm n)"; для уменьшения - "минус(хm n)". После нажатия "Enter" происходит коррекция номеров переменных в данном фрагменте и всех его подфрагментах, достижимых через выделенный оператор.

6. Напомним используемые в данном упражнении операции текстового редактора: а) Для перенесения фрагмента текста на новое место курсор подводится к началу этого фрагмента и нажимается F2; затем курсор подводится к концу фрагмента и снова нажимается F2 (начало и конец фрагмента выделены зелеными маркерами). Наконец, курсор подводится к той позиции, начиная с которой должен быть размещен фрагмент, и опять нажимается F2; б) Для перехода от скобки к двойственной ей курсор устанавливается на скобку и нажимается F1; в) Для вставки фрагмента текста курсор подводится к тому месту, с которого должна начинаться вставка, и нажимается "Insert" курсор становится голубым. Каждый набираемый с клавиатуры символ вставляется на позицию курсора, а курсор перемещается на одну клетку вправо. Выход из режима вставки - повторное нажатие "Insert" либо (для исправления ошибки) "Backspace"; г) Для копирования фрагмент текста курсор подводится к его началу и нажимается "PageDown". Затем курсор подводится к концу фрагмента и снова нажимается "PageDown" - фрагмент таким образом записан в буфере. Для извлечения его из буфера (что может происходить при повторных обращениях к текстовому редактору в произвольных интерфейсах системы) курсор подводится к тому месту, начиная с которого должна быть вставлена копия, и нажимается "PageUp".

   Для удаления фрагмента текста следует подвести курсор к началу фрагмента и нажать "Delete", затем подвести его к первой позиции после конца фрагмента и повторно нажать "Delete".

7. Для просмотра справочной информации о логических символах непосредственно из текстового редактора следует нажать клавишу F4 и набрать название нужного логического символа. После просмотра будет восстановлена текущая ситуация редактирования.

8. Сначала следует войти в программу символа "делит" и, спускаясь вдоль "главного ствола", найти фрагмент, начинающийся с оператора "обращение(программа)". Затем нажимается "Insert" - ссылка на ветвь этого фрагмента сохранена в буфере. После этого нужно выйти в главное меню по нажатию клавиши "PageUp"(при другом способе выхода ссылка на ветвь будет утеряна) и набрать в окне "Просмотр программы логического символа" название "включсимв". Внутри программы "включсимв" следует найти фрагмент оператора "обращение(арность)", войти в его редактирование и вставить после первого оператора переход "иначе 1". Войдя в редактирование подфрагмента по ссылке "иначе 1", набрать единственный оператор "продолжение". Наконец, после нажатия "Enter" вернуться в созданный подфрагмент "продолжение", выделить его оператор многоцветной указкой ("о", кир.) и нажать клавишу "к" для копирования выделенной ветви вместо ветви подфрагмента "продолжение".

9. Эта операция выполняется с помощью буфера текстового редактора: сначала в программе символа "делит" находим фрагмент оператора "обращение(вывод-символа)", входим в редактирование этого фрагмента и заносим (нажатиями "PageDown") его в буфер. Затем переходим в редактирование корневого фрагмента программы символа "включсимв", устанавливаем курсор перед оператором "равно(x4 ...)" и нажимаем "PageUp".

10. При неправильном наборе логического символа в правой верхней части экрана появляется текст "ошибка в логическом символе", причем курсор текстового редактора устанавливается на начало неправильно набранного символа. Если число закрывающих скобок больше числа открывающих, появляется текст "избыточная правая скобка" и курсор устанавливается на первую правую скобку, не имеющую двойственной левой. Если число открывающих скобок больше числа закрывающих, то появляется текст "незакрытая левая скобка" и курсор устанавливается в левый верхний угол экрана. В этом случае для поиска незакрытой скобки можно последовательно устанавливать курсор на все открывающие скобки, нажимая F1 для проверки наличия соответствующей закрывающей скобки.

11. Для автоматической проверки набранной программы нажимается клавиша "п". Если указать неправильное число операндов (у оператора "деление" оно равно 4), то в правом верхнем углу экрана появится текст "ошибка в числе операндов", причем ошибочный оператор будет выделен многоцветной указкой. Как обычно, для выхода из режима многоцветной указки нажимаем "о" и переходим в режим редактирования для исправления ошибки. Заметим, что такой контроль числа операндов распространяется только на операторы и операторные выражения, запрограммированные на ЛОСе. Если допущена ошибка в числе операндов для базисного оператора либо операторного выражения, то скорее всего она будет обнаружена немедленно, так как после нажатия "Enter" и перерисовки набранного текста программы произойдет искажение сразу нескольких операторов, соседних с тем, у которого число операндов не соответствует норме.

12. При автоматическом контроле правильности программы выявляются входные переменные, значения которых еще не определены, а также выходные переменные, значения которых уже были определены.

13. Для удаления программы логического символа следует войти в ее корневой фрагмент и нажать "Ctr-F7". Для удаления справочной информации о символе войти в ее просмотр (если она размещена на нескольких страницах, то найти последнюю станицу) и далее нажимать "Ctr-Del" столько раз, сколько понадобится для удаления всех ее страниц.

8.3.4 Запуск программы и ее отладочная трассировка

Тестирование процедур логической системы обычно предпринимается в реальном контексте их функционирования - в рамках приема либо блока интерфейса. Поэтому для проверки новой программы обращение к ней включается в соответствующие прием либо блок интерфейса, и далее либо запускается процесс решения задачи, в которой должен сработать рассматриваемый прием, либо выбирается режим интерфейса, обращающийся к новой программе.

Однако, для особых случаев предусмотрена также возможность запуска новой программы безотносительно к контексту предполагаемого ее применения. Для такого запуска нужно предварительно создать входные данные обращения к программе; это удобно делать в рамках некоторой другой программы. В качестве такой стандартной программы, обращающейся к тестируемой программе, выбрана программа логического символа "пуск". Ее первые два оператора "программа метка(икс(2))" следует оставлять без изменений, а все остальные операторы - изменять в соответствии с организацией тестирования. Запуск программы "пуск" осуществляется непосредственно из просмотра ее корневого фрагмента нажатием "Ctr-Enter"; после этого устанавливается режим пошаговой трассировки в рамках отладчика ЛОСа. По завершении программы "пуск" происходит возвращение в главное меню.

1. Предпринять пошаговое выполнение программы "включсимв(х1 х2)" при входных данных - термах "синус(плюс(х1 х2 х3))"и "степень(х2 плюс(х1 х3 синус (х2)))".

2. В процессе пошагового выполнения программы "включсимв(х1 х2)"при указанных выше входных данных посмотреть значения переменных х3 и х4.

3. В процессе пошагового выполнения программы "включсимв" посмотреть внешнюю обратившуюся к ней программу "пуск", а также внешнюю обратившуюся к программе "пуск" программу "вход" (она является программой общего интерфейса логической системы). Посмотреть значения программных переменных х2 и х3 для программы "пуск" в двух режимах - с обычной и скобочной прорисовкой формул. Посмотреть значение переменной х1 для программы "вход" (ее значением служит исходная задача).

4. Найти в задачнике раздел "Элементарная алгебра - Упрощение выражений - Разложение на множители-1" и войти в просмотр первой задачи этого раздела. Запустить решение этой задачи с прерыванием при обращении к оператору "заменавхождения". Определить значения программных переменных х1 (заменяемое вхождение) и х5 (заменяющий терм). Определить значения переменных х2, х3, х4 (вхождение текущего терма в список задачи; указатель условия либо посылки; текущая задача). Определить текущий уровень сканирования, при котором был применен прием, обратившийся к оператору "заменавхождения". Используя сквозной просмотр, рассмотреть текущую задачу х4.

5. В контексте предыдущего упражнения (т.е. в начале пошаговой трассировки внутри программы оператора "заменавхождения") предпринять пооператорную трассировку вплоть до обращения к оператору "сопровождение(x4 x1 x2 x3)". Войти в трассировку внутри последнего оператора. Затем выйти в главное меню; повторно войти в контекст предыдущего упраженения, установить прерывание при обращении к программе "сопровождение"и запустить программу до данного прерывания.

6. Войти в трассировку программы "заменавхождения" при решении указанной выше задачи через оглавление программ ЛОСа. После этого выбрать пункт 16 в подразделе оглавления программ, перечисляющем действия оператора "заменавхождения", и выйти в трассировку при достижении программой этого пункта. Проделать ту же операцию непосредственно из запуска задачи, без прерывания в начале выполнения программы "заменавхождения".

7. Запустить решение указанной выше задачи с прерыванием на 14-м пункте подраздела оглавления программ для оператора "заменавхождения". Далее, используя многоцветную указку, установить прерывание при обращении к оператору "равно(x14 справка(одз ...))", и запустить выполнение программы до этого прерывания.

8. Войти в прерывание при обращении к оператору "заменавхождения" в указанной выше задаче и определить результат обращения к операторному выражению "видумножение(...)", находящемуся в программе приема, обратившегося к "заменавхождения".

9. Войти в прерывание при обращении к оператору "заменавхождения" в указанной выше задаче, определить число на счетчике шагов и перезапустить решение задачи с прерыванием по достижении этого числа шагов.

10. Войти в оглавление приемов через главное меню, выбрать какой-либо концевой пункт этого оглавления (после номера концевого пункта идет не скобка, а точка), и нажать "End". Войти через главное меню в корневой фрагмент программы логического символа "оглавление" и вставить перед оператором "повторение" в этом фрагменте оператор "трассировка(стоп 0)". Затем вернуться в главное меню и войти в оглавление базы приемов. После того, как произойдет выход в отладчик ЛОСа по установленной контрольной точке (по оператору "трассировка(стоп 0)"), пошаговой трассировкой дойти до выполнения оператора "файл(терм x10 x12)" и посмотреть значение переменной x12. Затем вернуться в главное меню и убрать из программы "оглавление" контрольную точку.

11. Выбрать в задачнике произвольную задачу на разложение на множители, установить прерывание при обращении к логическому символу "заменавхождения" и запустить решение до этого прерывания. Затем посмотреть текущий кадр семантической трассировки, сопровождающий данное обращение.

12. Для той же задачи запустить ее решение с семантической трассировкой и при первом срабатывании приема посмотреть текущий фрагмент программы, реализующей этот прием. Найти в этой программе оператор "заменавхождения"; посмотреть текущую точку программы "заменавхождения" на момент прерывания.

13. Найти в задачнике раздел "Элементарная алгебра - Решение уравнений - Логарифмические уравнения - Системы уравнений" и войти в просмотр задачи номер 10. Войти в семантическую трассировку решения этой задачи и продолжать ее до появления кадра "Переходим к основанию логарифмов a". Перейти в отладчик ЛОСа и посмотреть список комментариев к текущей задаче. Найти в нем комментарий "нормлогарифм a", определивший новое основание логарифма. Найти комментарий "сопровождение A" и установить прерывание до момента изменения этого комментария, после чего продолжить решение до данного прерывания.

Указания

1. Упражнение выполняется после того, как ранее была создана программа символа "включсимв". Нужно войти в программу символа "пуск" и набрать корневой фрагмент: "программа метка(икс(2)) равно(х2 запись(синус запись(плюс икс(1) икс(2) икс(3)))) равно(х3 запись(степень икс(2) запись(плюс икс(1) икс(3) запись(синус икс(2))))) включсимв(х2 х3) выход". В этом фрагменте сначала вводятся входные данные х2, х3 для обращения к оператору "включсимв", затем реализуется такое обращение. Далее нажать "Ctr-Enter" - начнется пошаговая трассировка только что набранной программы "пуск" отладчиком ЛОСа. Здесь удобно сразу нажать клавишу "2" для установки пооператорного режима вместо слишком подробного пошагового, и тремя нажатиями "Enter" дойти до обращения к оператору "включсимв". Чтобы войти в трассировку внутри этого оператора, следует снова установить пошаговый режим (нажатием клавиши "1") и нажать "Enter". Далее опять устанавливаем пооператорный режим и последовательно проходим операторы программы "включсимв"до оператора "выход", после которого автоматически происходит возвращение в главное меню.

2. Действия аналогичны описанным выше, но после определения значений переменных х3, х4 в программе "включсимв" предпринимаем просмотр значений этих переменных. Для просмотра сначала нажимаем клавишу "х" (кир.), и далее набираем номер переменной. После нажатия "Enter" справа от набранной переменной появляется ее значение - в данном случае это будут наборы логических символов.

3. Чтобы посмотреть фрагмент программы, из которого произошло обращение к текущему фрагменту программы, нажимаем "PageUp". Делаем это дважды - сначала возвращаемся к программе "пуск", затем - к программе "вход", обеспечивающей интерфейс логической системы. Обратные переходы - с помощью "PageDown". Находясь в рамках программы "вход", можно с помощью клавиш "Home - End" пройтись по всей текущей цепочке ее фрагментов - от текущего фрагмента до корневого. Чтобы перейти к режиму просмотра термов х2,х3 в скобочной записи, нажимается клавиша "с"; для возвращения в стандартную математическую запись - нажимается "м" (кир.). При просмотре значения х1 в программе "вход" появится строчка "исследовать н1 т2 н3 т4". Записи "н1","н3" обозначают некоторые наборы (в данном случае - списки посылок и комментариев к посылкам); записи "т2", "т4" - некоторые термы либо наборы из единственного символа, отождествляемые с термами. Чтобы посмотреть, из

чего состоят наборы "нi", нажимаем клавишу "н" и набираем i; чтобы посмотреть термы "тi", нажимаем клавишу "т"и набираем i. Эту операцию можно повторять на любую глубину (кроме термов и наборов, могут встречаться ссылки на вхождения "вi", для которых принцип просмотра тот же). Если экран заполнен просмотренными значениями и нужно его расчистить, нажимаем "Delete" (если этого не делать, то при переполнении экрана начинается прокрутка выписанных значений вверх, не затрагивающая области фрагмента программы). После нажатия "Delete" можно восстановить текст фрагмента прогаммы, нажав клавишу "ф".

4. Для входа в задачник из главного меню нажать клавишу "з"; используя клавиши курсора, найти нужный раздел и в нем - пункт с номером требуемой задачи. Затем войти в просмотр этой задачи (курсор вправо). Для установки прерывания при обращении к программе логического символа "заменавхождения" нажать клавишу "Л" и набрать название данного символа. После нажатия "Enter" инициируется решение задачи с выходом в отладчик ЛОСа при первом же обращении к оператору "заменавхождения". Для просмотра значения переменной x1 нажимаем клавиши "х" (кир.) и "1", "Enter". Голубой цвет всего прорисованного после этого терма либо его части означает, что значением переменной является не сам терм, а выделенное голубым вхождение в него. Точно так же просматриваются значения переменных x2,...,x5. Чтобы определить текущий уровень сканирования, при котором был применен прием, обратившийся к оператору "заменавхождения", нажимаем клавишу "PageUp", переводящую в просмотр программы, обратившейся к программе "заменавхождения". После этого достаточно посмотреть значение переменной x5 (напомним, что в программе приема, используемого при сканировании задачи, значения первых 5 переменных определены уже при обращении к ней, и значением x5 при этом служит текущий уровень сканирования; x1 - текущая задача; (x2,x3,x4) - координата текущего вхождения в задачу). В данном случае текст программы приема уже достаточно велик; если бы нам было нужно найти в нем заданный оператор (например, "заменавхождения"), то помогла бы многоцветная указка, для перехода к которой достаточно нажать "курсор вниз" либо нажать левую клавишу мыши в нужной точке программы. Выход из многоцветной указки - нажатие левой клавиши мыши вне области программы либо нажатие (в зависимости от глубины просматриваемого подтерма - одно или несколько) клавиши "курсор вверх". Для входа в сквозной просмотр задачи x4 нужно вернуться (PageDown) в программу "заменавхождения", нажать клавишу "К" (кир.) и номер переменной, затем - "Enter". Тогда появляется список элементов просматриваемого набора; если после номера элемента стоит круглая скобка, то этот элемент - атомарный, и просмотр его элементов невозможен. Если после номера стоит точка, то элемент является набором, и можно посмотреть его элементы, используя клавишу "курсор вправо". Для возвращения к предыдущему уровню просмотра служит "курсор влево".

5. Для перехода от пошаговой трассировки к пооператорной нажать клавишу "2". Затем нажимать "Enter" до появления выделенного малиновым цветом оператора "сопровождение(x4 x1 x2 x3)". Чтобы войти в трассировку внутри этого оператора, вернуться к пошаговой трассировке (клавиша "1") и нажать "Enter". Для обрыва трассировки и выхода в главное меню нажимается "Esc". Повторно войдя в начало выполнения оператора "заменавхождения", нажать клавишу

"л" и набрать название оператора "сопровождение". Первым нажатием "Enter" завершить набор, вторым нажатием "Enter" - запустить программу до прерывания при входе в оператор "сопровождение".

6. Войти в просмотр задачи и нажать клавишу "л", переводящую в оглавление программ ЛОСа. В корневом меню этого оглавления выбрать пункт "Приемы решателя", далее - подпункт "Общие процедуры, используемые в приемах", далее - подпункт "Процедура ЗАМЕНАВХОЖДЕНИЯ". После этого нажать клавишу "курсор вправо". Автоматически запускается решение задачи, и при обращении к оператору "заменавхождение" организуется выход в отладчик ЛОСа с установкой пошагового режима трассировки. Чтобы теперь вернуться в оглавление программ ЛОСа, нажимается "Ctr-г". Далее выбирается пункт 16 и нажимается "курсор вправо" - снова организуется прерывание при достижении контрольной точки, соответствующей выбранному пункту (в нашем случае этой контрольной точкой является оператор "прием(1 0)"). При повторе прерывание по достижении пункта 16 устанавливается из первого обращения к оглавлению программ.

7. Войти в прерывание по достижении 14 - го пункта списка контрольных точек оператора "заменавхождения". Затем нажать "курсор вниз" - появляется многоцветная указка, выделяющая первый оператор программы. Используя "курсор вправо", дойти до выделения оператора "равно(x14 справка(одз ...))" и нажать "Enter". После нажатия запускается решение задачи до прерывания по достижении выделенного оператора.

8. Войти в прерывание при обращении к программе "заменавхождения" и нажать "PageUp" для просмотра внешней программы приема. Найти в ней (например, с помощью многоцветной указки) оператор вида "равно(xi видумножение(...))" - в нашем случае $i = 11$. Затем посмотреть значение переменной x11.

9. Войти в прерывание при обращении к программе "заменавхождения". В правом верхнем углу экрана имеется текст "шаг № N", где N - номер текущего шага интерпретатора ЛОСа, отсчитываемый с момента запуска задачи. Запомнить этот номер, нажать "Esc" для обрыва трассировки. Затем вернуться в просмотр задачи и нажать клавишу "Ш", после чего набрать номер N и нажать "Enter". После нажатия "Enter" устанавливается прерывание по достижении шага с нужным номером и запускается решение задачи. Заметим, что выход в отладчик здесь может произойти не в программе "заменавхождения", а непосредственно перед обращением к ней.

10. Подготовить к последующей трассировке оглавление базы приемов, как указано в упражнении (выбрать его концевой пункт и нажать "End"). Без такой подготовки выполнение всех требований упражнения будет невозможным, так как выполнение программы будет происходить без выхода в нужную точку. После входа в просмотр корневого фрагмента программы "оглавление" нажать клавишу "р" и поместить курсор перед оператором "повторение". Нажать "Insert" и набрать оператор "трассировка(стоп 0)". Затем снова нажать "Insert" (выход из режима вставки) и нажать "Enter". Эти действия следует выполнять чрезвычайно осторожно, так как ошибка в редактировании и нажатие "Enter" могут привести к порче программы "оглавление" (например, к удалению каких-либо

ее ветвей), после чего нужно будет восстанавливать программу логической системы из резервной директории "SLCOPY". По завершении редактирования нужно вернуться в главное меню (например, через "Esc") и нажать клавишу "г". Это приведет к входу в программу "оглавление" и немедленной остановке по достижении введенной контрольной точки "трассировка(стоп 0)". Здесь войти в пооператорную трасировку (клавиша "2") и, нажимая "Enter", дойти до выполнения оператора "файл(терм x10 x12)". Длительность трассировки будет зависеть от того, сколь длинным является путь до концевого пункта оглавления базы приемов от корневого меню. Посмотреть значение переменной x12 лучше с помощью процедуры сквозного просмотра (К12; К - кир.). Значением этой переменной служит содержимое логического терминала текущего концевого пункта оглавления базы приемов. Выйдя из трассировки ("Esc"), вернуться в редактирование программы символа "оглавление" и удалить оператор "трассировка(стоп 0)". Это можно делать либо с помощью "Delete" (сначала нажатие "Delete" на первой букве слова "трассировка", затем - на букве, расположенной после удаляемой закрывающей скобки), либо с помощью клавиши "пробел", используемой как ластик. Данную операцию следует выполнять исключительно осторожно, чтобы не удалить других операторов (в особенности операторов перехода). Если, все же, были искажены или удалены какие - то нужные операторы, следует нажать "Esc" для отмены редактирования, затем войти в редактирование повторно. Удаление контрольной точки "трассировка(стоп 0)" из программы лучше выполнить не откладывая, так как эта контрольная точка блокирует использование каких-либо оглавлений логической системы.

11. Для выхода из технической трассировки в семантическую нажать клавишу пробела и затем "Enter". Можно также, не выходя из режима технической трассировки, посмотреть текущую цепь задач на уровне обычной математической записи, нажав клавишу "з" и используя далее обычные возможности прокрутки (курсоры вверх-вниз; PageUp; PageDown; Ctr-курсоры вверх-вниз). Для возвращения в просмотр ситуации через отладчик ЛОСа нажимается "ф".

12. Войдя в просмотр задачи, нажать клавишу "р" (кир.). После появления на экране сообщения о преобразовании задачи нажать клавишу "ф". Оператор "заменавхождения" расположен в конце программы приема. Чтоюы посмотреть текущую точку внутри программы этого оператора (не завершенной на момент прерывания), нажать "PageDown".

13. Войдя в просмотр указанной задачи, нажать "р", и далее нажимать "Enter" до появления требуемого кадра. Затем нажать "ф". Для просмотра списка комментариев к текущей задаче x1 можно либо сразу войти в сквозной просмотр этой задачи (нажатием "К","1","Enter"), либо сначала посмотреть значение переменной x1 (нажатием "х", "1","Enter") и затем войти в сквозной просмотр набора комментариев н6 (нажатием "к", "6", "Enter"). В наборе комментариев сначала идут комментарии к отдельным условиям, а в конце расположен список общих комментариев к задаче. Найти этот список (выделить его при помощи "курсор вниз"), и нажать "курсор вправо". Далее найти комментарий "нормлогарифм т"; выделить его и снова нажать "курсор вправо" для просмотра терма "$_{\text{т}}$".

Чтобы устанавливать прерывание на момент изменения разряда набора (только с помощью оператора "изменение(...)"; изменения с помощью специальных опера-

торов списков задачи - посылок, условий, комментариев, весов, - этой установкой не контролируются), нужно сначала вызвать этот набор на экран, чтобы он был обозначен посредством "нi". В нашем случае для этого сначала вызываем для просмотра x1; затем - н6 (список комментариев); затем - н12 (последний элемент списка комментариев - список общих комментариев к задаче); затем - н13 (для проверки того, что это и есть искомый комментарий "сопровождение A"). Далее нужно набрать "и", "13", "Enter". После этого установка прерывания осуществлена, и для запуска решения нажимается "Enter". Заметим, что в случае, когда изменяемый набор является значением программной переменной xi, можно установить аналогичное прерывание нажатием "И", "i", "Enter".

8.3.5 Создание новой программы

Приведем несколько упражнений на создание ЛОС-программ. Упражнения сопровождаются подробными указаниями. Рекомендуется прочитать указания к первому упражнению, а затем пытаться самостоятельно выполнить остальные.

1. Научить решатель приводить формулу алгебры логики к виду дизъюнктивной нормальной формы.
2. Научить решатель приводить формулу алгебры логики к виду конъюнктивной нормальной формы.
3. Научить решатель приводить формулу алгебры логики к виду полинома Жегалкина.
4. Научить решатель приводить заданную дизъюнктивную нормальную форму к виду сокращенной д.н.ф.
5. Научить решатель проверять, поглощается ли заданная элементарная конъюнкция заданной д.н.ф.
6. Научить решатель решать систему булевых уравнений.
7. Научить решатель транспонировать матрицу.

Указания

1. Приведение к виду дизъюнктивной нормальной формы.
   (a) Прежде всего, в задачнике решателя нужно создать какой-нибудь пример. На нем будет происходить проверка создаваемых приемов. Кроме того, пример будет подсказывать, каких приемов не хватает. Для создания примера выполняем следующие действия. Из главного меню выбираем пункт "Оглавление задачника". В корневом меню оглавления задачника выбираем пункт "Дискретная математика", в нем - подпункт "Алгебра логики". Здесь вводим новое подменю "Мои задачи": нажимаем "м" и затем текстовым редактором вводим название "Мои задачи". Нажимаем "курсор вправо" и далее нажатием клавиши "к" создаем новый концевой пункт. Автоматически, после создания данного пункта, система перейдет в него. Экран расчистится, а в верхней его части будет прорисована горизонтальная линия - отделитель поля новой задачи от предыдущих (их пока нет) задач.

(b) Для тестирования приведения формулы алгебры логики к виду д.н.ф. выбираем какую-то формулу, чтобы в ней присутствовали всевозможные операции. Пусть это будет формула "$(a + b)(c \vee \neg(a \leftrightarrow d)) + (b \rightarrow d)$".

(c) Вводим целевую установку задачи. Вообще-то, решатель уже и так умеет приводить формулы алгебры логики к виду д.н.ф. Однако, нам нужно создать свои приемы. Чтобы старые приемы начали работать для приведения к д.н.ф., задача должна иметь соответствующую целевую установку. Если изменить ее, они срабатывать не будут, и поле действия для новых приемов будет расчищено. В целях обучения была создана специальная цель "тест" задач на преобразование. Ее можно рекомендовать как универсальную целевую установку для различных учебных задач. Никакие старые приемы на эту цель не реагируют. В нашем примере нажимаем "ц" и попадаем в оглавление целевых установок. Из корня этого оглавления выбираем пункт "Преобразовать выражение", и далее - "Преобразовать к заданному виду". В результате создается заготовка задачи на преобразование, имеющей цель "тест". Никаких посылок задачи в нашем примере вводить не нужно, так как указания на двоичные типы значений переменных, как относящиеся к о.д.з., система занесет в список посылок автоматически. Если бы нужно было заблаговременно занести в посылки какие-то особые утверждения, их нужно было бы набрать до выбора целевой установки.

(d) Вводим выбранную нами формулу для приведения к виду д.н.ф. Ввод каждого нового утверждения - посылки либо задачи - начинается и заканчивается нажатием Enter. Упражнения на ввод формул формульным редактором уже приводились выше. Напомним лишь, что сумма по модулю 2 вводится не как "+", а через последовательное нажатие клавиш "пробел",s,b. Второй символ - b - сокращение от слова boolean - сопровождает все операции алгебры логики: дизъюнкция - "пробел",d,b; конъюнкция - "пробел",k,b; эквивалентность - "пробел",e,b; импликация - "пробел",i,b. Отрицание вводится без пробела - n,b.

(e) После того, как пример создан, нажимаем "о" и убеждаемся, что решатель выдает исходное выражение, никак его не изменяя. Это означает, что ничто не мешает создавать "свои" приемы.

(f) Прежде всего, вспоминаем алгебру логики. Для преобразования формулы к виду д.н.ф. сначала избавляемся от операций "$\leftrightarrow$", "$+$", "$\rightarrow$", выражая их через дизъюнкцию, конъюнкцию и отрицание: $a \leftrightarrow b = (a{\cdot}b \vee \neg(a){\cdot}\neg(b))$; $a + b = (a \cdot \neg(b) \ \vee \ \neg(a) \cdot b)$; $a \rightarrow b = \neg(a) \ \vee b$.

Затем нужно опустить отрицания до переменных: $\neg(a \vee b) = \neg(a) \cdot \neg(b)$; $\neg(a \cdot b) = \neg(a) \ \vee \ \neg(b)$; $\neg(\neg(a)) = a$.

Наконец, следует раскрыть скобки по дистрибутивности: $a \cdot (b \vee c) = (a \cdot b) \ \vee \ (a \cdot c)$.

Далее нужно будет упрощать д.н.ф. Однако, все используемые для этого "очевидные" переходы, например, $a \vee a = a, a \cdot a = a$ и т.п. реализуются решателем самостоятельно. Поэтому дальнейшие приемы не понадобятся.

(g) Чтобы контролировать местоположение новых приемов, через главное меню нажатием "л" входим в оглавление программ. В корневом меню этого

оглавления выбираем пункт "Приемы решателя". Войдя в него, нажимаем "м" и создаем пункт "Мои приемы".

(h) Переходим к программированию. Начнем с простейшего приема, выражающего импликацию через дизъюнкцию и отрицание: $a \to b = \neg(a) \ \vee b$.

Прежде всего, следует определить тот логический символ, при усмотрении которого в задаче будет предприниматься попытка применения приема. Такой символ называется символом привязки. Чтобы уменьшить число бесплодных попыток, целесообразно выбирать наиболее редко встречающийся символ. В нашем примере все однозначно. Символом привязки служит символ булевой импликации "имп". Не следует путать его с символом "если-то" кванторных импликаций. Там возникает логическая связка, а здесь - функция алгебры логики.

Следующий шаг - определение уровня срабатывания приема. Обычно он выбирается из общих соображений, а потом корректируется после примерки на задачах. В нашем примере ничто не мешает выбрать уровень срабатывания маленьким, например, равным 1.

Теперь нужно зайти в программу символа "имп". Для этого в главном меню нажимаем "п" и вводим слово "имп" текстовым редактором. После нажатия Enter появляется корневой фрагмент программы символа "имп". Будем выбирать ту точку программы, в которой следует создать ответвление к новому приему. Видим операторы "решить", "стандарт(х2)", "уровень(0) иначе 2". Так как наш уровень равен 1, выполняем переход через "иначе 2". Для этого клавишей "курсор вправо" выделяем желтым цветом слова "иначе 2", и далее нажимаем "курсор вниз". Появляется фрагмент с операторами "уровень(1) иначе 1", "ветвь 2", равно(х4 1)", "тип(х1 описать)". В нашем примере будет решаться задача на преобразование. Поэтому после "тип(х1 описать)" нужно вставить переход через "иначе" к тому фрагменту, с которого и будет начинаться новая программа.

Для вставки перехода нажимаем "р". В правом верхнему углу экрана появляется красная полоса, указывающая, что имеет место режим редактирования программы. Подводим курсор к позиции после оператора "тип(х1 описать)". По умолчанию, режим вставки текста с текущей позиции отсутствует. Поэтому, для освобождения места, нажимаем клавишу F7. Все, что располагалось после курсора, сдвигается на одну строку вниз. Выбираем номер нового перехода. В общем, здесь годится любое число, пока не использованнное в качестве номера перехода из текущего фрагмента. После редактирования нумерация переходов все равно будет восстановлена сплошная и начинающаяся с единицы. Можно взять номер "с запасом". Пусть, например, это будет 9. Итак, вводим текст "иначе 9" и нажимаем Enter. Текста фрагмента перерисовывается без пробелов, а после оператора "тип(х1 описать)" обнаруживаем оператор "иначе 3". Используя клавиши курсора, выделяем этот оператор желтым и переходим через него.

Появляется новый фрагмент, состоящий из единственного оператора "продолжение". Нажимаем "р" и переходим к набору программы. Оператор "продолжение" нам будет не нужен. Можно удалить его, но проще выполнять набор сразу поверх него. Итак, что мы имеем на текущий момент,

согласно уже пройденным операторам предыдущих фрагментов. Текущий логический символ - "имп", причем x2 - вхождение его в условие задачи, x3 - вхождение этого условия в некоторый набор, x4 - указатель на условие, равный 1. Начнем со ввода операторов, уточняющих тип и цели задачи: "тип(x1 преобразовать)", "цель(x1 тест)". После этого остается лишь сформировать заменяющее выражение. Введем для него вспомогательную переменную. Так как переменные с x1 по x5 уже определены, берем переменную x6 и вводим оператор "равно(x6 запись(дн запись(отр первыйтерм(x2)) второйтерм(x2)))". Теперь нужно реализовать замену в задаче. Для таких замен создан оператор "заменавхождения($X_1X_2X_3X_4X_5X_6$)". Здесь (X_1, X_2, X_3) - координата вхождения в задачу заменяемого терма, т.е. X_1 - вхождение заменяемого терма в некоторый терм задачи, X_2 - вхождение этого терма задачи в задачу либо список посылок либо список условий, X_3 - указатель посылки (0) либо условия (1). Значением X_4 служит задача, значением X_5 - заменяющий терм, значением X_6 - список опций замены. В нашем примере вводим оператор "заменавхождения(x2 x3 x4 x1 x6 пустоеслово)". Замена осуществлена, однако после нее обязательно должен идти завершающий программу оператор "пересмотр". Он осуществит уменьшение уровня сканирования до 0 и организует откат к повторному сканированию задачи. Нажимаем Enter, завершающее создание нового фрагмента.

Теперь нужно зарегистрировать прием в разделе "Мои приемы" оглавления программ. Находясь в просмотре только что созданного фрагмента, нажимаем "о" и переходим в режим выделения операторов фрагмента. Выбираем тот оператор, перед которым нужно будет вставить ссылку из оглавления программ. Например, оператор "равно(x6 ...)". После этого нажимаем "Page Dn", которое переведет нас в оглавление программ. Скорре всего, это будет пустой экран для ранее созданного раздела "Мои приемы". Так или иначе, переходим в указанный раздел, и для регистрации в нем ссылки на новый прием нажатием клавиши "к" создаем новый концевой пункт, называем его "Прием "имп, и нажимаем Enter. После этого ссылка создана. Нажатие клавиши "курсор влево" будет переводить из оглавления программ в просмотр нового фрагмента, причем перед оператором "равно(x6 ...)" обнаружится ссылка "прием(4)". Заметим, что при трассировке решения задачи "по шагам", если сработает наш прием, на экране в качестве пояснения появится текст "Прием "имп, извлеченный из оглавления программ.

(i) Проверяем, как сработал новый прием. Для этого запускаем решение задачи по шагам клавишей "р". Сразу же будет прорисована замена согласно нашему приему, а под ней, в скобках, появится поясняющий текст. Нажатием "ф" здесь можно перейти в отладчик ЛОСа, где будет виден текущий фрагмент программы и можно будет посмотреть значения программынх переменных. Для возвращения в трассировку по шагам решения задачи нажимаем "р".

(j) По аналогии с приемом для импликации, создаем приемы для суммы по модулю 2 и эквивалентности. Предоставляем сделать это самостоятельно.

(k) Переходим к приемам, опускающим отрицания до переменных. Например, $\neg(a \vee b) = \neg a \cdot \neg b$. Можно было бы поступить так же, как и выше, однако

двуместная операция $\vee$ ассоциативна и коммутативна. В формуле она может иметь произвольное число операндов, большее одного. Прием будем делать с учетом этого обстоятельства, иначе он будет срабатывать только в двуместных случаях.

Так как дизъюнкция, предположительно, будет встречаться в задачах реже отрицания, в качестве точки привязки выбираем символ "дн". Не следует путать его с символом "или". В первом случае имеем функцию алгебры логики, во втором - логическую связку. Опуская уже описанную выше процедуру создания "пустого" фрагмента программы для нашего приема, переходим к перечислению операторов этой программы. Вначале идут операторы "тип(х1 преобразовать)", "цель(х1 тест)". Текущий логический символ - "дн". Нужно выйти на внешнюю операцию и проверить, что эта операция - булево отрицание "отр". Для этого вводим операторы "операнд(х6 х2)", "символ(х6 отр)". Будем заменять отрицание дизъюнкции на конъюнкцию отрицаний всех ее операндов. Список этих отрицаний дается выражением "выписка(х7 операнд(х2 х7) запись(отр подтерм(х7)))". Соответственно, заменяющее выражение имеет вид "сборка(кн выписка(х7 операнд(х2 х7) запись(отр подтерм(х7))))".

Наконец, добавляем оператор "заменавхождения(х6 х3 х4 х1 сборка(кн выписка(х7 операнд(х2 х7) запись(отр подтерм(х7)))) пустоеслово)" и оператор "пересмотр". Разумеется, можно было бы сначала присвоить заменяющее выражение какой-либо вспомогательной переменной. Однако, она используется лишь однократно, и ее вводить не обязательно.

Аналогично отрицанию дизъюнкции устроен прием для отрицания конъюнкции. Рекомендуется создать его самостоятельно. Что касается приема для двойного отрицания, то он осуществляет безусловную стандартизацию выражений, и решатель применит его самостоятельно, безотносительно к целевой установке задачи.

(l) Остается лишь прием для "раскрывания скобок" по дистрибутивности. В качестве точки привязки снова выбираем символ "дн". После уточнения типа задачи и ее цели помещаем следующие операторы: "операнд(х6 х2)", "символ(х6 кн)", "равно(х7 исключениеоперанда(х6 х2)", "заменавхождения(х6 х3 х4 х1 сборка(дн выписка(х8 операнд(х2 х8)запись(кн х7 подтерм(х8)))))", "пересмотр". Здесь для "коэффициента" при дизъюнкции введена вспомогательная переменная х7, которая далее используется в цикле умножений на операнды дизъюнкции. Конечно, данный прием не очень экономичен. На самом деле решатель сначала приводит к виду д.н.ф. каждое произведение "запись(кн х7 подтерм(х8))", упрощает его, и лишь затем соединяет все общей дизъюнкцией. Для этого создан оператор "станддн", имеющий рекурсивные обращения к самому себе. Кроме того, вместо выражения "запись(кн х7 подтерм(х8))" лучше использовать выражение "соединение(кн х7 подтерм(х8))". Оно не только создает конъюнкцию, но и устраняет вложенные конъюнкции, появляющиеся, если хотя бы один из операндов сам был конъюнкцией.

(m) После завершения создания приемов, приводящих к виду д.н.ф., можно проверить, как они работают на различных примерах. Чтобы выполнять

следующие упражнения, далее эти приемы придется удалить. Для этого можно предложить два способа:

i. Выйти в главное меню и нажать $Ctr - Backspace$. Тогда автоматически из директории COPY будет извлечена резервная версия системы и переустановлена в качестве основной. Разумеется, все изменения пропадут. Чтобы пользоваться этой возможностью и не потерять ничего ценного, следует периодически из главного меню активировать пункты "Уплотнение измененных файлов" и "Сохранение копий файлов". Кстати, это позволит контролировать различные нарушения целостности программы. Если они появляются, при активации указанных пунктов решатель обрывает свою работу и выходит в операционную систему. Чтобы продолжить им пользоваться, придется извлечь резервную копию.

ii. Воспользоваться регистрацией новых приемов в разделе "Мои приемы" оглавления программ. Через пункты этого раздела можно переходить к тем операторам "ветвь" либо "иначе", которые представляют собой ответвления из "старого" фрагмента программы к новым фрагментам. Войдя в режим редактирования старого фрагмента, следует удалить такой оператор ответвления. Либо, выделив его желтым цветом, до перехода в режим редактирования, нажать $Ctr - Del$. Оба эти действия требуют осторожности, так как по ошибке можно удалить какую-либо старую ветвь программы, иногда очень большую. После удаления новых фрагментов программы следует вручную удалить соответствующие пункты из раздела "Мои приемы".

2. Программы приемов для приведения к конъюнктивной нормальной форме создайте самостоятельно.

3. Для приведения к полиному Жегалкина используем тождества, выражающие основные элементарные функции алгебры логики через сумму по модулю 2, конъюнкцию и константы 0,1. Именно:$\neg a = a+1$, $a \vee b = \neg(\neg a \cdot \neg b) = (a+1)(b+1)+1 = ab+a+b$; $a \to b = \neg a \vee b = (a+1)b+(a+1)+b = ab+a+1$; $a \leftrightarrow b = a+b+1$. Понадобится также раскрывание скобок: $a \cdot (b+c) = ab+ac$. Приведение подобных членов будет выполняться решателем без дополнительных приемов.

4. Для приведения заданной дизъюнктивной нормальной формы к виду сокращенной д.н.ф. используются преобразования обобщенного склеивания:

   $ab \vee \neg(a)c = ab \vee \neg(a)c \vee bc$

   и поглощения:

   $a \vee ab = a$

   Первое из них лишь усложняет выражение, однако необходимо для получения новых "минимальных" элементарных конъюнкций. При его применении, чтобы процесс оказался обрывающимся за конечное число шагов, следует проверять два условия:

   (a) Произведение bc не должно иметь противоположных сомножителей.

   (b) Никакая другая элементарная конъюнкция преобразуемой д.н.ф. не должна оказаться частью произведения bc.

Что касается преобразования поглощения, то оно уже реализовано в решателе как прием общей стандартизации и будет применяться сразу, без каких-либо ограничений. Этот прием можно не создавать.

В данном примере преобразование затрагивает лишь два каких-то члена корневой дизъюнкции. Поэтому точку привязки придется выбирать в одном из них. Удобно сделать это так, чтобы символом привязки стало булево отрицание "отр". Тогда сразу же находится переменная a, и вторую элементарную конъюнкцию можно будет искать по наличию в ней множителя a. Итак, начинаем составлять программу: "тип(х1 преобразовать)", "цель(х1 тест)", "первыйсимвол(х2 х6)". Теперь х6 - переменная a. Далее нужно определить элементарную конъюнкцию, к которой относится $\neg a$.

Рассматриваем вхождение внешней операции: "операнд(х7 х2)". Если х7 - вхождение символа "кн", то имеем несколько сомножителей. Иначе других сомножителей нет, так что $\neg a$ и есть искомая конъюнкция. Так как придется умножать выражения b, c, удобнее работать не с ними самими, а со списками их сомножителей. В первом случае список сомножителей выражения c дается выражением "выписка(х8 и(операнд(х7 х8)не(равно(х8 х2))) подтерм(х8))". Во втором случае c равно 1, и этот список пуст. Итак, добавляем к программе операторы "операнд(х7 х2)"; "равно(х8 вариант(символ(х7 кн)выписка(х9 и(операнд(х7 х9)не(равно(х9 х2))) подтерм(х9)) пустоеслово))". Теперь х8 - набор сомножителей выражения c. Заметим, что связанная переменная х8 была заменена на х9. В принципе, здесь это не обязательно, но лучше ее отделить от той переменной х8, которой присваивается окончательное значение.

Переходим к поиску второй конъюнкции. Для этого сначала нужно найти корневую дизъюнкцию. Если по вхождению х7 расположен символ "кн", то нужно выйти на внешнюю операцию. Иначе х7 и есть искомое вхождение дизъюнкции. Соответственно, оператор "альтернатива(символ(х7 кн)операнд(х9 х7)равно(х9 х7))" дает вхождение х9 корневой дизъюнкции. Просматриваем операнды дизъюнкции х9, отличные от исходной элементарной конъюнкции. Для этого, прежде всего, определяем вхождение х10 этой исходной конъюнкции: "равно(х10 вариант(символ(х7 кн)х7 х2))". Затем выполняется просмотр операндов: "операнд(х9 х11)", "не(равно(х11 х10))". Проверяем наличие у конъюнкции х11 сомножителя a: "альтернатива(символ(х11 кн)и(операнд(х11 х12)символ(х12 х6))и(символ(х11 х6)равно(х12 х11)))". Теперь х12 - вхождение переменной a. Как и для первой конъюнкции, определяем список сомножителей выражения b: "равно(х13 вариант(символ(х11 кн)выписка(х14 и(операнд(х11 х14)не(равно(х14 х12))) подтерм(х14)) пустоеслово))".

Проверяем, что списки х8, х13 не имеют противоположных элементов: "не(существует(х14 и(входит(х14 х8)входит(заменазнака(отр х14)х13))))". Здесь "заменазнака(отр х14)" навешивает на х14 отрицание "отр" и удаляет двойное отрицание, если оно после этого возникает.

Составляется список сомножителей произведения bc: "равно(х14 объединениесписков(х8 х13))". Если в исходных произведениях b, c не было повторяющихся множителей, то и в списке х14 нет одинаковых элементов.

Осталось проверить, что среди членов дизъюнкции нет таких, которые поглощали бы новый член: "не(существует(х15 и(операнд(х9 х15)не(равно(х15 х10))не(равно(х15 х11))) включается(наборчленов(кн подтерм(х15))х14)))". Конечно, это не очень экономно: каждый раз придется разбивать на сомножители все элементарные конъюнкции. Очевидно, лучше было бы с самого начала преобразовать д.н.ф. в формат списка списков сомножителей. Но для учебного примера сойдет.

Наконец, формируем заменяющий терм: "равно(х15 соединение(дн подтерм(х9) унисборка(кн х14)))". Завершают программу операторы "заменавхождения(х9 х3 х4 х1 х15 пустоеслово)", "пересмотр".

Заметим, что действия приема могут оказаться избыточно трудоемкими. Их можно очень существенно оптимизировать. Рекомендуем сделать это как самостоятельное упражнение. Однако, оказалось, что такая оптимизация вряд ли оправдана. Основное время при работе решателя тратится не на реализацию приема, а на сканирование задачи и поиск приема. Оптимизация трудоемкости даже "в разы" на этом фоне обычно остается незамеченной. Кроме того, возникающие в логических процессах данные обычно достаточно компактны. Появление чрезмерно громоздких данных означает попросту, что логический процесс плохо организован. Их следует выносить во внешние "нелогические" вычислительные алгоритмы. Поэтому оптимизация приемов по трудоемкости не должна являться первоочередной задачей. Исключение составляют лишь редкие особые случаи. Гораздо важнее, чтобы приемы было легко создавать автоматически. В нашем примере тестовые примеры будут выполняться описанной процедурой практически мгновенно.

5. Поглощение элементарной конъюнкции K дизъюнктивной нормальной формой D означает, что выполняется неравенство $K \leq D$. Иными словами, множество точек, в которых K обращается в единицу, представляет собой подмножество точек, в которых D равна единице. Если элементарная конъюнкция некоторой дизъюнктивной нормальной формы поглощается дизъюнкцией остальных ее конъюнкций, то ее можно отбросить, не изменяя функции, реализуемой дизъюнктивной нормальной формой. Поэтому проверка поглощения обычно используется процедурами, упрощающими д.н.ф.

   Для нашего упражнения нужно сначала создать тестовую задачу. Пусть это будет по-прежнему задача на преобразование с целью "тест". Условие будем задавать в виде "$K \leq D$". Ответом пусть служит константа "истина", если поглощение имеет место, и константа "ложь" в противном случае.

   Сразу заметим, что такой способ постановки задачи выбран в чисто учебных целях. Более грамотным было бы ввести задачу на описание без неизвестных с целью "проверка". В этом случае пришлось бы навесить на неравенство квантор общности по всем переменным. Если ограничиться лишь попыткой усмотрения поглощения, без явного указания на непоглощаемость, то можно было бы решать задачу на доказательство неравенства. Наконец, для ускорения проверки можно было бы создать вспомогательный оператор, и обращаться к нему в процессе минимизации д.н.ф.

В нашем примере введем в раздел "Мои задачи" задачу с условием $bc \leq ab \vee \neg(a)c$. Алгоритм проверки несложен. Находим значения переменных, при которых конъюнкция K обращается в единицу, подставляем эти значения в д.н.ф., упрощаем ее и проверяем, равна ли она тождественно единице. Последние два шага суть обращения к вспомогательным задачам: сначала - к задаче на преобразование, затем - к задаче на описание, условием которой служит уравнение - равенство д.н.ф. нулю. Если получается ответ "ложь", то поглощение имеет место, иначе - нет. В качестве точки привязки выберем символ "меньшеилиравно".

Начинаем составлять программу. Первые два оператора - "тип(х1 преобразовать)", "цель(х1 тест)". Определяем левую и правую части неравенства: "равно(х6 первыйтерм(х2))", "равно(х7 второйтерм(х2))". Определяем список х8 переменных, входящих в конъюнкцию х6: "равно(х8 параметры(х6))". Находим список х9 значений этих переменных, при которых конъюнкция обращается в 1: "равно(х9 выписка(х10 х11 и(входит(х10 х8)разряд(х6 х10 х11)) вариант(существует(х12 и(операнд(х12 х11)символ(х12 отр)))0 1)))". Определяем результат х10 подстановки значений х9 вместо переменных х8 в д.н.ф. х7: "результподст(х8 наборслов(х9)х7 х10)". Заметим, что подставляться должны термы, а не логические символы. Поэтому использовано выражение "наборслов", преобразующее символы 0,1 в однобуквенные термы.

Теперь нужно обратиться к задаче на преобразование, упрощающей выражение х10. Она будет применять средства, уже имеющиеся в решателе. Создаем набор х11, представляющий данную задачу: "равно(х11 набор(преобразовать набор(набор(истина)) набор(0) набор(пустоеслово пустоеслово)х10 0 пустоеслово набор(упростить одз)))". В качестве списка посылок выбираем единственную константу "истина". При этом используем цель "одз", наличие которой приведет к появлению в списке посылок указаний на двоичные значения переменных утверждения х10.

Обращаемся к решению задачи х11: "уровеньобращения(5)", "равно(х12 ответзадачи(х11))". Если х12 равно 1, то уже можно выдавать ответ. Делаем разветвление: "заголовок(х12 1) иначе 2". По метке "2" будем продолжать, если после упрощения не возникло константы 1. В текущем же фрагменте продолжаем программу операторами "заменавхождения(х2 х3 х4 х1 истина пустоеслово)", "пересмотр".

После перехода по метке "2" нужно создать задачу на описание. В этом случае есть гарантии, что выражение х12 имеет непустой список переменных. Они понадобятся как неизвестные задачи на описание. Определяем этот список х13: "равно(х13 параметры(х12))". Создаем уравнение х14: "равно(х14 запись(равно х12 0))". Наконец, вводим задачу на описание: "равно(х15 набор(описать набор(набор(истина))набор(0) набор(пустоеслово пустоеслово)набор(х14)набор(0) набор(пустоеслово пустоеслово) набор(полный явное прямойответ одз префикс(неизвестные х13) префикс(параметры х13))0))". Указание на то, что все неизвестные несущественные, должно привести к тому, что на задачу будет выдан ответ "истина" при наличии решения и "ложь" в противном случае. Заметим, что если вместо цели "явное" использовать цель "пример", то при отсутствии решений, вместо константы "ложь", выдавался бы отказ. Впрочем, в нашем примере только так отказ и может быть выдан, так что этот вариант тоже подходит.

Обращаемся к решению задачи: "уровеньобращения(5)", "равно(x16 ответзадачи(x15))". На всякий случай проверяем, что не получен отказ: "не(равно(x16 отказ))". Определяем логическую константу x17 - ответ упражнения: "равно(x17 вариант(заголовок(x16 ложь)истина ложь))". Наконец, осуществляем замену: "заменавхождения(x2 x3 x4 x1 x17 пустоеслово)", "пересмотр".

В заключение заметим, что в решателе не происходит обращений к указанным задачам. Вместо этого результат подстановки констант в д.н.ф. попросту преобразуется вспомогательным оператором "сокращднф" к виду сокращенной д.н.ф. и проверяется, получена ли единица. Но в учебных целях стоит попробовать создавать обращения к решению задач.

6. Для решения систем булевых уравнений нужно было бы создавать задачу на описание. Однако, решатель уже обучен решать такие задачи, и для данного упражнения пришлось бы находить "старые" приемы и каким-то образом их отключать. Чтобы избежать этого, в учебных целях будем оформлять задачу на решение системы булевых уравнений как задачу на преобразование, имеющую цель "тест". Условием задачи пусть будет терм "набор($A_1 = B_1, \ldots A_n = B_n$)". Чтобы ввести его формульным редактором, нужно просто последовательно вводить уравнения, отделяя их запятыми. Если $n = 1$, то условием будет равенство $A_1 = B_1$. Выбрать конкретную систему уравнений для тестирования программы предоставляем читателю.

   На этом примере продемонстрируем, как создаются программы новых операторов ЛОСа. Начнем с того, что введем в словарь системы новый символ - заголовок оператора. Пусть, например, это будет символ "булуравн". Проверяем, что символа с таким названием пока нет: в главном меню нажимаем "п", набираем слово "булуравн" и нажимаем Enter. Вместо перехода к фрагменту программы окно ввода символа расчищается. Это и указывает на отсутствие символа. Нажимаем Esc, выходим из окна ввода символа и нажимаем "р". Выбираем пункт "Ввод названия нового символа", вводим символ "булуравн" и нажимаем Enter. Система возвращается в главное меню. Снова нажимаем "п" и вводим "булуравн". Теперь система переходит в просмотр программы символа "булуравн". Так как этой программы пока нет, экран пустой. Нажимаем F3 и Enter. Экран по-прежнему пуст, но теперь мы находимся в просмотре справочных текстов, сопровождающих символ "булуравн", причем инициировали текствый редактор для ввода текста.

   Пусть наш оператор, получая на вход набор булевых уравнений и список переменных - неизвестных системы - выдает утверждение, являющееся решением системы. Такое утверждение будет построено при помощи логических связок "и", "или" из равенств, фиксирующих значения неизвестных, и утверждений "boolean(x)", означающих, что неизвестная x может принимать любое двоичное значение. Формта обращения к оператору тогда можно выбрать таким: "булуравн(x1 x2 x3)". Здесь x1 - набор уравнений, x2 - набор неизвестных, x3 - выходная переменная.

   Набираем текстовым редактором следующий текст: "булуравн(x1 x2 x3). x1 - набор булевых уравнений, x2 - набор неизвестных. Переменной x3 присваивается утверждение, являющееся решением системы уравнений x1".Обращаем

внимание, что после указания формата обращения "булуравн(x1 x2 x3)" ставится точка. Она позволяет различать описания операторов ЛОСа и описания понятий логического языка системы. После набор текста нажимаем Enter и "пробел". Это возвращает в просмотр программы, пока отсутствующей.

Заметим, что для использования оператора "булуравн" придется создавать прием, считывающий условие задачи, преобразующий его во входные данные оператора, обращающийся к оператору и заменяющий условие задачи на преобразование на найденный оператором ответ. Оставим создание такого приема на конец упражнения, а пока займемся программой самого оператора.

Нажимаем "р" и переходим к набору операторов программы оператора "булуравн". Начинается программа с операторов "программа", "метка(икс(4))". Первый из них указывает, что создается программа оператора ЛОСа, второй - что первые три программные переменные уже заняты, а первая свободная переменная - x4.

Так как оператор будет получать входные данные из условия задачи на преобразование, можно предположить, что каждое уравнение будет содержать хотя бы одну переменную. Иначе приемы решателя, еще до обращения к оператору, преобразуют равенство либо в константу "истина", либо в константу "ложь". В первом случае его можно будет не передавать оператору, во втором - сразу выдать ответ "ложь".

Систему булевых уравнений обычно решают разбором случаев. Выбирается некоторая неизвестная, имеющая наибольшее число вхождений в уравнения, и разбираются два случая: когда она равна 0 и когда равна 1. Чтобы выбрать такую неизвестную, составим набор x4 количеств вхождений неизвестных списка x2 в уравнения списка x1: "равно(x4 выписка(x5 входит(x5 x2) суммавсех(x6 x7 и(входит(x6 x1)разряд(x6 x5 x7))1)))". Чтобы найти наибольшее значение x5 набора x4, вначале присваиваем переменной x5 значение 0, а затем увеличиваем его в процессе просмотра списка x4. Одновременно вводим переменную x6, которой будет присвоена неизвестная с числом вхождений x5: "равно(x5 0)", "равно(x6 0)", "длялюбого(x7 x8 если серия(x7 x2 x8 x4)меньше(x5 буква(x8))то и(замена(x5 буква(x8))замена(x6 буква(x7))))".

После того, как неизвестная x6 выбрана, начинаем разбор случаев для ее значений. Предварительно создадим пустой накопитель x7, куда будем заносить результаты рассмотрения подслучаев. Добавляем операторы: "равно(x7 пустоеслово)", "или(равно(x8 0)равно(x8 1))". Здесь x8 - текущее значение переменной x6.

Мы будем подставлять x8 вместо переменной x6 в уравнения списка x1 и упрощать результаты подстановки. Для упрощения будет служить вспомогательная задача на преобразование. Если какой-либо результат окажется константой "ложь", то рассмотрение подслучая x8 обрывается. Если результатом будет константа "истина", то уравнение из дальнейших расмотрений отбрасывается. В остальных случаях результат упрощения уравнения будет регистрироваться в накопителе x9, который тоже инициируем пустым словом.

Переходим к записи операторов: "равно(x9 пустоеслово)", "входит(x10 x1)", "результподст(набор(x6)набор(набор(x8))x10 x11)", "равно(x12 набор(преобразовать набор(набор(истина)) набор(0) набор(пустоеслово пустоеслово) x11 0 пустоеслово набор(упростить)))", "уровеньобращения(3)", "равно(x13 ответзадачи(x12))".

Здесь x10 - текущее обрабатываемое уравнение списка x1, x11 - результат подстановки в него значения x8 вместо неизвестной x6, x12 - вспомогательная задача на преобразование, решаемая с пустым списком посылок, x13 - результат упрощения уравнения x11.

Сначала рассмотрим случай, когда x13 - константа "ложь". Тогда предпринимаем откат к рассмотрению следующего варианта x8. Вводим операторы: "заголовок(x13 ложь) иначе 1", "обрыв". Оператор "обрыв" обрывает цикл рассмотрения уравнений x10; переход "иначе 2" выполняется к новому фрагменту, который продолжит анализ терма x13. После набора указанных операторов нажимаем Enter и переходим к редактирования нового фрагмента через "иначе 1".

Здесь проверяем, что терм x13 - не константа "истина", и тогда регистрируем его в списке x9: "не(заголовок(x13 истина))", "замена(x9 суффикс(x9 x13))", "продолжение". Снова нажимаем Enter.

Теперь нужно завершить цикл просмотра уравений x10 списка x1. Для этого возвращаемся в редактирование фрагмента, содержащего оператор "входит(x10 x1)", и помещаем после него оператор "иначе 2". Нажимаем Enter. Оператор сразу же преобразуется в "иначе 1". Переходим через это "иначе" и вводим операторы, обрабатывающие накопитель x9. В нем теперь содержатся упрощенные после подстановки значения x8 уравнения.

Если накопитель x9 пуст, то все уравнения обратились в тождества. В этом случае добавляем к равенству переменной x6 константе x8 утверждения "двоичное(x)" для всех оставшихся неизвестных x, и конъюнкцию этих утверждений, как итог рассмотрения подслучая, заносим в накопитель x7. Соответствующие операторы таковы: "равно(x9 пустоеслово) иначе 1", "равно(x10 унисборка(и префикс(запись(равно x6 x8)выписка(x11 и(входит(x11 x2)не(равно(x11 x6)))запись(двоичное x11)))))", замена(x7 суффикс(x7 x10))", "продолжение".

Переход через "иначе 1" осуществляется к новому фрагменту программы, где анализируется случай непустого списка x9. Ввиду того, что все новые уравнения подвергались упрощению, они неконстантные, т.е. в списке x2 остаются неизвестные, отличные от x6. Тогда осуществляем рекурсивное обращение к оператору "булуравн". Добавляем операторы: "булуравн(x9 вычеркивание(x2 набор(x6)) x10)", "не(заголовок(x10 ложь))", "замена(x7 суффикс(x7 соединение(и запись(равно x6 x8)x10)))", "продолжение".

На этом рассмотрение подслучая для значения неизвестной x6 завершается. Чтобы обработать накопитель x7, возвращаемся к фрагменту, где имелся оператор "или(равно(x8 0)равно(x8 1))" и вставляем после него переход "иначе 3". Нажимаем Enter и редактируем новый фрагмент, переход к которому создали. Если список x7 пуст, то следует выдать окончательный результат - константу

"ложь", иначе - дизъюнкцию утверждений списка x7. Для этого вводим операторы: "равно(x8 вариант(равно(x7 пустоеслово) набор(ложь) унисборка(или x7)))", "результат(x3 x8)", "выход". На самом деле случай пустого списка x7 можно было бы не выделять, так как оператор "унисборка" в случае пустого набора термов выдает единицу соответствующей операции. В нашем случае логическая связка "или" имеет единицу "ложь".

На этом программа оператора "булуравн(x1 x2 x3)" завершена. Однако, нужно еще создать прием сканирования задачи, который будет обращаться к этому оператору. В качестве символа привязки выберем символ "равно". Уровень срабатывания пусть будет равен 3, чтобы все упрощения, связанные с константами, уже произошли. Вводим операторы: "уровень(3)", "тип(x1 преобразовать)", "цель(x1 тест)", "равно(x4 1)". Чтобы прием не срабатывал лишь однократно, будем использовать комментарий "булуравн" к задаче, блокирующий повторное срабатывание. Для такой блокировки вводим операторы "коммент(x1 булуравн)", "замечание(x1 булуравн)". Далее присваиваем переменной x6 условие задачи: "равно(x6 буква(x3))". Составляем список уравнений x7: "равно(x7 вариант(заголовок(x6 набор)набороперандов(левыйкрай(x6)) набор(x6)))". В качестве неизвестных x8 берем все параметры этих уравнений: "равно(x8 параметры(x7))".

Если в списке x7 имеется константа "ложь", то система несовместна. Тогда условие заменяем на константу "ложь": "Входит(ложь x7)иначе N", "заменавхождения(левыйкрай(буква(x3))x3 x4 x1 ложь пустоеслово)", "пересмотр". Здесь N - какой-либо не используемый в фрагменте номер перехода.

Составляем список x9, полученный из x7 отбрасыванием констант "истина": "равно(x9 выписка(x10 и(входит(x10 x7)не(заголовок(x10 истина)))x10))". Если список x9 пуст, то наши уравнения суть тождества, и заменяем условие задачи на "истина": "равно(x9 пустоеслово)иначе M", "заменавхождения(левыйкрай(буква(x3))x3 x4 x1 истина пустоеслово)", "пересмотр".

Наконец, осуществляем обращение к оператору "булуравн": "булуравн(x9 x8 x10)". "заменавхождения(левыйкрай(буква(x3))x3 x4 x1 x10 пустоеслово)", "пересмотр".

7. Начнем с создания тестовой задачи для транспонирования матрицы. Матрица представляется выражением вида "строки(набор($A_1, \ldots, A_n$))", где $A_1, \ldots, A_n$ - строки матрицы, представленные выражениями вида "набор($B_1 \ldots B_m$)". Прорисовывается она обычным образом, и для ввода ее в таком виде предусмотрен специальный интерфейс. Однако, проще будет ввести ее непосредственно в виде терма "строки(...)". Для этого, после входа в формульный редактор, нажимается "Ctr-Enter", текстовым редактором вводится слово "строки", нажимается Enter, и далее формульным редактором вводится все остальное: заключенный в скобки набор наборов элементов матрицы. Каждый набор для отдельной строки заключается в скобки, элементы набора отделяются запятыми. По окончании ввода матрицы нажимается Enter, и матрица перерисовывается в обычном виде. Выбор матрицы для проверки работы алгоритма предоставляем читателю.

   Для транспонирования матрицы создадим новое операторное выражение ЛОСа. Выберем название его, например, "транспматр". Введем новый логический

символ с таким названием и перейдем в редактирование его программы. Предварительно условимся, что операторное выражение будет иметь вид "транспматр(x1)", где x1 - выражение "строки(...)" указанного выше вида. Значением должно служить представление транспонированной матрицы в том же формате.

Программу начинаем с оператора "обращение(0)", указывающего, что реализуется операторное выражение. Затем идет оператор "метка(икс(2))", задающий номер 2 первой не определенной программной переменной. Далее составляем список x2 наборов элементов строк: "равно(x2 выписка(x3 операнд(первыйоперанд(x1)x3)набороперандов(x3)))". Чтобы получить строки транспонированной матрицы, просматриваем элементы первой строки как указатели столбца, и для каждого такого элемента выписываем соответствующий столбец. Эта работа выполняется оператором "равно(x3 выписка(x4 позиция(x4 начало(x2)) выписка(x5 входит(x5 x2) буква(соотвпозиция(x4 начало(x2)x5)))))". Далее формируется итоговое выражение x4: "равно(x4 запись(строки сборка(набор выписка(x5 входит(x5 x3)сборка(набор x5)))))". Программу операторного выражения завершает оператор "ответ(x4)".

Теперь создаем прием сканирования задачи, обращающийся к оператору "транспматр". Символом привязки будет символ "строки". Вводим операторы: "уровень(1)", "тип(x1 преобразовать)", "цель(x1 тест)", "коммент(x1 транспматр)", "замечание(x1 транспматр)", "заменавхождения(x2 x3 x4 x1 транспматр(подтерм(x2)) пустоеслово)", "пересмотр". Для блокировки повторных срабатывания здесь использован комментарий "транспматр".

Глава 9

Программы интерфейса логической системы

В процессе обучения логической системы приходится не только пользоваться ее интерфейсами, но и развивать эти интерфейсы. Поэтому создано так называемое оглавление программ, доступное из главного меню, через которое можно выйти на различные точки программ интерфейсов. Это оглавление можно рассматривать как своего рода древовидный "help", привязанный к участкам программы и содержащий краткие пояснения, что именно происходит на данном участке. При нажатии клавиши "курсор вправо" на концевом пункте оглавления программ - переход к соответствующей точке программы. Такой точкой служит фиктивный оператор "прием(N)", где N - номер контрольной точки в программе текущего логического символа. Для возвращения в оглавление программ нажимается *End*.

Чтобы создать в программе новую контрольную точку и связать ее с новым пунктом оглавления программ, нужно из просмотра программы перейти в режим просмотра операторов (клавиша "o" кир.), выделить тот оператор, перед которым вставляется новая контрольная точка, и нажать "Page Down". Данное нажатие переведет в оглавление программ. В этом оглавлении нужно создать новый концевой пункт, в качестве названия которого поместить краткое пояснение к контрольной точке. После нажатия "Enter" создается оператор "прием(N)", связанный с новым пунктом оглавления. Можно убедиться в этом, нажав "курсор вправо" и "End". Чтобы убрать новый пункт вместе с оператором "прием(N)", достаточно лишь удалить этот пункт в оглавлении. Не рекомендуется создавать новые пункты оглавления программ иными способами, так как они не будут связаны с точками программ.

Заметим, что основная часть операторов, реализованных на ЛОСе, доступна не через оглавление программ, а через пункт "Операторы ЛОСа" главного меню. Как правило, это сравнительно небольшие операторы. При выходе на концевой пункт оглавления операторов и нажатии "курсор вправо" появляется краткое описание оператора. Чтобы перейти к его ЛОС-программе, нажимается "Home", а для возвращения - "End".

Хотя программы большинства интерфейсов и описаны в монографии, для первого знакомства с ними приведем ниже краткие пояснения. В большинстве случаев их будет достаточно для самостоятельного изменения или дополнения данных программ.

9.1 Обработчики команд

За исключение редких случаев, передача информации логической системе осуществляется через оператор ЛОСа "клавиатура(x1)". После нажатия клавиши выходной переменной x1 этого оператора присваивается логический символ - код нажатой клавиши. Если была нажата левая либо правая кнопка мыши, то переменной x1 присваивается символ "мышь". Дополнительную информацию о координатах мыши система получает, обращаясь к оператору "мышь(x1 ж2 x3)". Если была нажата левая кнопка, переменной x1 присваивается 0, если правая - 2. При этом x2,x3 - символьные номера столбца и строки курсора мыши.

Во многих ситуациях удобно эмулировать последовательность нажатий клавиш, чтобы для какого-либо "автоматического" действия использовать уже имеющиеся программы ручного интерфейса. Для этого исходная задача сопровождается комментарием (автоклавиатура P), где P - набор логических символов, кодирующих необходимые клавиши. Чтобы такой комментарий был воспринят, вместо оператора "клавиатура(x1)" используется оператор "автоклавиатура(x1)". Если комментарий "автоклавиатура P" имеется, переменной x1 присваивается первый элемент списка P, а сам этот список укорачивается на единицу. При отсутствии комментария происходит обращение к ручному вводу символа x1.

После обращения к оператору ввода - "клавиатура(x1)" либо "автоклавиатура(x1)" - размещается линейная цепочка фраментов программы, обрабатывающих отдельные команды x1. В начале каждого фрагмента размещается оператор "равно(x1 S)" или дизъюнкция нескольких таких операторов. Если цепочка не закончилась, после данного оператора размещается переход "иначе 1" к следующему элементу цепочки. В редких случаях начало фрагмента цепочки может выглядеть иначе. Цепочки такого вида называем обработчиками команд. Несколько иначе устроен обработчик команд главного меню. Подробнее об этом будет сказано ниже. Здесь отметим лишь, что вместо оператора "автоклаватура(x1)" в этом обработчике используется оператор "главноеменю(1 x1)". Внутри ЛОС-программы последнего используется оператор "автоклавиатура".

Обычно обработчик команд регистрируется в оглавлении программ. Для этого выделяется отдельное подменю оглавления, которое называется "Обработчик команд", либо "Цикл обращений к клавиатуре", и т.п. Первый пункт подменю, непосредственно связанный с операторами "автоклавиатура" либо "клавиатура", называется "Исходная точка". Затем идут пункты, соответствующие отдельным командам или группам команд.

Если нужно получить код клавиши при создании ЛОС-программы, достаточно, находясь в режиме редактирования программы, нажать клавишу F8, а далее - ту клавишу, код которой требуется вставить в программу. Соответствующий код (логический символ) будет прорисован начиная с текущей позиции курсора. Если нужно, обратно, вспомнить, какой клавише соответствует уже расположенный в программе логический символ, достаточно выделить его (например, левой кнопкой мыши), после чего нажать Ctr-к (кир.). Название клавиши будет прорисовано в верхней части экрана. При нажатии любой клавиши восстановится текст программы. Обычно название клавиши, отвечающей за то или иное действие, указано в контекстном меню, вызываемом по F1.

Если нужно пополнить обработчик команд новыми действиями, для которых требуется подобрать ранее не использованную клавишу, то достаточно выделить (напри-

мер, левой кнопкой мыши) оператор "клавиатура" либо "автоклавиатура" данного обработчика, после чего нажать Ctr-н (кир.) На экране будет прорисован список названий всех не использованных в рассматриваемом обработчике клавиш. Точнее, всех клавиш, для которых в программе обработки команды отсутствует (корневое или расположенное внутри сложного оператора) равенство этой команды коду клавиши. После выбора подходящей клавиши (любой из списка) и нажатии клавиши, отличной от клавиш прокрутки вверх - вниз, произойдет откат к главному меню (точнее, внутренний перезапуск логической системы).

9.2 Оглавления логической системы

Оглавления чрезвычайно важны в интерфейсе логической системы. Задачник системы, база приемов, база теорем, справочник по системе, комментарии к ЛОС-программам основных блоков, различные вспомогательные справочники - организованы с помощью оглавлений. Фактически оглавления играют не просто роль древовидного классификатора для быстрого поиска объектов того или иного типа, но и роль функционального интерфейса, позволяющего инициировать различные процедуры.

9.2.1 Структуры данных оглавления

Структура данных оглавления представляет собой древовидную систему указателей-списков в каком-либо информационном блоке. Эти указатели-списки далее называем меню оглавления. Каждое меню имеет метки следующих типов:

1) Номера пунктов меню - логические символы $1, 2, \ldots$. По такой метке i имеется переход к паре объектов. Первый элемент данной пары - текстовый терминал, определяющий текст i - го пункта меню. Второй элемент пары - либо соответствующее i -му пункту подменю оглавления, либо логический терминал. В последнем случае данный логический терминал может содержать косвенную ссылку на подменю оглавления - терм "логсимвол(A_1A_2)", где A_1 - логический символ, по которому требуется осуществить переход в корневом каталоге текущего информационного блока к некоторому указателю - списку S; A_2 - логический символ, по которому нужно перейти далее из S к корню искомого подменю оглавления. Если логический терминал не содержит терма вида "логсимвол(...)", то он соответствует концевому пункту оглавления. Косвенные ссылки нужны в очень больших оглавлениях, которые целиком не умещаются в одном файле информационного блока. Заметим, что в отсутствии косвенных ссылок все переходы внутри оглавления (как переходы между указателями - списками) происходят внутри одного такого файла. Чтобы можно было использовать другой файл того же информационного блока, нужно выбрать логический символ A_1, отличный от того, к ветви которого относится оглавление (все объекты ветви одного и того же логического символа размещаются в одном файле), и организовать косвенную ссылку на ветвь оглавления, выносимую таким образом в отдельный файл. Обычно в качестве символа A_2 при этом выбирается логический символ "оглавление". Для "разрезания"оглавления с применением косвенных ссылок в интерфейсе оглавлений предусмотрена специальная операция (см. ниже).

2) По метке "позиция" - переход к логическому терминалу, хранящему терм "число(i j)". Здесь i - номер пункта оглавления, выбранного в последний раз при работе с данным меню (либо 0); j - номер первого пункта, с которого будет происходить

выдача меню на экран. Информация, сохраняющаяся в терминалах "позиция", позволяет прослеживать в оглавлении "текущий путь"от корня, и при повторном входе в оглавление сразу попадать в ту его точку, из которой имел место последний выход. Изменяя с помощью специального оператора текущий путь в оглавлении, можно обеспечивать автоматические переходы к различным его пунктам.

3) По метке "замечание" - переход к логическому терминалу, хранящему дополнительную информацию о ветви оглавления. Эти терминалы фактически относятся к структуре данных использующей оглавление внешней процедуры. Например, в случае базы теорем они содержат информацию о разделах, к которым относятся ветви оглавления, позволяющую логической системе самостоятельно ориентироваться в оглавлении, изначально предназначенном для работы с ним человека - оператора.

4) По метке "примечание" - переход к текстовому терминалу, хранящему сопроводительный текст для ветви оглавления.

Корни оглавлений размещаются в информационных блоках с использованием следующего соглашения. Берется указатель-список, достижимый из корневого каталога по метке "оглавление". Из этого указателя по меткам - логическим символам, рассматриваемым далее как названия типов оглавлений, и выполняются переходы к корням оглавлений. Типы используемых в системе оглавлений и дополнительные их особенности будем указывать в описании тех блоков, которые используют эти оглавления (кроме того, см. перечень типов оглавлений в справочной информации для логического символа "вид").

Для работы с оглавлениями используется оператор "оглавление(a b)". У него a - ссылка на корневой указатель - список оглавления. Оператор реализует интерфейс выбора концевого пункта оглавления и изменения самого оглавления. При обращении к нему сразу осуществляется переход к последнему ранее рассматривавшемуся при работе с данным оглавлением пункту (концевому либо промежуточному). Выходной переменной b присваивается пара (ссылка на текстовый терминал выбранного концевого пункта - ссылка на его логический терминал). Если концевой пункт выбран не был (выход из оглавления по "Esc"и т.п.), то оператор ложен.

9.2.2 Интерфейс оглавлений

Хотя вкратце интерфейс оглавлений уже был описан выше, приведем эдесь более полное его описание.

При прорисовке меню оглавления пункты нумеруются последовательными натуральными числами. Если пункт является концевым, то после его номера идет точка. Если же пункт оглавления сам является меню, то после его номера идет правая скобка.

Текущий пункт меню оглавления выделен синим цветом. Для смены этого пункта используются клавиши "курсор вверх"и "курсор вниз". В случае большого меню, не умещающегося целиком на экране, можно использовать также клавиши "PageUp"и "PageDown".

Для входа в подменю либо в интерфейс, обслуживающий выбранный концевой пункт оглавления, нажимается клавиша "курсор вправо"либо (что менее предпочтительно, так как более мобильным является перемещение с помощью четырех клавиш курсора) "Enter".

Для возвращения в надменю нажимается клавиша "курсор влево" либо "Esc". Нажатие "Esc" в корневом меню выводит из оглавления во внешнюю процедуру, обратившуюся к его просмотру; нажатие "курсор влево" в корневом меню игнорируется.

Для одношагового выхода из оглавления (в главное меню либо в другую внешнюю процедуру интерфейса), минуя все промежуточные надменю, используется клавиша "End".

Для одношагового перехода к нижнему из прорисованных на экране пункту меню используется клавиша "Ctr-курсор вниз"(одношаговый переход вверх не предусмотрен).

При нажатии клавиши для одной из цифр $0, 1, \ldots, 9$ происходит переход к первому из прорисованных на экране пунктов меню, имеющему данную последнюю цифру.

Возможно перемещение по оглавлению с помощью мыши. Если подвести курсор мыши к нужному пункту меню и нажать левую клавишу мыши, то происходит переход к подменю (либо концевому интерфейсу) этого пункта. При нажатии правой клавиши мыши происходит возвращение в надменю.

Оглавление можно изменять непосредственно в процессе работы с ним. Если нужно создать новый пункт оглавления для подменю, то нажимается клавиша "м"либо "М". В первом случае пункт будет создан в конце меню, и для него необходимо сначала расчистить место прокруткой оглавления вверх. Во втором случае новый пункт будет создан непосредственно перед текущим (выделенным синим цветом) пунктом оглавления, и окно текстового редактора, в котором следует набирать текст для нового пункта, разместится непосредственно поверх текста текущего и следующих за ним пунктов. После нажатия указанной клавиши появляется окно текстового редактора, с помощью которого вводится текст пункта. По завершении ввода (клавиша "Enter") прорисовывается модифицированное меню. Если во время ввода нажать "Esc", то будет восстановлен исходный вид меню.

Если нужно создать новый концевой пункт оглавления, то нажимается клавиша "к"либо "К". В первом случае пунтк будет создан в конце меню, во втором случае - непосредственно перед текущим пунктом оглавления. В остальном действия те же, что и при вводе нового пункта для подменю.

Для перенесения группы пунктов одного или нескольких меню на другое место используется буфер оглавления. Для отбора текущего пункта в этот буфер нажимается клавиша "б". В результате цвет пункта становится красным (если переместить указатель текущего пункта в другое место, то отобранный в буфер пункт сохраняет малиновую окраску). Чтобы исключить из буфера ранее занесенный в него текущий пункт, снова нажимается клавиша "б".

Чтобы перенести отобранные в буфер пункты на новое место, либо нажимается клавиша "и" - тогда отобранные пункты будут исключены со своих исходных позиций и занесены в конец текущего меню в том же порядке, в каком они заносились в буфер, либо нажимается клавиша "И" - тогда отобранные пункты после исключения с исходных позиций регистрируются непосредственно перед текущим пунктом текущего меню.

В особых случаях перенесение пункта оглавления на другое место оказывается невозможным. Это происходит, если при разрезании оглавления на фрагменты, отнесенные к различным логическим символам, старая и новая позиции пункта оказались относящимися к различным файлам информационного блока.

Для удаления текущего пункта меню следует нажать клавишу "Ctr-Del". Если этот пункт представлял собой подменю, то удаление его произойдет только в случае пустого подменю.

Для изменения текста текущего пункта следует нажать клавишу "р"(кир.). Тогда цвет текста становится черным и возникает курсор текстового редактора. По окончании редактирования (клавиша "Enter"в текстовом редакторе) текст пункта модифицируется. Если в текстовом редакторе нажать "Esc", то восстановится исходный вид пункта.

В процессе развития оглавления его ветвь может оказаться чрезмерно большой для сохранения ее в одном файле информационного блока (размеры таких файлов ограничены, ввиду особенностей двоичного кодирования адресов в используемой версии интерпретатора ЛОСа, 512 килобайтами). Тогда к данной ветви применяется операция разрезания оглавления. Выбирается некоторый логический символ "A по возможности, не чрезмерно загруженный в рассматриваемом информационном блоке, и данная ветвь закрепляется за этим символом. Фактически выполняются следующие действия: выбирается текущий пункт, определяющий переход к отрезаемой ветви; нажимается "Ctr-F6"; в появляющемся диалоговом окне (после слов "Косвенная ссылка:") текстовым редактором набирается терм "логсимвол(A оглавление)", и нажимается "Enter". Не рекомендуется применять данную операцию без крайней необходимости.

Кроме перечисленных выше, процедура "оглавление(...)"реализует ряд других операций, тесно связанных с внешними процедурами, обращающимися к оглавлениям. Эти операции будут указаны далее в разделах, посвященным таким процедурам.

9.2.3 Операторы для работы с оглавлениями

Приведем ряд операторов, часто используемых при работе с оглавлениями. Использование их предполагает, что содержащий оглавление информационный блок является активным.

Чтобы получить последовательность ссылок на пункты текущего пути оглавления, используется оператор "узлыоглавления(a b)". У него a - тип оглавления; выходной переменной b присваивается набор ссылок. Каждая такая ссылка - пара (ссылка на текстовый терминал пункта - ссылка на сам пункт,т.е. на указатель либо терминал). Начало набора b соответствует корню оглавления, причем вместо ссылки на текстовый терминал (отсутствующий у корня) берется 0.

Если нужно получить последовательность ссылок на пункты произвольного (не обязательно текущего) пути оглавления, применяется оператор "пунктыоглавления(a b c)". У него a - тип оглавления; b - набор меток перехода (символьных номеров пунктов) вдоль требуемого пути. Выходной переменной c присваивается набор ссылок (текстовый терминал пункта - пункт), аналогичный выходному набору оператора "узлыоглавления". Единственное отличие - в наборе c ссылка на корень оглавления отброшена. Иногда нужно получить ссылки не на все пункты пути, а только на последний пункт. Тогда можно применять оператор "пунктоглавления(a b c)". Входные данные у него те же, что у оператора "пунктыоглавления", но выходной переменной c присваивается пара ссылок (текстовый терминал последнего пункта пути - последний пункт).

Чтобы сделать текущим путем оглавления путь, определяемый заданной последовательностью b символьных номеров пунктов, используется оператор "путьвоглавлении(a b c)". Здесь a - тип олгавления. Выходной переменной c присваивается пара ссылок (текстовый терминал концевого пункта пути b - концевой пункт этого пути). Чтобы получить набор символьный номеров пунктов, определяющих текущий путь в оглавлении, применяется оператор "цепьвоглавлении(a b)". У него a - тип оглавления; выходной переменной b присваивается искомый набор.

Чтобы перечислить ссылки на все пункты заданного меню оглавления, используется оператор "подпункт(a b c d)". У него a - ссылка на представляющий меню указатель - список. Выходной переменной b присваивается метка перехода к очередному пункту меню; c - ссылка на текстовый терминал этого пункта; d - ссылка на сам пункт.

Оператор "смоглавление(a b)"перечисляет ссылки на все пункты оглавления (концевые и неконцевые), достижимые из заданного пункта. a - ссылка на исходный пункт (указатель либо логический терминал). b - пара (ссылка на текстовый терминал пункта - пункт) для текущего перечисляемого пункта. Исходный пункт a из перечисления исключается.

Чтобы просмотреть все концевые пункты оглавления, расположенные в некоторой его ветви, может быть полезен оператор "следтерминал(a b c d e)". Он осуществляет переход к очередному концевому пункту внутри заданной ветви. a - тип оглавления; b - путь (набор символьных номеров выбираемых пунктов) от корня оглавления к корню его ветви; c - путь от корня оглавления к его логическому терминалу, расположенному внутри ветви. Если удается найти логический терминал, расположенный в ветви после заданного, то переменной d присваивается путь к этому терминалу от корня оглавления; переменной e - пара (ссылка на текстовый терминал, сопровождающий найденный логический терминал - ссылка на логический терминал). Иначе оператор ложен.

Оператор "ветвьоглавления(a b c d)"перечисляет все концевые пункты оглавления, достижимые из некоторого его меню, причем через пункты этого меню с номерами, не меньшими заданного. У него a - ссылка на указатель- список, представляющий данное меню; b - путь к меню от корня оглавления; c - символьный номер того пункта меню, с которого начинается перечисление. Выходная переменная d перечисляет тройки (содержимое логического терминала очередного концевого пункта - пара (ссылка на текстовый терминал, сопровождающий логический терминал - ссылка на логический терминал) - путь к концевому пункту от корня оглавления). Заметим, что этот оператор обслуживает интерфейс оглавлений для коррекции его структур данных при перестановках, вставках и удалениях пунктов; поэтому, кроме указанных выше, им выполняются также некоторые дополнительные действия (обработка комментария (оглавление A) к посылкам исходной задачи - буфера оглавления, который нужно корректировать при указанных выше операциях).

Более полный список операторов, связанных с оглавлениями, можно получить, выбрав в главном меню пункт "Операторы ЛОСа" и перейдя далее вдоль пути "Операторы и операторные выражения, реализованные на ЛОСе" - "Оглавления".

9.2.4 Типы оглавлений

Каждое оглавление имеет свое имя (иногда называемое также типом оглавления). Обычно переход к оглавлению происходит от корневого каталога информационного

блока по двум меткам - сначала по метке "оглавление", и далее по метке - типу оглавления. Перечислим типы T основных оглавлений логической системы (независимо от этого список типов оглавлений приведен в справочной информации к символу "вид", вызываемой из редактора программ ЛОСа нажатием клавиши F3).

1. T = "прием" - оглавление базы приемов. Оно расположено в 8-м информационном блоке. Концевой терминал оглавления базы приемов содержит набор термов "прием($A_1A_2A_3$)", где A_1, A_2, A_3 - стандартная ссылка на прием ГЕНОЛОГа, состоящая из логического символа A_1, номера A_2 узла статьи символа A_1, хранящего теорему приема, и заголовка приема A_3. Заголовок одного из термов "прием(...)" здесь может быть заменен на символ "текприем", для выделения текущего приема, с которого нужно начинать просмотр приемов данного терминала. Кроме термов "прием(...)", в терминале может встретиться терм "замечание($B_1 \dots B_m$)", перечисляющий замечания $B_1, \dots, B_m$ к данному концевому пункту оглавления, используемые различными работающими с базой приемов процедурами. Из неконцевого пункта оглавления по метке "замечание" может быть переход к логическому терминалу, хранящему замечания $B_1, \dots, B_m$ аналогичных типов.

2. T = "программа" - оглавление программ ЛОСа. Оно находится в 9-м информационном блоке. Концевой терминал оглавления программ хранит терм "программа(A_1A_2)", где A_1 - логический символ, A_2 - номер узла U статьи символа A_1 в 9-м информационном блоке, связанного с некоторым пунктом в программе символа A_1. Этот пункт выделяется в данной программе при помощи вспомогательного оператора "прием(A_2)", где A_2 - последовательность цифр (вместо десятичной записи числа). По метке "оглавление" из узла U - переход к логическому терминалу, хранящему путь в оглавлении программ ЛОСа к рассматриваемому его концевому терминалу.

3. T = "заголовок" - справочное оглавление заголовков приемов. Это оглавление находится в 5-м информационном блоке, где расположены также многие другие справочные оглавления. Обычно концевой терминал справочного оглавления содержит единственный терм "прием($A_1A_2A_3$)". Здесь A_1 - лог.символ, A_2 - номер узла статьи символа A_1, от которого по метке "терм" имеется переход к логическому терминалу, хранящему тот терм S, который является вспомогательным обозначением, определяемым данным концевым пунктом справочного оглавления. A_3 - имя справочного оглавления. По метке "команды" от узла статьи лог.символа в 5-м инф.блоке - переход к указателю - списку, единственная метка которого - имя рассматриваемого оглавления - ведет к указателю - списку, от которого по метке "оглавление" - переход к лог.терминалу, хранящему путь к соответствующему концевому пункту оглавления.

4. T = "условие" - справочное оглавление фильтров приемов (5-й инф.блок).

5. T = "идентификатор" - справочное оглавление указателей приемов (5-й инф. блок).

6. T = "прогрблок" - справочное оглавление информационных элементов прогр. блока, используемых компилятором ГЕНОЛОГа (5-й инф.блок).

7. T = "меню" - оглавление меню (5-й инф. блок).

8. T = "оператор" - справочное оглавление операторов ЛОСа (5-й инф.блок).

9. T = "нормистина" - оглавление блоков диалога, используемых в логической системе. Оно размещено в 5-м инф.блоке. Концевой терминал оглавления содержит терм "прием(A_1 A_2 нормистина)", где A_1 - логический символ, A_2 - номер узла статьи этого символа, хранящий описание блока диалога (см. ниже).

10. T = "теорема" - оглавление базы теорем. Оно хранится в 6-м информационном блоке. Концевой терминал оглавления содержит набор термов "теорема(A_1A_2)", где A_1 - логический символ, A_2 - номер узла статьи этого символа, из которого по метке "терм" - переход к логическому терминалу, содержащему теорему. Этот узел называем также узлом данной теоремы. Заголовок одного из термов "теорема(...)"может быть заменен на "нормуравн" (ранее здесь использовался символ "тектеорема", который затем получил другое название) - для указания на теорему, с которой начинается просмотр при обращении к терминалу. Терм "замечание($B_1 \dots B_m$)" в концевом терминале оглавления указывает набор замечаний $B_1, \dots, B_m$ к концевому пункту оглавления. Замечания к неконцевому пункту оглавления хранятся в логическом терминале, к которому от указателя этого пункта имеется переход по метке "замечание".

11. T = "задачи" - оглавление задачника. Оно размещено во 2-м информационном блоке. Концевой логический терминал оглавления задачника содержит терм "задача(A_1A_2)", где A_1 - логический символ, A_2 - номер узла статьи символа A_1, хранящего задачу. Как и в предыдущих случаях, для замечаний к концевому пункту оглавления могут быть использованы термы "замечание(...)" в концевом логическом терминале либо переходы по метке "замечание" от промежуточного указателя оглавления. Описание структур данных, используемых для хранения задач во 2-м информационном блоке, будет дано ниже в отдельной главе, посвященной программам интерфейса задачника.

12. T = "раздел" - справочное оглавление типов замечаний к разделу базы теорем (5-й инф.блок).

13. T = "блоктеорем" - справочное оглавление типов комментариев к теореме в базе теорем (5-й инф.блок).

14. T = "слово" - справочное оглавление типов комментариев к представлению слова либо терма фразы (5-й инф.блок).

15. T = "текстформ" - оглавление справочника по логической системе. Хранится в 7-м инф.блоке.

16. T = "цели" - оглавление типов целевых установок задач (2-й инф.блок).

17. T = "приемы" - оглавление типов установок на синтез приема (5-й инф.блок).

18. T = "следствия" - оглавление типов услановок на лог.вывод (5-й инф. блок).

19. T = "логсимвол" - оглавление понятий, используемых в логическом языке системы (5-й инф.блок).

20. T = "протокол" - оглавление типов протоколов базы теорем (5-й инф.блок)

21. T = "логвывод" - оглавление типов приемов вывода теорем(9-й блок). Концевой пункт этого оглавления содержит ссылку "программа(A1 A2)" на ЛОС-программу приема программирующего вывода (A1 - логический символ, A2 - номер контрольной точки программы).

22. T = "клавиатура" - оглавление логических символов, являющихся кодами клавиш (5-й инф.блок).

9.2.5 Программа оператора "оглавление"

Напомним, что обращение к оператору имеет вид "оглавление(x1 x2)", где x1 - ссылка на корневой указатель-список оглавления. Оператор реализует интерфейс выбора концевого пункта и присваивает переменной x2 пару (ссылка на текстовый терминал - ссылка на логический терминал) для выбранного концевого пункта. Помимо этого, оператор реализует множество других интерфейсов, связанных с оглавлением - редактирование оглавления и запуск различных процедур обработки его разделов. Фактически, это универсальный "пульт управления" для самых разных действий логической системы.

Выйти на начало программы оператора "оглавление" можно через пункт "Интерфейс оглавлений" - "Исходная точка" оглавления программ. Ограничимся лишь кратким описанием этой программы (более подробное описание можно найти в первом томе монографии). Переменная x3 используется для сохранения набора пар (ссылка на указатель оглавления - метка перехода из него) для пути от корня оглавления к его текущему меню. Начало пути - конец списка x3. При этом значением переменной x4 служит ссылка на текущее меню оглавления.

В различных циклах автономного решения задач или вывода теорем оглавление обеспечивает выбор очередного пункта обработки. Для такого выбора заблаговременно создаются специальные комментарии к посылкам исходной задачи. При этом надобности в прорисовке текущего меню не возникает. Иначе - процедура прорисовывает текущее меню соответственно сохраненному или установленному ранее текущему пути.

Заметим, что при выходе из оглавления, включая внутренний перезапуск системы, оглавление "запоминает" текущий путь, так что при следующем обращении к нему сразу восстанавливается старое место просмотра. Впрочем, если старый текущий пункт - подменю, при возвращении в оглавление произойдет переход не к этому пункту,а к первому пункту подменю.

Прорисовка текущего меню сопровождается созданием структуры данных, характеризующей это меню. Роль такой структуры данных играет переменная x5 - набор $(A_1 \dots A_n)$ наборов A_i, содержащих информацию о пунктах меню. Каждое A_i имеет вид $(B_1 \dots B_5)$, где B_1 - метка перехода к объекту i - го пункта меню; B_2 - ссылка на текстовый терминал этого пункта; B_3 - ссылка на объект, переход к которому осуществляется через данный пункт меню; B_4 - переменная с номером 1, если пункт предшествует серии пунктов, выданных на экран, иначе - пара (символьный номер строки, с которой выдается текст данного пункта - символьный номер строки, на которой этот текст заканчивается); B_5 - 0, если i -й пункт меню ссылается на подменю, и 1 , если он является концевым. Переменной x6 присваивается символьный номер текущего пункта отображаемого меню оглавления, переменной x7 - символьный номер пункта меню, начиная с которого происходит прорисовка на экране.

Для перенесения пунктов оглавления на новое место используется буфер оглавления, роль которого играет комментарий (оглавление A) к посылкам исходной задачи. Здесь A - набор троек (ссылка на меню оглавления - набор меток перехода вдоль пути от корня оглавления к этому меню - метка перехода к занесенному в буфер пункту меню).

При прорисовке оглавления текущий пункт выделяется синим цветом, остальные пункты - черные. Если пункт занесен в буфер оглавления, его цвет становится розовым, а если при этом он еще и текущий, то - красным. Обработчик команд оглавления вынесен в отдельное подменю "Интерфейс оглавлений" - "Обработчик команд". Его исходная точка - оператор ввода "автоклавиатура(x11)". Далее идет цепочка программ обработки команд x11. Ограничимся здесь лишь кратким перечислением основных действий обработчика:

1. Вход в подраздел либо выбор концевого пункта оглавления (по клавише "курсор вправо" либо нажатием левой кнопки мыши).

2. Возвращение в надраздел (по клавише "курсор влево" либо нажатием правой кнопки мыши)

3. Смена пункта в текущем разделе (по клавишам "курсор вверх - вниз")

4. Выход из оглавления (по клавише End)

5. Ввод нового концевого пункта либо нового подраздела.

6. Редактирование текста текущего пункта оглавления (по клавише "р")

7. Занесение в буфер оглавления либо удаление из него текущего пункта (по клавише "б")

8. Извлечение из буфера оглавления группы пунктов и вставка их начиная с концевой (клавиша "и") либо текущй (клавиша "И") позиции. Со старых мест эти пункты удаляются. Такое преобразование оглавления требует перенумерации многих его пунктов, т.е. фактическое изменение меток перехода к ним. Соответственно, должны изменяться встречные ссылки на пункты оглавления из тех объектов, на которые эти пункты ссылаются. Это достаточно сложная процедура, использующая операторы "указатель(перестановка ...), "указатель(плюссимв ...)", "ветвьоглавления(...)".

9. Удаление пункта оглавления (клавиша $Ctr - Del$). Удаление подменю допускается лишь в том случае, когда оно пустое. Удаление концевого пункта сопровождается удалением встречных ссылок на оглавление из тех объектов, на которые этот пункт ссылается. Во многих случаях удаляются и сами эти объекты. Кроме того, изменяется нумерация сохранившихся пунктов меню, и это также требует коррекции обратных ссылок. Наконец, переустанавливается текущий путь в оглавлении, так что синим цветом будет выделяться пункт, следующий за удаленным.

10. Обработка клавиш $PageUp, PageDown$ для просмотра оглавления.

11. Для перенесения теорем или приемов из одного пункта оглавления в другой они заранее регистрируются в специальных буферах. Интерфейс оглавления позволяет извлечь их из этих буферов и зарегистрировать в текущем пункте, удалив из старого пункта.

12. Копирование в текущем меню подраздела базы приемов, выбранного в альтернативной копии решателя (т.е. находящейся в директории ALT).

13. Запуск цикла просмотра приемов, теорем либо задач, удовлетворяющих заданному запросу. Сам запрос формулируется в виде ЛОС-программы ("текприем" для просмотра приемов, "тектеорема" для просмотра теорем, "текзадача" для просмотра задач).

14. Переход в различные help-оглавления.

15. Поиск приема либо теоремы по заданным логическому символу и номеру узла этого символа.

16. Переходы между оглавлениями программ, базы приемов, задачника и базы теорем.

17. Запуск цикла решения задач либо тестирования вывода теорем - последовательный либо распараллеленный. Начало и конец обрабатываемого фрагмента оглавления устанавливаются заблаговременно.

18. Переход к справочнику по системе.

19. Просмотр и редактирование комментариев к текущему пункту оглавления (логический терминал).

20. Просмотр всех теорем текущего раздела базы теорем либо приемов в виде одного списка.

21. Сброс буфера базы приемов либо базы теорем либо задачника (в эти буферы помещаются добавленные, измененные либо удаленные приемы, выведенные теоремы либо специально отобранные задачи).

22. Переход к корню буфера базы приемов, теорем либо задач.

23. Переходы между текущим экземпляром логической системы и экземпляром, находящимся в директории ALT.

24. Просмотр общей информации о решении задач раздела.

25. Операции с архивом генератора приемов.

9.3 Контекстные меню

Чтобы получить информацию о возможных действиях в текущем контексте и инициирующих эти действия клавишах, нужно нажать F1. Исключение составляет только текстовый редактор, в котором вместо F1 нажимается Ctr-F1. Такие нажатия выводят на экран контекстное меню, которое выглядит как обычное ЛОС-оглавление.

Концевые пункты оглавления кратко описывают действие, после чего в скобках помещается название нужной клавиши. Чтобы активировать действие, можно либо нажать "курсор вправо", находясь на данном концевом пункте, либо нажать клавишу вручную. Ручное нажатие данной клавиши вызовет соответствующее действие, даже если не переходить в контекстное меню. Для часто используемых действий клавиши легко запоминаются, а контекстное меню становится ненужным.

В действительности контекстные меню суть лишь ветви одного-единственного оглавления "меню", расположенного в 5-м информационном блоке. Чтобы просматривать и редактировать оглавление "меню", можно нажать "м" из корневого меню логической системы. Здесь действуют обычные правила просмотра и редактирования оглавлений.

При добавлении нового пункта, после выбора клавиши, активирующей действие этого пункта, нужно нажать "курсор вправо", нажатием *Enter* перейти в текстовый редактор, нажать $F8$, и далее - выбранную клавишу. Тогда будет прорисован логический символ - код данной клавиши. После нажатия *Enter* создание пункта завершается.

Чтобы программа вызывала нужное контекстное меню, заблаговременно в ней должен быть реализован оператор "меню(набор($i_1 \dots i_n$))", где $i_1, \dots, i_n$ - последовательность символьных номеров переходов от корня оглавления к корню контекстного меню. Например, для меню оглавления базы приемов - оператор "меню(набор(1 1))". Данный оператор устанавливает текущий путь в оглавлении "меню" на корень нужного контекстного меню, а также создает комментарий (меню ($i_1 \dots i_n$)) к посылкам исходной задачи. После этого нажатие $F1$ (для текстового редактора - $Ctr - F1$) вызовет обращение к оператору "текменю(0)". Последний оператор реализует вход в оглавление "меню" начиная с корня ветви контекстного меню. Оператор "оглавление" будет блокировать попытки выйти за рамки этой ветви. При выборе концевого пункта оглавления создается комментарий "автоклавиатура (S)" для логического символа S, соответствующего этому пункту. Исключение составляет случай, когда выбирается пункт перехода к справочнику по системе (снова клавиша $F1$). Тогда сразу же выполняется переход к данному справочнику.

Чтобы не спутать оглавление контекстного меню с обычным оглавлением, номера его пунктов отделены от своих текстов синей вертикальной линией.

9.4 Интерфейс главного меню

При запуске логической системы автоматически создается фиктивная задача на исследование, единственной посылкой которой служит однобуквенный терм "вход". Целью этой задачи является тот же символ "вход". Данная задача впоследствии фигурирует как "исходная задача". Ее роль - почтовый ящик для обмена сообщениями между самыми разными процедурами системы. Таким почтовым ящиком служит список общих комментариев к посылкам задачи.

Общий интерфейс системы реализован в виде программы символа "вход", обращение к которой происходит при сканировании списка посылок исходной задачи. Выйти на начальную точку этой программы можно через пункт "Общий интерфейс" - "Исходная точка" оглавления программ.

Перед выходом на обработчик команд главного меню предпринимает ряд несложных действий по инициализации работы системы. Например, устанавливаются размеры и

позиция окна системы. Если некоторые необходимые для работы системы информационные блоки еще не созданы, то они создаются. Прорисовывается главное меню.

Для тестирования приемов логической системы приходится использовать циклы автоматического решения задач либо вывода теорем. По завершении каждого шага такого цикла реализуется внутренний перезапуск системы (оператор "трассировка(0)"). Этот перезапуск зануляет все регистры и служебные массивы, после чего возвращает в начало программы "вход" для вновь сформированной исходной задачи. Информация о предыстории сохраняется лишь в информационных блоках системы. Поэтому, для выбора очередного шага цикла, в начале программы "вход" предпринимается извлечение информации о продолжении цикла из файлов и создание таких комментариев к посылкам исходной задаче, которые обеспечивали бы правильное продолжение цикла. Эти комментарии учитываются обработчиком команд главного меню, который и передает управление нужным программам. Обычно это лишь первый шаг в цепочке передач управления, завершающейся активацией очередного шага цикла.

Использование внутреннего перезапуска после каждого шага цикла существенно увеличивает устойчивость системы к ошибкам и сбоям различного рода, так как не накапливаются дефекты структур данных, вызванные багами программы. Видимо, их осталось крайне мало, но при полномасштабных длительных прокрутках они все же проявляют себя. Обычно это происходит при параллельной прокрутке многими потоками, когда возможности отладки и повторного наблюдения бага ограничены. Если, после сбоя некоторого шага прокрутки, запустить тот же шаг повторно, он практически всегда выполняется без сбоя. Такой повторный запуск для некоторых типов прокрутки выполняется автоматически, с возвращением в программу логической системы, если она была остановлена операционной системой. При неудаче информация о сбое фиксируется для специального анализа после прокрутки, и происходит запуск следующего шага. Только так и удается осуществлять прокрутки, насчитывающие многие тысячи шагов.

Заметим одну существенную особенность программы "вход". Для нее заблокирован сервис автоматического сдвига номеров переменных во всех операторах после текущего оператора. Это предотвращает возможные коллизии, проистекающие из-за изменения исполняемой в данный момент программы. При необходимости номера переменных придется изменять вручную.

Обработчик команд программы "вход", или что то же самое, обработчик команд главного меню, запускает все имеющиеся в логической системе процедуры. В любой момент времени стэковый кадр данной программы является первым стэковым кадром цепочки обращений к процедурам. Если требуется перейти из одной ветви программы логической системы к другой ее ветви, нужно сначала вернуться в обработчик команд главного меню, а уже из него переходить к новой ветви. Такие переходы направляются различными комментариями к посылкам исходной задачи. Перед первым переходом создается комментарий (новпозиция S) к посылкам исходной задачи, где S - логический символ, кодирующий нажатие клавиши в главном меню, обеспечивающей первый шаг движения к новой ветви. Если этот первый шаг - обращение к оглавлению, то сразу появится концевой пункт текущего пути оглавления. Для заблаговременной установки текущего пути используется оператор "путьвоглавлении". Если для завершения перехода нужны какие-то еще нажатия клавиш, они заблаговременно сохраняются в комментарии (автоклавиатура ...). Кроме

комментария (новпозиция S), имеется ряд специальных комментариев к посылкам исходной задачи, тоже обеспечивающих передачу главному меню нужного кода клавиши. Здесь мы их не рассматриваем, так как они легко вычитываются из оператора обращения к главному меню в ЛОС-программе. Для отката к обработчику команд главного меню обычно достаточно просто выйти из текущего оглавления по оператору "стоп". В более сложных случаях можно использовать какие-либо комментарии к посылкам исходной задачи, обрывающие текущее действие вплоть до отката к главному меню.

Выйти на оператор обращения к главному меню можно через пункт "Общий интерфейс" - "Обращение к главному меню" оглавления программ. Этот оператор имеет заголовок "альтернатива" и занимает пол-экрана, так как прежде всего он анализирует наличие комментариев, указывающих на выбор кода x9 нужной клавиши. Например, комментарий "анализслова" переводит в режим ввода нового слова текстового анализатора, комментарий "внешкадр" инициирует переход в оглавление приемов ГЕНОЛОГа, и т.п. В том числе, здесь же анализируется комментарий (новпозиция S). Если нет никаких указаний на автоматический выбор кода x9, реализуется обращение к оператору "главноеменю(1 x9)", который выполняет стандартную обработку выбора пунктов главного меню. Заметим, что при выборе пункта "Просмотр программы логического символа" оператор выдает лишь код "отборприемов" клавиши "п", а собственно ввод нужного логического символа реализуется уже программой обработки команды "п".

9.5 Интерфейс просмотра задач и процесса их решения

9.5.1 Интерфейс просмотра задач из задачника

Вход в просмотр задачника происходит из главного меню по нажатии клавиши "з". Выйти на соответствующую точку программы можно через пункт "Общий интерфейс" - "Обращение к оглавлению задачника" оглавления программ. Здесь выполняется обращение к оператору "оглавление", которому передается ссылка на корневой указатель-список оглавления задачника. После этого обращения выполняется оператор "списокзадач(x1 x2 x3)", который, собственно, и реализует интерфейс просмотра и редактирования задач. При обращении к оператору значением переменной x1 служит путь в оглавлении задачника к указателю-списку x2, подпункты которого суть отдельные задачи. x3 - логический символ, по которому от x2 имеется переход к текущей задаче списка. Выйти на программу оператора "списокзадач" можно через пункт "Интерфейс просмотра и редактирования задач" - "Процедура СПИСОКЗАДАЧ" оглавления программ. Напомним, что оператор "списокзадач" прорисовывает не только текущую задачу, а всю "ленту задач", образованную задачами концевого меню. Друг от друга задачи отделяются горизонтальными линиями, причем прорисовка начинается с текущей задачи. Для просмотра всей ленты можно пользоваться обычными средствами прокрутки, описанными выше - клавишами курсора (вверх - вниз), клавишами Page Up и Page Down, а также колесиком мыши. Для выхода из оператора "списокзадач" нажимается "курсор влево". При этом оператор переустанавливает текущий путь в оглавлении задачника таким образом, чтобы текущим пунктом меню оказался пункт, соответствующий задаче, расположенной на экране под самой верхней горизонтальной чертой. В цикле автоматического решения задач,

перед обращением к оператору "списокзадач", создается комментарий "автоклавиатура" для нажатия клавиши "о", инициирующей решение текущей задачи.

Для извлечения задачи из информационного блока задачника оператор "списокзадач" использует процедуру "извлечениезадачи", обращающуюся к процедуре "элементызадачи". Для прорисовки - процедуры "высотаполосы" и "смполоса".

Для выполнения прорисовки оператор "списокзадач" формирует набор x4 троек (A_1 A_2 A_3). Здесь A_1 - логический символ, являющийся меткой перехода от указателя-списка x2 к подпункту задачи A_2, A_3 - набор информационных элементов для последовательных фрагментов текста задачи. Каждый такой фрагмент прорисовывается в отдельной горизонтальной полосе. Используются информационные элементы следующих типов:

1. (терм A_1 A_2). A_1 - очередной терм задачи (посылка либо условие), A_2 - либо 0, либо (в случае прорисовки текстовым редактором) пара (B_1, B_2), где B_1 - исходная строка для прорисовки, B_2 - высота полосы для прорисовки, либо (в случае формульного редактора) тройка (B_1, B_2, B_3), где B_1, B_2 - те же, а B_3 - указатель формульного редактора для прорисовки терма.

2. (отрезок A). Указатель на горизонтальную линию, с которой начинается прорисовка задачи. A - либо 0, либо строка для прорисовки линии.

3. (слово A_1 A_2). Ссылка на текстовый терминал, достижимый из корня 2-го информационного блока (т.е. задачника) по меткам A_1, "слово". A_2 - либо 0, либо пара (B_1, B_2), где B_1 - исходная строка для прорисовки текста, B_2 - высота полосы прорисовки.

4. (набор A_1 A_2). A_1 - набор троек одного из следующих типов:

   (a) (слово B_1 B_2), где B_1 - ссылка на текстовый терминал, достижимый из корня 2-го информационного блока по меткам B_1, "слово"; B_2 - либо 0, либо пара (столбец - строка) для начала прорисовки.

   (b) (буква B_1 B_2), где B_1 - логический символ, кодирующий букву; B_2 - либо 0, либо пара (столбец - строка) для начала прорисовки.

   (c) (терм B_1 B_2), где B_1 - терм или логический символ, B_2 - либо (в случае текстового редактора) пара (столбец - строка) для начала прорисовки терма, либо (в случае формульного редактора) тройка (столбец - строка - указатель формульного редактора для прорисовки терма).

   A_2 - либо 0, либо пара (C_1, C_2), где C_1 - исходная строка полосы, C_2 - высота полосы. Фрагмент состоит из последовательно расположенных в одной полосе указанных текстов, термов и букв. По мере возможности, термы прорисованы формульным редактором.

5. (геомредактор A_1 A_2), где A_1 - тройка (описание чертежа в формате оператора ГЕОМРЕДАКТОР - верхняя строка рамки - нижняя строка рамки); A_2 - либо 0, либо пара (B_1 B_2), где B_1 - исходная строка для прорисовки чертежа, B_2 - высота полосы чертежа. До прорисовки последние элементы тройки A_1 могут отличаться от $B_1, B_1 + B_2$, а затем становятся им равны.

6. (пусто A) - указатель на пропуск вертикальной полосы высотой 19 пикселей. A - либо 0, либо набор (B), где B - номер строки.

7. (текстформ A_1 A_2), где A_1 - текстформульная структура оператора ТЕКСТФОРМ; A_2 - 0 либо пара (B_1 B_2), где B_1 - исходная строка для прорисовки текстформульного элемента, B_2 - высота текстформульной полосы.

8. (шахматы A_1 A_2 A_3), где A_1 - задача на исследование, описывающая текущую позицию; A_2 - либо 0, либо тройка (ответ - трудоемкость - замедление); A_3 - либо 0, либо пара (C_1 C_2), где C_1 - исходная строка для прорисовки изображения, C_2 - высота изображения.

Для заполнения списка x4 используются обращения к оператору "извлечениезадачи", который первоначально формирует указанные информационные элементы без данных о их размещении и размерах полос. Эти данные доопределяются оператором "высотаполосы". Наконец, оператор "смполоса" прорисовывает указанную ему полосу.

Начальная точка обработчика команд оператора "списокзадач" может быть найдена через пункт "Интерфейс просмотра и редактирования задач" - "Процедура СПИСОКЗАДАЧ" - "Цикл обращений к клавиатуре" - "Исходная точка". После перехода через "ветвь 1" здесь выполняется обращение к оператору "автоклавиатура x10". Не вдаваясь в подробности (их можно найти в первом томе монографии, хотя лучше - вычитать непосредственно из ЛОС-программ), перечислим основные команды обработчика:

1. Возвращение в оглавление задачника.

2. Прокрутка ленты задач на одну строку вверх или вниз.

3. Переход к первой задаче списка.

4. Переход к следующей задаче списка.

5. Переход к последней задаче списка.

6. Переход к началу просматриваемой задачи (к ее верхней линии) либо (если уже были на верхней линии) к предыдущей задаче.

7. Переход к следующей странице списка (т.е. прорисовка задач после последней выданной на экран задачи)

8. Переход к предыдущей странице списка.

9. Выделение мышью задачи либо ее элемента.

10. Создание бланка новой задачи.

11. Удаление выделенного условия либо выделенной посылки.

12. Удаление целевой установки.

13. Удаление чертежа, сопровождающего задачу.

14. Ввод нового условия либо новой посылки.

15. Изменение выделенного условия либо посылки.

16. Ввод либо изменение чертежа, сопровождающего задачу.

17. Просмотр и изменение оглавления целевых установок задач.

18. Ввод или изменение целевой установки задачи через оглавление.

19. Просмотр и редактирование комментариев к задаче.

20. Завершающая обработка задачи.

21. Перенесение терма на новое место.

22. Копирование в задачу списка выделенных ранее термов.

23. Отмена выделения элементов и задач.

24. Удаление выделенной задачи.

25. Включение либо выключение режима полного просмотра посылок и условий (без отбрасывания простейших сопровождающих утверждений либо с отбрасыванием).

26. Ввод и изменение целевой установки задачи текстовым редактором.

27. Копирование либо перенесение выделенной задачи.

28. Удаление сохраненного ответа задачи.

29. Вход в режим выделения подтерма.

30. Включение либо выключения режима просмотра термов задачи в скобочной записи.

31. Запуск решения задачи с необходимой трассировкой либо без трассировки.

32. Переход к просмотру следующей задачи списка выделенных задач (клавиша "ш"). Такой список создается заблаговременно в комментарии (подборнеизвестных ...). Например, при просмотре всех задач с большим замедлением, и т.п.

33. Инициализация либо редактирование задачи на естественном языке.

34. Установка либо сброс режима полного скобочного просмотра, с выделением подтермов многоцветной указкой.

35. Выбор режима трассировки (пропуск малоинформативных шагов и т.п.) - клавиша "т".

36. Просмотр и редактирование списка приемов, заблокированных при решении задачи.

37. Ввод либо редактирование рисунка.

38. Переход к просмотру ЛОС-программы вычислений, созданной для задачи на программирование.

39. Просмотр и редактирование сопровождающей задачу исходной текстовой версии ее формулировки.

40. Ввод новой задачи на анализ изображения (фотографии).

41. Переход в режим прорисовки текста задачи, сопровождающего ее формальную запись (сохраняется при прокрутке ленты задач).

42. Переход к оглавлению логического языка системы.

9.5.2 Интерфейс просмотра текущей цепи решаемых задач

Обращение к просмотру текущей цепи решаемых задач происходит из отладчика ЛОСа. Выйти на эту точку программы отладчика можно через пункт "Отладчик ЛОСа" - "Просмотр текущей информации с помощью оператора ЦЕПЬЗАДАЧ" - "Исходная точка". Заметим, что в действительности здесь создается список данных не только для задач текущей цепи, но и для пакетных операторов, через посредство которых выполняются переходы от одной задачи к другой. Чтобы учесть такие промежуточные точки, а также учесть особые типы задач - текстовые (в системе называются также текстформульными, ибо между фрагментами текста допускаются вставки термов логического языка), вводится понятие обобщенной задачи. Перечислим типы таких обобщенных задач:

1. Обычные задачи на преобразование, доказательство, исследование и описание.

2. Текстформульная задача - тройка (текстформ A_1 A_2), где A_2 - текстформульная структура, A_1 - пара (логический символ - номер узла статьи этого символа), ссылающаяся на узел данной задачи в задачнике.

3. Задача нормализатора - набор (нормализатор A_1 A_2 A_3 A_4), где A_1 - название нормализатора, A_2 - список посылок, A_3 - обрабатываемый нормализатором терм, A_4 - список комментариев.

4. Задача проверочного оператора - набор (легковидеть A_1 A_2 A_3 A_4), где A_1 - название проверочного оператора, A_2 - список посылок, A_3 - проверяемое утверждение, A_4 - список комментариев.

5. Задача пакетного синтезатора - набор (синтезатор A_1 A_2 A_3 A_4), где A_1 - название синтезатора, A_2 - список посылок, A_3 - реализуемое утверждение, A_4 - список комментариев.

6. Задача пакетного анализтора - набор (анализатор A_1 A_2 A_3 A_4), где A_1 - название анализатора, A_2 - либо задача, для посылок которой реализуется вспомогательный вывод (в момент обращения к анализатору), либо (в процессе работы аналлизатора) внутренняя задача анализатора, A_3 - уровень обращения к анализатору; A_4 - целевая установка обращения.

7. Фиктивная задача для выдачи информации о текущем примененном приеме - (прием A). A - набор информационных элементов следующих типов (только наиболее часто используемые):

(a) (замена A_1 A_2) - применяется преобразование замены. A_1 - указатель на преобразуемый терм. В случае обычной задачи A_1 есть пара (вхождение посылки либо условия - указатель посылки (0) либо условия (1); в случае нормализатора A_1 есть 0; в случае анализатора A_1 есть заменяемое утверждение из буфера анализатора. A_2 - заменяющий терм.

(b) (вывод A_1 A_2) - вывод новой посылки (при $A_1 = 0$) либо нового условия (при $A_1 = 1$). A_2 - новое утверждение.

(c) (исключение A_1 A_2) - удаление условия либо посылки обычной задачи. A_1 - вхождение удаляемого утверждения, A_2 - уазатель посылки (0) либо условия (1).

(d) (прием A) - A есть ссылка на примененный прием: либо тройка (логический символ - номер узла этого символа - заголовок приема) в случае приема ГЕНОЛОГа, либо (в случае приема ЛОСа) пара (логический символ - номер N контрольной точки "прием(N)").

(e) (ответ A) - выдается ответ A. Если A - логический символ "легковидеть", то выдается ответ "истина" проверочного оператора.

(f) (Текзадача A) - A есть вхождение текущей обобщенной задачи в цепь обобщенных задач.

(g) (замечание A_1 A_2 A_3 A_4) - вводится (при $A_1 = 1$) либо удаляется (при $A_1 = 0$) комментарий A_4. A_2 - указатель посылок (0) либо условий (1); A_3 - либо 0 (в случае общего комментария), либо тот терм, к которому относится комментарий.

(h) (титр A_1 A_2 B_1 ... B_n) - комментарий для выдачи сопровождающих срабатывание приема пояснений. Здесь A_1 - тройка (логический символ - номер узла этого символа - заголовок приема), ссылающаяся на прием ГЕНОЛОГа; A_2 - номер фрагмента текстового шаблона (число); $B_1, \ldots, B_n$ - набор термов, вставляемых в этот фрагмент.

(i) (попыткаспуска A_1 A_2) - применяется попытка замены условия задачи на описание либо доказательство на новое условие, следствием которого оно является. A_1 - вхождение заменяемого условия, A_2 - заменяющее условие.

(j) (Замена A_1 A_2 A_3) - преобразование замены группы утверждений A_2 на группу утверждений A_3. A_1 - указатель посылок (0) либо условий (1).

(k) (разборслучаев A) - инициируется разбор случаев в пакетном нормализаторе по дизъюнкции A.

(l) (подслучаи A) - переход к рассмотрению подслучая A в накетном нормализаторе, описываемого набором утверждений A.

(m) (сборкатекста A) - указатель на формирование поясняющего текста. A - последовательность пар одного из типов: (слово A) (по меткам A, "слово" во 2-м информационном блоке - переход к текстовой заготовке); (буква A) (A - логический символ, кодирующий выдаваемую букву. Кодировка та же, что для клавиш); (терм A) (A - выдаваемый терм).

8. Фиктивная задача для отображения информации о ранее разобранных подслучаях при разборе случаев по дизъюнктивному условию либо дизъюнктивной

посылке задачи - (разборслучаев A_1 A_2 A_3). A_1 - дизъюнктивный член, соответствующий подслучаю; A_2 - найденный для него ответ; A_3 - задача, для которой проводится разбор случаев.

9. Фиктивная задача для отображения информации о текущем рассматриваемом подслучае при разборе случаев в нормализаторе - (подслучаи A). A - набор утверждений, описывающих подслучай.

10. Задача на анализ рисунка - тройка (битмэп A_1 A_2), где A_1, A_2 - логический символ и номер узла, ссылающиеся на узел задачи в задачнике.

11. Задача на анализ изображения (фотографии) - тройка (изображение A_1 A_2), где A_1, A_2 - логический символ и номер узла, ссылающиеся на узел задачи в задачнике.

Текущая цепь задач - набор обобщенных задач, создаваемый отладчиком ЛОСа при трассировке "по срабатываниям приемов". Начиная с указанной выше точки "Просмотр текущей информации с помощью оператора ЦЕПЬЗАДАЧ", заполняются накопитель x16 обобщенных задач и накопитель x17 ссылок на их стэковые кадры в глобальном стэке интерпретатора ЛОСа. Переменной x19 присваивается набор информационных элементов, характеризующих примененный прием, переменной x20 - вхождение в список x16 фиктивной задачи, описывающей текущее применение приема, либо новой задачи.

После элемента цепи обобщенных задач, описывающего срабатывание приема, обычно добавляется также несколько кадров для вспомогательных задач либо пакетных операторов, решение которых предшествовало его срабатыванию. Имеется в виду отрезок от начала программы приема до его завершающего оператора. Информация для этих кадров собирается интерпретатором ЛОСа автоматически в комментарии (буфер A) к посылкам исходной задачи. Элементы набора A суть наборы $(A_1 \ldots A_5)$, где A_1 - первая не определенная переменная в стэковом кадре программы приема перед вспомогательным обращением; A_2 - копия вспомогательной задачи либо пара (название пакетного оператора - набор его входных данных); A_3 - число на счетчике шагов на момент обращения; A_4 - сначала 0, а затем число шагов на счетчике на момент выхода; A_5 - сначало логический символ 0, а затем результат обращения (ответ задачи, либо выданный нормализатором терм, либо набор значений, полученных синтезатором, либо логический символ "истина" для обращения к проверочному оператору, либо символ "отказ"). Эти элементы расположены в наборе A в том же порядке, в котором выполнялось обращение. Чтобы интерпретатор начал собирать указанную информацию, заблаговременно (в начале трассировки по срабатываниям приемов) вводится комментарий (буфер пустоеслово). При попытке применить прием могут наблюдаться откаты по его программе. В этом случае данные об "отброшенных" решениях вспомогательных задач удаляются из комментария (буфер ...) автоматически.

Информация о промежуточных обращениях к задачам и операторам необходима, чтобы при трассировке рассмотреть подробности срабатывания приема. Во многих случаях вся основная работа выполняется как раз промежуточными задачами, и без ее показа действия приема будут непонятными. Однако, к моменту выдачи информации о срабатывании приема все вспомогательные действия уже выполнены. Сохранять полный их протокол - достаточно затратное дело, поэтому интерфейс

дает возможность просто запустить решение этих задач или пакетных операторов повторно.

Обращение к оператору "цепьзадач" имеет вид "цепьзадач(x1 x2 x3 x4)", где x1 - набор обобщенных задач, определенный для текущего шага трассировки отладчиком ЛОСа; x2 - вхождение в x1 фиктивной задачи, описывающей текущее применение приема, либо новую задачу. 3 - набор информационных элементов, характеризующих примененный прием. Оператор реализует интерфейс просмотра и изменения задач набора x1, а также сохранения элементов этих задач в текстах задачника. Измнения задач относятся только к текущему процессу решения, причем вспомогательные задачи изменениям не подлежат. Переменной x4 присваивается информация для продолжения трассировки. В частности, элемент "трассировка" в наборе x4 указывает на продолжение трассировки. Выйти на начальную точку программы оператора "цепьзадач" можно через пункт "Интерфейс просмотра и редактирования задач" - "Процедура ЦЕПЬЗАДАЧ" - "Исходная точка".

Для прорисовки обобщенных задач оператор "цепьзадач" использует процедуры "элементызадачи", "инфзадачи", "высотаполосы" и "Смполоса". Действия аналогичны тем, которые выполнялись оператором СПИСОКЗАДАЧ. На экран выдается информация о примененном приеме, над ней в цепи задач расположены текущая задача и ее надзадачи, а под ней - информация о вспомогательных задачах и пакетных операторах.

Для перехода к начальной точке обработчика команд этого оператора нужно перейти далее к пункту "Цикл обращений к клавиатуре" - "Исходная точка". Перечислим основные команды обработчика:

1. Сдвиги вверх-вниз цепи задач на одну полосу.
2. Переходы к следующей либо предыдущей страницам.
3. Выделение мышью задачи либо ее элемента.
4. Отмена выделения элементов и задач.
5. Прокрутка цепи задач для возвращения к просмотру текущей задачи.
6. Прокрутка цепи задач для возвращения к первоначальной задаче из задачника.
7. Удаление условия либо посылки (только на период решения)
8. Включение либо выключение режима полного просмотра посылок и условий (без отбрасывания очевидных сопровождающих утверждений).
9. Ввод нового условия либо новой посылки.
10. Изменение выделенного условия либо посылки.
11. Ввод либо изменение чертежа.
12. Изменение целевой установки.
13. Ввод и изменение целевой установки текстовым редактором.
14. Копирование списка выделенных термов.

15. Вход в выделение подтерма.
16. Включение либо выключение режима просмотра в скобочной записи.
17. Переход к началу просматриваемой на экране задачи либо к предыдущей задаче цепи задач.
18. Переход к следующей задаче цепи задач.
19. Просмотр и изменение комментариев задачи.
20. Очередной шаг трассировки.
21. Повторение исходного кадлра текущего шага трассировки.
22. Обращение к повторному решению вспомогательной обобщенной задачи.
23. Обрыв повторной трассировки вспомогательной задачи.
24. Включение либо отключение просмотра посылок выделенного пакетного оператора.
25. Переход к просмотру ЛОС-программы.
26. Переход к просмотру приема ГЕНОЛОГа, сработавшего на текущем шаге.
27. Выбор режима трассировки.
28. Переход к задачнику.
29. Сброс установок на прерывания и продолжение решения.
30. Включение режима просмотра информации о символах задачи, выделяемых левой кнопкой мыши.

9.6 Программа отладчика ЛОСа

Программа отладчика ЛОСа реализована на самом ЛОСе. Найти ее начальную точку можно через пункт "Отладчик ЛОСа" - "Исходная точка" оглавления программ. Отладчик выполнен как программа фиктивного операторного выражения "прерывание". Она запускается интерпретатором ЛОСа, как только выполняются условия текущего установленного прерывания (например, прерывания при обращении к программе заданного символа либо к заданному оператору, либо при достижении заданного числа шагов работы интепретатора, либо прерывания на каждом шаге работы интерпретатора, и т.п.). При этом в глобальном стэке как бы проводится черта, отделяющая кадр символа "прерывание" от предыдущего кадра стэка. По завершении выполнения программы символа "прерывание" возобновляется выполнение прерванной программы. Разумеется, программа "прерывание" имеет доступ ко всем предыдущим кадрам глобального стэка и при необходимости может их изменять. Это создает потенциальную возможность автоматического слежения "сверху" за всеми действиями решателя. Пока она почти не используется.

При обращении к программе "прерывание" все ее программные переменные, начиная с x1, не определены. Первым делом, при помощи оператора "трассировка(тип x1)",

программа устанавливает тип x1 обращения к программе. Такие обращения возникают либо при обнаружении ошибки, либо из трассировки. x1 - символьное число, согласно следующему списку:

1. переполнение зоны задач.
2. переполнение глобального стэка.
3. превышение допустимого сброса при откате.
4. исчерпание переменных.
5. нет программы логического символа, к которому обращается интерпретатор.
6. превышение максимально допустимого размера текста.
7. некорректное обращение к оператору.
8. переполнение стэка выражений.
9. обращение из трассировки.
10. контроль зацикливания по размеру глобального стэка.
11. составление протокола решения.
12. контроль отката к повторной попытке решения задачи.

Здесь же перечислим типы установок на прерывание (символьные номера). Эти установки заблаговременно устанавливаются либо отменяются оператором "трассировка(...)".

1. прерывание при выдаче ответа задачи заданного кадра глобального стэка.
2. прерывание при преобразовании задачи заданного кадра.
3. прерывание при обращении к программе заданного логического символа.
4. прерывание при обращении к фрагменту программы по ссылке в блоке программ.
5. прерывание при обращении к оператору по ссылке в зоне программ.
6. пошаговая трассировка (самая подробная, с отслеживанием шагов внутри сложных операторов).
7. пошаговая трассировка в заданном кадре глобального стэка.
8. пооператорная трассировка в заданном кадре.
9. прерывание при выходе из заданного кадра глобального стэка.
10. прерывание при выходе на кадр задачи.
11. прерывание по достижении заданного числа шагов.

12. прерывание при обращении в заданном кадре к оператору по ссылке в зоне программ.

13. прерывание при обращении в заданном кадре к фрагменту программы по ссылке в блоке программ.

14. прерывание при обращении в заданном кадре к оператору ЗАМЕНА по заданной переменной.

15. прерывание при обращении к оператору ИЗМЕНЕНИЕ для разряда заданного набора.

16. прерывание при обращении к оператору КОНТРОЛЬПРИЕМА (отслеживание начальных моментов применения приемов ГЕНОЛОГА).

17. прерывание при обращении в заданном кадре к операторам СТАНДСЛЕДСТВИЕ, КОНТРОЛЬНОРМАЛИЗАЦИИ либо при изменении значения переменной x1.

18. прерывание при обращении к оператору "прием(A)" логического символа B (A, B заданы).

19. режим составления протокола процедурой РЕШЕНИЕ.

20. прерывание при переустановке трассировки после отката по ЛИМИТу.

21. прерывание при выдаче отказа на задачу по исчерпании средств.

22. прерывание при обращении к оператору "Контрользамены".

23. прерывание при обращении к оператору "ответ" в программе справочника ФРАЗА.

После определения типа прерывания x1 оператор "трассировка(набор x2)" присваивает переменной x2 набор ($A_1 \dots A_7$) сохраненных перед входом в прерывание значений: A_1 - логический символ "нет" либо ответ (операторного выражения либо задачи, при получении которого произошло прерывание); A_2 - текущий логический символ программы интерпретатора; A_3 - уровень обращения; A_4 - смещение текущего кадра глобального стэка; A_5 - смещение вершины стэка выражений; A_6 - регистр истинности; A_7 - заголовок текущего оператора ЛОСа. По выходе из программы "прерывания" интерпретатор автоматически восстановит все необходимое для продолжения прерванной программы. Набор x2 будет использован только самой программой "прерывание" для анализа текущего контекста.

Оператор "трассировка(прерывание x3)" присваивает переменной x3 ссылку на последний в глобальном стэке кадр программы символа "прерывание", т.е. на текущий кадр отладчика. Эта ссылка - логический символ, номер которого равен смещению в глобальном стэке начала кадра.

Если усматривается, что никакие действия не нужны, то происходит немедленный выход из программы "прерывание". Если прерывание произошло при ошибке, то на экран выдается сообщение об этой ошибке.

Переменной x5 присваивается набор ссылок на кадры глобального стэка, начиная с того кадра, информация о котором будет отображаться на экране, и кончая последним кадром прерывания. Обычно между первым и последним кадрами набора x5 в глобальном стэке находятся некоторые вспомогательные кадры. При трассировке по преобразованиям задачи первый кадр набора x5 - кадр текущей задачи. Переменной x6 при этом присваивается набор ссылок на операторы продолжения по выходе из кадров набора x5; переменной x7 - тип установки на прерывание (см. выше).

Составляется список x8 ссылок на фрагменты программы, относящиеся к цепи текущего фрагмента программы. Первый элемент этого списка - логический символ, к программе которого относится фрагмент. Дальше идут ссылки на сами фрагменты - от корневого до текущего. Ссылкой на фрагмент служит четверка (смещение фрагмента в текущем массиве зоны программ - номер текущего массива - смещение в фрагменте программы оператора "ветвь" или "иначе", через который реализуется переход к подфрагменту; в случае текущего фрагмент вместо этого берется смещение текущего оператора - уровень перечисления).

Далее, в зависимости от типа прерывания, либо происходит обращение к оператору ЦЕПЬЗАДАЧ, либо прорисовывается текущий фрагмент программы, причем розовым цветом прорисовывается тот оператор, перед выполнением которого произошло прерывание. Интерфейс оператора ЦЕПЬЗАДАЧ был описан выше. Если же прорисован фрагмент программы, то включается обработчик команд отладчика ЛОСа. ВЫйти на начальную его точку можно через пункт "Отладчик ЛОСа" - "Обработчик команд" - "Исходная точка".

Заметим, что попытка трассировки отладчика ЛОСа возможна лишь после обращения к обработчику его команд. Если вставить оператор "трассировка(стоп 0)" до этого, то попытка его выполнения вызовет снова обращение к отладчику, тот - дойдет до оператора "трассировка(стоп 0)", и т.д. до бесконечности. Система зависнет.

Перечислим основные команды отладчика:

1. Просмотр текущего фрагмента (повторный вызов).

2. Переход к просмотру цепи задач (клавиша "з"). Для возвращения в режим просмотра фрагментов из просмотра цепизадач служит клавиша "ф".

3. Просмотр описания примененного приема ГЕНОЛОГа либо ЛОСа (определяются, соответственно, по пройденной контрольной точке "контрольприема(...)" либо "прием(...)".

4. Вход в оглавление программ (клавиша "Ctr-г").

5. Вход в оглавление приемов (клавиша "г").

6. Вход в оглавление теорем (клавиша "е").

7. Просмотр значений программных переменных и компонент этих значений в режиме "прокрутки вверх". Соответствующие клавиши были указаны выше в разделе, посвященном интерфейсу отладчика. Значения программных переменных определяются из просматриваемого кадра глобального стэка непосредственно. При показе набора его элементы обозначаются буквой, указывающей тип элемента ("н" - набор, "в" - вхождение, "т" - терм), и номером элемента.

Нумерация элементов сквозная по всем типам. Логические символы и переменные прорисовываются непосредственно. Чтобы можно было ссылаться на элемент по его номеру, в процессе просмотра формируются накопители x10 и x11. x10 - набор элементов; x11 - набор четверок (буква, указывающая клавишу, по которой просматривался объект ("x", "н", и т.п.) - номер объекта - строка для прорисовки - тип просмотра (0 - формульный, 1 - текстовый)). Набор x10 формируется только для элементов просматриваемых наборов, набор x11 - для всех просматриваемых объектов.

8. Сквозной просмотр набора - значения программной переменной. Позволяет просмотреть набор "вглубь" до простейших элементов, не являющихся наборами.

9. Установка прерываний различных типов.

10. Обрыв различных циклов автоматической прокрутки.

9.7 Просмотр и редактирование программ ЛОСа

Заметим, что хотя редактор программ и прорисовывает фрагмент программы ЛОСа в виде набора термов, в действительности эта программа хранится и исполняется в измененном виде - как последовательность групп $(A\ B_1 \dots B_n)$, где A - заголовок оператора ЛОСа; $B_1, \dots, B_n$ - операнды. Каждый такой операнд - либо конкретное значение, либо группа аналогичного вида, вычисляющая значение операнда. Вычисляемые операнды сохраняются в так называемом стэке выражений, который сбрасывается по мере использования его элементов для реализации операторов A. Интерпретатор, фактически, эмулирует ЛОС-машину. При этом в зоне программ исполняемая программ хранится вивде, обеспечивающем наискорейшую ее реализацию. Этот вид неэкономичен с точки зрения объема памяти (слишком много пустот, заполненных нулями). Для хранения в файлах блока программ пустоты устраняются. Таким образом, приходится работать с тремя разными форматами представления ЛОС-программ - в блоке программ, в зоне программ и в редакторе программ. Для преобразования из формата редактора программ в формат зоны программ служит функция интерпретатора konvert, для обратного преобразования - функция dekonvert.

Заметим, что прорисовка фрагмента программы осуществляется плотным образом, без каких-либо пробелов. Даже слово, обозначающее оператор, при выходе его на правый край экрана, просто продолжается с левой позиции следующей строки. Для новичка это может показаться неудобным и требующим улучшений. В действительности таким образом обеспечивается возможность целостного восприятия возможно большей части программы, которая вся находится перед глазами. Если ее отображать так, как принято в традиционных редакторах программ СИ, пришлось бы использовать интенсивные прокрутки вверх-вниз, и это было бы действительно неудобно. Конечно, без дополнительных средств структурирования сложных операторов ЛОС-программы, занимающей целый экран, обойтись невозможно. Таким средством, и притом вполне эффективным, оказалась многоцветная указка, включаемая в режиме просмотра операторов фрагмента. Она выделяет особым цветом текущий просматриваемый операнд сложного оператора, при переходе к операнду этого операнда цвет меняется, и т.д. Углубление в терм происходит при помощи клавиши "курсор вниз",

возвращение в надтерм - "курсор вверх", смена операндов одного уровня - клавишами "курсор влево - вправо". Глаз легко находит выделенный заданным цветом оператор, в то время как остальная часть текста программы воспринимается как затененная. При небольшой тренировке чтение ЛОС-программ с помощью многоцветной указки становится ничуть не более трудным, чем при традиционном структурировании программ. Однако, перед глазами находится несравненно больший объем информации, и не приходится слишком напрягать память.

Редактор программ ЛОСа представляет собой ветвь программы "вход", корень которой достижим через пункт "Общий интерфейс" - "Вход в редактор ЛОСа для заданного лог.символа" оглавления программ. Прежде всего, здесь предпринимается обращение к текстовому редактору, с помощью которого в рамочке главного меню вводится тот логический символ x13, программу которого требуется просмотреть.

Предусмотрена возможность автоматического получения символа x13 согласно комментарию (цепьвхождений s P) к посылкам исходной задачи. Этот комментарий определяет путь к некоторому фрагменту программы символа s, присваиваемого переменной x13. Набор P имеет вид $(C_0 \dots C_n)$ где C_i - терм "набор(D_1 D_2 D_3)". Каждая пара (D_1, D_2) - ссылка на фрагмент пути, D_3 - символьный номер перехода из этого фрагмента к подфрагменту. C_n - корень программы символа s, C_0 - терм "набор(D_1 D_2)", ссылающийся на последний фрагмент пути.

Переменным x15, x16 присваивается ссылка на корневой фрагмент программы символа x13 в блоке программ.

Дальнейшие действия прослеживаются через раздел "Редактор ЛОСа" оглавления программ. Здесь выделен оператор "повторение", являющийся точкой отката для перерисовки текущего фрагмента программы. Если имелся комментарий (цепьвхождений ..., то ссылка (x15, x16) переустанавливается на определямый им фрагмент программы символа x13. В любом случае переменной x12 оказывается присвоен набор троек (A_1 A_2 A_3), где (A_1, A_2) ссылка на фрагмент пути от корня к текущему фрагменту, A_3 - номер следующего перехода вдоль этого пути (число). Последняя тройка соответствует корню программы символа x13, первая - фрагменту, предшествующему текущему фрагменту. Переменной x17 присваивается сам текущий фрагмент.

Чтобы можно было выделять подтермы операторов фрагмента программы, прорисованных на экране, создается список x18 данных о размещении операторов. Каждый элемент этого списка - пятерка (столбец - строка для начала прорисовки оператора - вхождение оператора в фрагмент программы - исходная позиция текста оператора в буфере текстов - заключительная позиция в этом буфере).

Обработчик команд редактора программ ЛОСа может быть найден в разделе "Редактор ЛОСа" - "Обработчик команд просмотра фрагментов" оглавления программ. Перечислим основные команды этого обработчика:

1. Смена текущего оператора перехода (т.е. оператора "ветвь" либо "иначе"). Этот оператор прорисовывается желтым цветом. Для смены используются клавиши "курсор влево - вправо".

2. Переход к подфрагменту через текущий оператор перехода.

3. Возвращение к надфрагменту.

4. Завершение просмотра фрагментов программы и возвращение к главному меню (с внутренним перезапуском либо без него).

5. Вход в режим просмотра операторов текущего фрагмента. В этом режиме включается многоцветная указка, позволяющая выделять подтермы оператора. Сам текущий оператор прорисовывается сине-зеленым, его операнды - желтым, операнды операндов - зеленым, затем - малиновым, далее - кориневатым, и далее цвета изменяются по циклу. Для данного режима имеется свой обработчик команд, который будет приведен ниже.

6. Редактирование фрагмента программы. При переходе в данный режи в правом верхнем углу экрана появляется красная полоска, призывающая к аккуратности. Если случайно программа будет испорчена (например, отброшена нужная ее ветвь) и затем нажата клавиша Enter, завершающая редактирование, то исправить программу уже будет невозможно. Придется вводить ее заново или копировать из резервной версии решателя, находящейся в директории ALT. Во-время замеченную оплошность можно устранить, нажав Esc вместо Enter и таким образом вернувшись к первоначальной версии программы. В программе текстового редактора предусмотрены команды, относящиеся к работе с ЛОС-программой. Например, по нажатию Ctr-к текущий текст сохраняется и реализуется возвращение в просмотр всего дерева фрагментов программы, начиная с текущего фрагмента. Для возвращения в редактирование сохраненного текста можно нажать Home.

   При редактировании все новые переходы через "ветвь", "иначе" автоматически доопределяются как переходы к фрагменту, состоящему из единственного оператора "продолжение". Их нужно уточнять впоследствии.

7. Удаление подфрагмента программы, к которому есть ссылка из текущего фрагмента.

8. Удаление всей программы логического символа.

9. Поиск фрагмента программы текущего символа, содержащего заданный оператор.

10. Просмотр справочной информации о логическом символе (текущем либо другом, вводимом текстовым редактором).

11. Переход к просмотру программы следующего (имеющего на единицу больший номер) логического символа.

12. Контроль ЛОС-программы.

13. Разрезание фрагмента на два последовательных фрагмента (в конце первого из них помещаются оператор "ветвь" для перехода ко второму и оператор "продолжение".

14. Объединение двух последовательных фрагментов (в конце первого - операторы "ветвь" и "продолжение".

15. Поисковик в рамках текущей ветви программы.

16. Сохранение ссылки на корень ветви программы для ее копирования.

17. Замена текущей ветви программы на копию другой ветви, согласно сохраненной ссылке на последнюю.

18. Переключатель для работы с директорией ALT, где сохранена альтернативная копия логической системы.

19. Сохранение ссылки на ветвь альтернативной копии либо копирование этой ветви в основной экземпляр логической системы.

20. Работа со справочными оглавлениями.

21. Работа со справочником по системе.

22. Запуск программы символа "пуск", используемой для тестирования простых цепочек ЛОС-операторов.

23. Создание и редактирование блоков диалога.

24. Переход к корню программы текущего логического символа.

25. Обработчик команд мыши.

26. Переключение между основным и альтернативным экземплярами логической системы (клавиша Ctr-Tab; в альтернативном экземпляре правый верхний угол экрана помечается красным указателем ALT).

27. Возвращение к прерванному процессу редактирования сохраненного по Ctr-к фрагмента программы.

28. Коррекция оператора отката, завершающего фрагмент.

29. Поиск оператора, к которому происходит окат от текущего фрагмента.

30. Удаление из текущего фрагмента всех контрольных точек "трассировка(стоп 0)".

После перехода в режим просмотра операторов текущего фрагмента реализуется другой обработчик команд. Его можно найти в разделе "Редактор ЛОСа" - "Обработчик команд просмотра операторов в фрагмент" оглавления программ. Перед обращение к оператору "автоклавиатура(x31)", вводящему текущую команду, переменной x21 оказывается присвоена пара, первым элементом которой служит вхождение текущего оператора в фрагмент программы, переменной x25 - вхождение текущего подтерма этого оператора. Перечислим основные команды обработчика:

1. Возвращение из режима просмотра операторов в режим просмотра фрагментов.

2. Переход к следующему операнду либо (в корневом случае) к следующему оператору (курсор вправо).

3. Переход к предыдущему операнду либо (в корневом случае) к предыдущему оператору (курсор влево).

4. Вход в просмотр подтермов текущего терма (курсор вниз).

5. Возвращение к надтерму (курсор вверх).

6. Копирование ранее выделенной ветви программы. Ветвь выделяется в режиме просмотра фрагментов нажатием Insert. Нажатием "к" выделенная ветвь вставляется вместо всей ветви текущего фрагмента.

7. Разрезание фрагмента программы. Все, что расположено после выделенного оператора, переносится во вторую половину, а первая половина завершается операторами "ветвь N", "продолжение".

8. Переход в оглавление программ - либо для создания новой контрольной точки, связывающей это оглавление с программой, либо для просмотра пункта оглавления программ, уже связанного с последней пройденной контрольной точкой (оператором "прием(N)").

9. Увеличение либо уменьшение номеров программных переменных начиная с текущего оператора (обрабатывается вся ветвь текущего фрагмента программы).

10. Просмотр справочной информации о выделенном логическом символе либо о выделенной переменной (последняя берется из разделов оглавления программ, упоминающих эту переменную).

11. Обработчик команд мыши: выделение мышью оператора и просмотр справочной информации об этом операторе.

12. Переход к просмотру корневого фрагмента программы текущего логического символа.

13. Коррекция оператора отката. Ррежим коррекции этого отката ранее устанавливается нажатием "Ctr - курсор влево" из просмотра фрагмента программы, содержащего данный оператор. При этом в правом верхнем углу появляется выделенное красным слово "откат". Чтобы скорректировать этот откат для возвращения к текущему оператору, в режиме просмотра операторов фрагментов снова нажимается "Ctr - курсор влево". Фактически откат будет выполняться к первому же перечисляющему оператору, расположенному не позднее текущего оператора.

14. Вставка контрольной точки "трассировка(стоп 0)" перед текущим оператором.

15. Просмотр названия клавиши по ее логическому символу (Ctr - к).

16. Просмотр списка названий клавиш, не использованных в обработчике команд, связанным с текущим оператором ввода (Ctr - н).

9.8 Просмотр и редактирование информационных блоков

Напомним, что данные логической системы хранятся в так называемых информационных блоках. Каждый такой блок представляет собой один файл либо группу

файлов, причем за каждым блоком закреплен определенный номер. Для работы с информационными блоками выделена серия "файл(...)" реализуемых интепретатором операторов ЛОСа. В каждый момент выделяется некоторый активный информационный блок, и все эти операторы работают только с ним. Помимо доступа к информационным блокам, осуществляемым различными специальными интерфейсами системы (задачник, оглавления, база приемов ГЕНОЛОГа и т.п.), имеется возможность прямого доступа к этим блокам. Эта возможность используется крайне редко - обычно в каких-либо аварийных ситуациях. Поэтому интерфейс прямого доступа почти не развивался и обеспечивает лишь простейшие действия.

Чтобы войти в просмотр и редактирование информационного блока с номером N, нужно выбрать в главном меню пункт "Ресурсы и установки", и далее в окне "Просмотр и изменение информационного блока №" ввести десятичную запись числа N.

Начальная точка программы, работающей с информационными блоками, может быть найдена через пункт "Общий интерфейс" - "Вход в просмотр и редактирование информационных блоков" оглавления программ. Обработчик команд этой программы представлен в разделе "Интерфейс просмотра и редактирования инф. блоков" оглавления программ.

Если просматривается указатель-список, то перед обращением к оператору ввода оказывается определен набор x14 шестерок (столбец - строка - исходная позиция в буфере текстов - финальная позиция в буфере текстов - метка перехода либо номер объекта в списке объектов, переход к которым реализуется по одной и той же метке - ссылка на объект) для прорисовки элементов указателя. Значением переменной x15 служит текущая позиция набора x14. Заметим, что прорисовка указателя-списка происходит по группам. Каждая группа начинается с метки перехода в указателе, а после нее идут номера объектов, относящихся к данной метке перехода. Нумерация начинается каждый раз с единицы. Т.е. для неконцевого пункта оглавления всегда два номера: 1 - к текстовому терминалу, 2 - к объекту (подменю либо концевому логическому терминалу).

Приведем список основных команд обработчика:

1. Возвращение в главное меню (Esc, End)
2. Переход по текущему ребру указателя - списка, либо ввод метки для перехода в указателе - каталоге, либо просмотр текущего текстового черно-белого терминала (курсор вниз)
3. Переход к следующему либо предыдущему объекту текущего указателя-списка (курсоры вправо - влево)
4. Возвращение к внешнему указателю (курсор вверх)
5. Обращение к справочнику по системе (F1)
6. Ввод нового ребра указателя (списка или каталога) либо редактирование терминала (Insert)
7. Уплотнение текущего информационного блока (F3)
8. Удаление всего текущего указателя-списка либо терминала (Ctr-Del)

9. Просмотр текстового цветного терминала либо логического терминала (л)

10. Просмотр и редактирование битмэпов шахматных фигур

11. Занесение текущего объекта в буфер для копирование (к). Вставка - при нажатии Insert вместо выбора типа нового объекта нажимается "в".

9.9 Текстовый редактор

Система имеет свой собственный несложный текстовый редактор, реализованный на ЛОСе. Набираемый текст сохраняется в массиве интерпретатора ЛОСа, называемом буфером текстов. Далее его можно использовать либо как обычный текст, либо (если был набран логический символ либо группа термов в скобочной записи) переводить в логический формат. Последнее выполняется оператором "кодтекста(x1 x2 x3)". Если x1 = 0, то содержимое буфера текстов преобразуется в терм; если x1 = 1 - в набор термов; если x2 = 0 - в набор логических символов и термов; если x1 = 3 - в логический символ. Переменной x2 присваивается результат преобразования (набор в зоне задач либо, при x1 = 3, логический символ. Если x3 = 0, то преобразование успешно выполнено. Иначе переменные x2, x3 содержат информацию об ошибке в тексте.

Буфер текстов состоит из 4000 32-битных слов. Старшие 16 бит в таком слове не используются; младшие 16 представляются в виде (A_1 A_2 A_3), где A_1 - 4-битный номер цвета фона, A_2 - 4-битный номер цвета символов, A_3 - 8-битный номер символа. Кроме основного буфера текстов, введены два вспомогательных (с номерами 1 и2), а также буфер текстового редактора. Размеры всех этих буферов - такие же, как у основного буфера. Вспомогательные буферы используются для временного сохранения содержимого основного буфера; буфер текстового редактора используется для сохранения текстового фрагмента и многократного его использования при редактировании текста. Для работы с указанными буферами служат операторы "видео(0)", "видео(пусто)", "видео(значение)", "видео(вставка)", "видео(текстредактор)", "видео(исключение)", "видео(актив)", "видео(команды)", "видео(вхождение)", "видео(сборканабора)", "видео(копия)".

Главным образом, текстовый редактор используется в логической системе для редактирования ЛОС-программ. Поэтому никакое форматирование текста в нем не предусмотрено. По достижении текстом правой границы экрана он просто продолжается со следующей строки, без каких-либо знаков переноса. Как уже говорилось, для работы с программами ЛОСа такой режим (при использовании многоцветной указки) оказался удобнее версии с форматированием. Впрочем, для прочих надобностей имеется так называемый текстформульный редактор (см. ниже), который позволяет и форматировать текст, и вставлять в него формулы.

Никакого смысла в использовании "чужих" текстовых редакторов нет, так как по мере развития системы проще встраивать дополнительные действия в ее собственный редактор, не выходя из среды программирования ЛОСа.

Текстовый редактор реализован как оператор ЛОСа "текстредактор(x1 x2 x3 x4 x5 x6 x7 x8)", где x1 - исходная позиция буфера текстов, начиная с которой будут вводиться коды букв; x2,x3,x4 и x5 - координаты рамки редактирования, т.е. столбец-строка левого верхнего угла и правого нижнего угла; x6 - цвет фона; x7 - цвет символов.

Переменной x8 присваивается заключительная позиция (последняя занятая) буфера текстов. Если x7 - символ 0 либо "метка", то перед началом редактирования сохраняется старое изображение. Если x6 равно 0, то редактор использует латинскую клавиатуру.

Выйти на начальную точку программы оператора "текстредактор" можно через пункт "Вспомогательные процедуры интерфейсов" - "Текстовый редактор" - "Исходная точка" оглавления программ. Приведем основные команды обработчика команд текстового редактора:

1. Сдвиги курсора.
2. Завершение редактирования.
3. Получение справочной информации о логическом символе (после нажатия F4 возникает окно диалога, в котором набирается нужный символ, и по завершении его ввода на экран выдается информация).
4. Установка на перевод следующего нажатия клавиши в логический символ - название кода этой клавиши.
5. Откат на курсора на предыдущую позицию и стирание символа на этой позиции (Backspace). Эта же клавиша отменяет режимы вставки, удаление и перенесения фрагмента.
6. Переход в режим вставки символов (по умолчанию его нет) и выход из него. Вставка происходит только с той позиции, на которой курсор находился в момент включения режима вставки.
7. Выбор начала удаляемого фрагмента текста и его конца (сразу после выбора конца происходит удаление).
8. Отмена редактирования текста.
9. Просмотр оглавления операторов ЛОСа.
10. Увеличение либо уменьшение шага курсора (Tab)
11. Перевод кусора в начало строки (Home)
12. Перевод курсора в конец строки (End)
13. Изменение цвета фона.
14. Изменение цвета символов.
15. Справка о командах текстового редактора (F3)
16. Переход к парной скобке (F1)
17. Выбор начала и конца переносимого фрагмента, а также позиции, начиная с которой его следует разместить (F2). Для отмены последней из перечисленных позиций нажимается Backspace.
18. Удаление строки (F6) либо вставка пустой строки (F7)

19. Переключение латинской клавиатуры на ввод кириллицы (Ctr-F9). Используется, если у клавиатуры нет кириллицы. Выключение - той же клавишей.

20. Выделение начала и конца фрагмента текста, заносимого в буфер копирования (PageDown).

21. Вставка фрагмента из буфера копирования начиная с текущей позиции.

22. Вставка завершающего фрагмента программы оператора "продолжение" либо пары операторов "трассировка(стоп 0) продолжение".

23. Сохранение редактируемого текста фрагмента программы и выход в просмотр ранее созданных надфрагментов этого фрагмента.

24. Перерисовка экрана по содержимому буфера текстов.

25. Просмотр справки о набранном операторе ЛОСа.

Заметим, что переключение между латиницей и кириллицей при наборе с клавиатуры выполняется клавишами F11 (кириллица) и F12 (латиница). Это переключение выполняется в любой ситуации интерпретатором ЛОСа и не имеет отношения к программе текстового редактора.

9.10 Формульный редактор

Необходимость пополнять формульный редактор при обучении решателя возникает постоянно. Для упрощения этого процесса большинство логических символов прорисовываются формульным редактором в виде тех же словообразований, которыми они задаются в словаре. Чтобы такая прорисовка стала возможной, достаточно ввести (в виде псевдотеоремы приема) на ГЕНОЛОГе формат прорисовки символа и создать по нему серию приемов справочников, к которым обращается формульный редактор. В этом формате, в частности, можно определить кодовые комбинации клавиш (обычно - две последовательно нажимаемые клавиши) для ускоренного ввода символа. Процедура создания приемов справочников формульного редактора автоматизирована. Тем не менее, в особых случаях такая простейшая форма прорисовки может оказаться нежелательной, и тогда для расширения формульного редактора понадобится представление о его внутреннем устройстве.

Действия формульного редактора реализуются оператором "формредактор(x1 x2 ... x10)". У него x1-x4 задают координаты рамки редактирования (столбец-строка левого верхнего угла и солбец-строка правого нижнего угла); x5 - цвет фона. Редактор работает в двух режимах: ручной набор формулы (терма) и создание по заданному терму такой структуры данных, которая позволяет прорисовать его на экране в стандартной математической записи (точнее говоря, почти-стандартной). В первом режиме при обращении к оператору переменной x6 присваивается цвет символов, во втором режиме - тот терм, для которого нужно выполнить прорисовку. Переменные x7-x10 являются выходными переменными оператора. Переменной x7 присваивается корень так называемого дерева указателей - структуры данных для прорисовки терма в стандартной записи. Переменной x8 присваивается терм, полученный при ручном вводе. Переменные x9,x10 используются для уточнения размеров области

прорисовки: x9 - правый столбец собственной рамки набранной формулы, x10 - пара (верхняя строка рамки - нижняя строка рамки).

Программа формульного редактора использует ряд вспомогательных процедур, реализованных, ради ускорения вычислений, непосредственно на СИ. Эти процедуры собраны в оператор ЛОСа "формблок($N\ldots$)", где N - символьный номер реализуемой процедуры.

9.10.1 Структура данных, используемая для прорисовки формул в стандартной математической записи

Для прорисовки формулы на экране создается древовидная структура данных, воспроизводящая (быть может, с некоторыми отклонениями) древовидное строение формулы. Эта структура данных называется деревом указателей, а элементы ее (вершины) - указателями формульного редактора. Для ссылки на дерево указателей используется его корень. Каждый указатель определяет прорисовку части формулы (как правило, отдельной операции, предиката, квантора и т.п.) и хранит необходимую для этого геометрическую информацию. Он представляет собой набор $(A_1 \ldots A_7)$, элементы которого несут следующую нагрузку:

1. A_1 - символьный номер, называемый типом указателя. При инициализации указателя он иногда получает тип 0, который впоследствии заменяется на ненулевое значение. Тип 1 имеют указатели большинства операций и отношений, прорисовываемых "префиксным образом" как $f(t_1 \ldots t_m)$. Тип 2 имеют указатели "инфиксных записей" - $(t_1 f t_2 f \ldots f t_n)$. Тип 3 - указатели одноместных отношений, прорисовываемые с помощью тире ($t-$число, $t-$функция, и т.п.). Тип 4 получили указатели для прорисовки выражений "величина(...)" (десятичные числа, отличные от цифр), "степень(...)", "связприставка(...)". Последняя конструкция вводится самим формульным редактором - при анализе кванторов и описателей переменные связывающей приставки группируются под символом "связприставка". Тип 5 имеют указатели, используемые при прорисовке модуля. Тип 6 - указатели для прорисовки переменных; тип 7 - указатели функциональных переменных, и тип 8 - указатели матриц.

2. A_2 - логический символ, являющийся заголовком указателя и обычно совпадающий с названием соответствующей операции (отношения, и т.п.) в прорисовываемой формуле. Если $A_1 = 6$, то A_2 есть код клавиатуры, соответствующий переменной.

3. A_3, A_4 - координаты "начала" подтерма, определяемого ветвью указателя. A_3 - столбец; A_4 - строка. Предполагается, что у любого подтерма выделена его "главная" текстовая полоса (высотой в 19 пикселей), на которой (кроме ряда исключений, таких, как матрицы) прорисована корневая операция. A_4 есть верхняя строка этой полосы. Этот принцип ссылки на верхнюю строку текстовой полосы является в формульном редакторе стандартным.

4. A_5 - информация о внешней скобке подтерма, определяемого ветвью указателя. Если $A_5 = 0$, то такой скобки нет. При наличии скобки A_5 представляет собой пару (h, l), у которой h - высота части скобки, выступающей вверх над 19-пиксельным вертикальным отрезком главной полосы подтерма. Верхняя и нижняя части скобки строго симметричны по отношению к этой 19-пиксельной

полосе. Если скобка еще не закрыта (в процессе набора фрмулы), то $l = 0$. Иначе l есть пара (столбец - строка), определяющая координаты закрывающей скобки. Под строкой здесь понимается номер верхней строки той текстовой полосы, на уровне которой должна быть размещена закрывающая скобка.

5. A_6 - специальная информация для прорисовки(см. ниже).

6. A_7 - набор ссылок на подуказатели данного указателя. Обычно они соответствуют операндам операции (отношения, и т.п.) данного указателя и размещены в том же порядке.

Информационный элемент A_6 используется следующим образом:

1. Если $A_1 = 1$, то A_6 есть пара $(P_1\ P_2)$, где P_1 - координата (столбец - строка) текста заголовка указателя без учета внешней скобки; P_2 - либо 0, либо содержит дополнительную информацию для прорисовки сложных операций (дроби, радикалы, интегралы и т.п.).

   При A_2 = "дробь" значением P_2 служит пара (длина дробной горизонтальной черты в пикселях - указатель U на завершение набора дроби). По завершении набора дроби U становится равно символу "пустоеслово".

   При A_2 = "корень" (что соответствует случаю радикала степени от 3 до 9) либо A_2 = "квадркорень" (для квадратного радикала) значением P_2 служит четверка $(B_1\ B_2\ B_3\ B_4)$. Здесь B_1 - смещение вверх от текущей строки (т.е. от верхней строки текущей 19-пиксельной полосы) к верхней строке радикала; B_2 - смещение от текущей строки до нижней строки радикала; B_3 - указатель на завершение набора радикала (1 - набор завершен, 0 - не завершен); B_4 - длина верхней горизонтальной линии радикала. Под завершением набора в случае дробей и радикалов понимается ситуация, когда цвет прорисовки их линий становится черным.

   При A_2 = "суммавсех" либо "произведениевсех" P_2 представляет собой тройку (смещение вверх от текущей строки до верхней линии знака суммы или произведения - длина верхней горизонтальной линии знака - указатель завершения набора). Знак прорисовывается симметричным относительно текущей 19-пиксельной полосы образом. Указателем завершения набора служит 1 (до завершения - 0).

   При A_2 = "интеграл" либо "факториал" либо "числосочетаний" элемент P_2 представляет собой пару (смещение вверх от текущей строки до верхнего края знака - указатель завершения набора, такой же, как для "суммавсех").

2. Если $A_1 = 2$ (инфиксная операция), то значением A_6 служит набор $(0, (B_{21}, B_{22}), \ldots, (B_{n1}B_{n2}))$, где n - число операндов; (B_{i1}, B_{i2}) - координата позиции символа операции A_2 перед i - м операндом.

3. Если $A_1 = 3$, то A_6 есть координата (столбец-строка) начала текста, идущего после тире.

4. Если $A_1 = 4$ и A_2 - символ "степень", то A_6 есть индикатор завершения набора степени (он становится равен символу "пустоеслово", когда набор завершен).

5. Если $A_1 = 6$ и вводится переменная с индексом, то A_6 служит для указания на завершение набора индекса (равно "пустоеслово", когда индекс набран).

6. Если $A_1 = 8$ и A_2 есть логический символ "элементы", то значением A_6 служит набор $(B_1 \dots B_{n-1}C)$. Здесь $B_1, \dots B_{n-1}$ - смещения по горизонтали от левого края матрицы до начала прорисовки элементов 2-го,3-го, ... , n - го столбцов; C - смещение до закрывающей скобки матрицы. Данный указатель соответствует некоторой строке матрицы; сама матрица представлена указателем типа 8 с заголовком A_2 = "Набор", у которого $A_6 = 0$. Указатели для строк - подуказатели последнего указателя.

9.10.2 Вспомогательные процедуры формульного редакатора

Для прорисовки терма по заданному дереву указателей, а также для выполнения ряда других действий с этим деревом используется оператор "блокредактора(n a_1 $\dots a_m$)". Фактически он представляет собой семейство независимых подоператоров, причем n - номер (символьный) подоператора этого семейства. Приведем краткое описание некоторых из подоператоров оператора "блокредактора".

При $n = 2$ подоператор позволяет корректировать дерево указателей для параллельного сдвига изображения. У него a_1 - пара $(d_1 d_2)$, указывающая направление d_1 (0 - влево, 1 - вправо) и величину d_2 (число пикселей, символьный номер) сдвига по горизонтали; a_2 - пара $(e_1 e_2)$, указываюшая направление e_1 (0 - вверх, 1 - вниз)и величину e_2 сдвига по вертикали. a_3 есть максимально допустимая при сдвиге величина номера строки изображения (при превышении ее оператор ложен), a_4 - корректируемое дерево указателей.

При $n = 3$ подоператор осуществляет прорисовку терма по его дереву указателей. В этом случае a_1, a_2 - цвет фона и символов; a_3 - дерево указателей. a_4 - либо 1 (обычная прорисовка), либо 0 (виртуальная прорисовка, при которой на экране ничего не изменяется и которая служит для определения позиции курсора после прорисовки терма), либо символьный номер $n + 1$, где n - номер строки, начиная с которой происходит явная прорисовка (вышележащая часть терма на экран не выводится). Последние две возможности являются техническими и используются внутри программы формульного редактора. Выходным переменным a_5, a_6 присваиваются символьные номера столбца и строки, на которых размещается курсор по окончании прорисовки терма.

При $n = 8$ подоператор осуществляет перевод дерева указателей, полученного при ручном вводе терма, в терм. Здесь a_1 - дерево указателей; выходной переменной a_2 присваивается терм. При обнаружении ошибки в наборе терма оператор ложен.

При $n = 9$ подоператор осуществляет определение самой верхней из строк, занятой символами подтерма. У него a_1 - дерево указателей; a_2 - выходная переменная, которой присваивается символьный номер указанной строки.

Формульный редактор использует некоторое фиксированное сопоставление переменным их обозначений. Переменные от первой до двадцать пятой обозначаются последовательными малыми латинскими буквами; начиная с двадцать шестой идут большие латинские буквы; далее - те же латинские буквы с индексами. Оператор "видпеременной(a b c)"позволяет находить по переменной ее обозначение и обратно. Если $b = 0$, то он определяет по переменной a пару c, первый элемент которой есть логический символ

- код латинской буквы, второй элемент - 0 либо индекс в формате десятичного числа. Если $b = 1$, то он по паре c указанного вида выдает соответствующую переменную a. Чтобы прорисовать заданную переменную в заданном месте экрана так, как это делает формульный редактор, служит оператор "записьпеременной(t_1 t_2 t_3 t_4 t_5 t_6 t_7)". У него t_1, t_2 - символьные номера столбца и строки для прорисовки; t_3 - переменная; t_4, t_5 - цвет фона и символов. Выходным переменным t_6, t_7 присваиваются символьные номера столбца и строки позиции, расположенной после прорисованной переменной. В некоторых ситуациях (например, в редакторе приемов ГЕНОЛОГа) приходится одновременно работать с формульным и текстовым редактором, где переменные обозначаются как xi, и тогда возникает необходимость в таблице-подсказке, указывающей соответствие между разными обозначениями одной и той же переменной. Такая таблица прорисовывается оператором "обозначпеременных(a b c)". У него a - терм либо набор термов; b - номер первой строки на экране, начиная с которой будет прорисовываться соответствие между формульными и текстовыми обозначениями. Таблица представляет собой список разделенных запятыми пар "формульное обозначение переменной - текстовое ее обозначение"; она прорисовывается голубым цветом. Выходной переменной c присваивается символьный номер первой строки под таблицей.

Чтобы можно было выделять подтермы прорисованного терма, перекрашивая их в другой цвет, нужно определять соответствие между вхождениями в терм и поддеревьями дерева указателей. Для этого используется оператор "формподтерм(a b c d)". Здесь a - терм; b - его дерево указателей; c - вхождение подтерма в a. Выходной переменной d присваивается поддерево дерева b, необходимое для перекраски c. Перекраска выполняется оператором "блокредактора(3 ...)", в котором выбран требуемый цвет символов. Заметим, что не для всех вхождений в терм такое выделение поддерева указателя возможно; если оно невозможно, то оператор ложен. Приведем еще один оператор аналогичного типа (он используется оператором "формподтерм") - "формоперанды(a b c)". Здесь a - дерево указателей; b - вхождение терма, которому соответствует a. Переменная c перечисляет пары "дерево указателей для максимального допускающего отдельную прорисовку подтерма терма b - вхождение корня этого подтерма".

При решении задач иногда приходится вводить новые переменные. Чтобы при прорисовке этих переменных формульным редактором придерживаться обычных соглашений, принятых для обозначения объектов заданного типа (целые числа - буквы $m, n, k, \ldots$; точки плоскости - $A, B, C, \ldots$; функции - $f, g, h, \ldots$, и т.п.), можно использовать оператор "обозначения(a b c)". У него a - указатель способа выбора новой переменной; b - набор информационных элементов, уточняющих этот способ. Выходной переменной c присваивается новая переменная. Рассматриваются следующие способы выбора новой переменной:

1. a = "задача". В этом случае b - тройка (задача Z - переменная x - набор переменных A). Выбирается такая новая переменная y, которая будет замещать в задаче Z переменную x, и при этом не входит в набор A. Здесь оператор применяется в ситуации, когда обозначение переменной, быть может, неудачное, уже было введено, и требуется выполнить его коррекцию.

2. a = "неизвестная". В этом случае b - список переменных, от которых должна отличаться новая неизвестная.

3. a = "теорема". В этом случае b - тройка (терм T - переменная x - набор переменных A). Выбирается такая новая переменная y, которая будет замещать в терме T переменную x, и при этом не входит в набор A. Как и в случае 1), здесь имеет место коррекция уже введенного, быть может, неудачного, обозначения для переменной x теоремы T из базы теорем.

4. a = "целое". В этом случае b - список переменных, от которых должен отличаться новый целочисленный параметр.

5. a = "функция". В этом случае b - задача, для которой выбирается новый символ функциональной переменной.

6. a = "точка". В этом случае b - список переменных, от которых должно отличаться обозначение новой точки.

9.10.3 Обработчик команд формульного редактора

Выйти на оператор ввода обрабочика команд можно через пункт "Вспомогательные процедуры интерфейсов" - "Формульный редактор" - "Основной блок формульного редактора" - "Обращение к клавиатуре" оглавления программ. Введенный символ присваивается переменной x15. Напомним, что формульный редактор работает в двух режимах - либо при ручном наборе формул, либо в режиме создания дерева указателей для прорисовки уже имеющейся формулы. В первом случае x15 - реально введенный логический символ, во втором случае x15 равно 0.

Перед определением символа x15 определены значения следующих переменных. x11, x12 - координаты текущей позиции курсора. x13 - буфер, в который при ручном наборе может быть занесена копия некоторого ранее созданного терма, для вставки его в набираемую формулу. x14 - стэк указателей формульного редактора. Последний элемент набора x14 соотвествует указателю всей формулы, первый - текущему вводимому подтерму.

Оператором ввода в данном случае служит оператор "формклавиатура(x1)". Процедура "формклавиатура" видоизменяет последовательность сигналов, поступающих с клавиатуры, в соответствии с некоторыми правилами. Она имеет "память", роль которой выполняют комментарии (формклавиатура ...) и (кортеж ...) к посылкам исходной задачи. Используя эту память, процедура в определенных случаях выдает результат вообще без нажатия клавиш; в то же время, в ряде случаев для выдачи результата может понадобиться несколько нажатий клавиш. Преобразования сигналов с клавиатуры происходят в соответствии со следующими соглашениями:

1. Если имеется комментарий (кортеж A_1 A_2) к посылкам исходной задачи, то перед каждой вводимой буквой, кроме первой, вставляется символ умножения (кодируемый логическим символом "спуск"). A_1 - указатель первой буквы, вначале равный 0, а после ввода первой буквы заменяемый на 1. A_2 - буфер, в котором сохраняется очередной буквенный символ, в то время как вместо него выдается "умножение". При следующем обращении результат будет извлечен сразу из этого буфера, а в буфер будет занесен ноль. Такое преобразование используется при вводе обозначений типа $\triangle(ABC)$, когда несколько буквенных символов A, B, C должны вводиться подряд без символов разделителей (оператор "формклавиатура" автоматически вводит здесь в качестве разделителей

символы умножения, которые при определении результирующего терма будут отброшены). В прочих ситуациях формульный редактор блокирует ввод двух букв подряд либо (см.ниже) рассматривает эти буквы как кодовую комбинацию, преобразуемую в отдельный символ.

2. Вводимые с клавиатуры буквенные символы, а также специальные сигналы - пробел, слэш, звездочка, сохраняются в буфере A_1 комментария (формклавиатура A_1 A_2). Буфер A_2 этого же комментария используется для немедленной выдачи хранящихся в нем символов. Чтобы распознавать кодовые комбинации буквенных клавиш, служит справочник "префикс", которому передается содержимое буфера A_1, причем символом обращения к справочнику служит первый элемент буфера A_1. Если он выдает переменную, то заполнение буфера продолжается; если 0 - буфер сбрасывается; если выдается другой логический символ, то он рассматривается как результат распознавания кодовой комбинации клавиш. Таким образом, могут выдаваться не только коды клавиш, но также множество дополнительных кодов для последующей обработки формульным редактором.

3. Если идет процесс накопления информации о кодовой комбинации клавиш, то при поступлении очередной буквы оператор "формклавиатура" выдает сигнал об откате (логический символ "8"), занося поступившую букву в буфер A_2. При следующем обращении она будет оттуда извлечена в качестве результата.

4. Для обработки символа, поступившего после нажатия клавиши "слэш", используется справочник "суффикс", определяющий фактически выдаваемый результат.

5. Если кодовая комбинация клавиш начинается с символа "пробел", то для ее распознавания применяется не справочник "префикс", а справочник "формклавиатура".

6. Двойное нажатие клавиши "звездочка" воспринимается как кодовая комбинация, приводящая к прорисовке умножения в виде точки (при однократном нажатии применяется обычное слитное расположение сомножителей).

Если x15 отлично от 0, то предпринимается обращение к справочнику "формредактор" на символе x15. Результатом такого обращения служит четверка (i, r, p, s) характеристик графического изображения кодируемого посредством x15 символа s. i - тип символа, равный 1 для префиксных операций, 2 - для инфиксных (включая записи типа "a-число"); 3 - для степени; 4 - для "Enter" и "курсор вниз"; 5 - для скобок; 6 - для цифр и точки; 7 - для модуля, целой части, перечня и набираемой в стандартном виде матрицы; 8 - для константных символов ("е","пи","пусто" и т.п.). r - длина (символьный номер) записи символа s в 8-пиксельных единицах; p - зависящая от символа вспомогательная информация (для инфиксных операций - указатель их относительной "силы" при расстановке скобок; равенство p символу "пустоеслово" означает, что s имеет нестандартные размеры по вертикали). В случае переменной либо i равно 9 (при обработке уже имеющейся формулы), либо x16 равно 0 (при ручном вводе формулы).

Если x15 равно 0 (формула уже имеется и нужно лишь ее прорисовать), то процедура "модульредактора" перечисляет четверки указанного вида, которые появлялись бы при ручном вводе формулы.

Программа коррекции заготовки дерева указателей далее разбивается на ветви, согласно типу i очередной четверки x16. Выйти на просмотр этих ветвей можно через раздел "Вспомогательные процедуры интерфейсов" - "Формульный редактор" - Основной блок формульного редактора" - "Обработка очередного входного набора" оглавления программ. Прорисовка на экране обеспечивается процедурой "выводсимвола", которая контролирует размеры ранее набранных элементов формулы и при необходимсоти их изменяет (например, изменяет высоту радикала, длину дробной черты, перерисовывает знаменатель, если новый его элемент "упирается" в дробную черту, и т.п.).

Имеется ряд специальных нажатий клавиши, не преобразуемых в элементы набираемой формулы, а вызывающих специальные действия формульного редактора:

1. Отказ от набора формулы (Esc)
2. Откат для последнего введенного символа набираемой формулы (Backspace).
3. Обращение к справочнику по системе (F1)
4. Принудительный переход на новую строку (Ctr-курсор вниз)
5. Обращение к help-оглавлению логического языка системы (F2)
6. Расчистка текстовых пояснений, возникших после обращения к оглавлению логического языка (Ctr-F2)
7. Ввод логического символа текстовым редактором (Ctr-Enter). Завершение ввода - как обычно, по Enter.
8. Подготовка буфера x13 для вставки ранее выделенного в задачнике терма A. x13 становится равно терму "набор(A)" (Insert)
9. Подготовка буфера x13 для вставки терма через интерфейс входа в задачник (PageUp)
10. Увеличение рамки редактирования (используется при наборе пометки вершины графа) - "Ctr-курсор вверх" либо "Ctr-курсор вправо".

9.11 Текстформульный редактор

Для создания и просмотра форматированных текстов с вставленными в них формулами и чертежами применяется текстформульный редактор. Интерфейс его был описан выше. В системе с помощью текстформульного редактора реализованы общий справочник и (частично) интерфейс просмотра задач, в том числе интерфейс редактирования задач на текстовый анализ. Текст составляется из элементов, создаваемых текстовым, формульным и геометрическим редакторами. Между этими элементами могут вставляться указатели абзаца, пустой строки и пропуска заданного числа позиций (без перехода на новую строку). Реализуется текстформульный редактор оператором "текстформ(a b)", который получает в качестве входного данного текстформульную структуру a и набор b информационных элементов, определяющих режим редактирования. Текстформульная структура содержит полное описание редактируемого текстформульного массива, причем это описание хранится не в файлах, а в

оперативной памяти. Поэтому после применения оператора "текстформ"необходимо сохранить в файлах имевшие место изменения - такие действия предусматриваются особо в тех блоках интерфейса, которые обращаются к текстформульному редактору.

Текстформульный редактор использует для временного хранения в оперативной памяти текстовых фрагментов так называемый массив текстов (см. выше базисные операторы ЛОСа "видео(начало)", "видео(конец)", "видео(образ ...)", "видео(прообраз ...), "видео(сброс ...)", "видео(заменавхождения ...)", "видео(см ...)", "видео (посылка ...)", "видео(вычеркивание ...)"). Этот массив позволяет сохранять до 56000 букв; текстовые фрагменты в него заносятся из буфера текстов и возвращаются тоже в буфер текстов.

Как уже отмечалось выше, входными данными оператора "текстформ(a b)"являются текстформульная структура a и набор информационных элементов b. Текстформульная структура представляет собой набор (S) длины 1, состоящий из набора S элементов, определяющих последовательно расположенные текстформульные фрагменты. Типы этих элементов таковы:

1. (слово A_1 A_2) - A_1 и A_2 суть символьные номера начала и конца текстового фрагмента в массиве текстов;

2. (терм A) - A есть терм;

3. (позиция A) - A есть указатель форматирования. Если $A = 1$, то следующий фрагмент прорисовывается с новой строки. Если $A = 2$, то выполняется пропуск строки. Если $A = (3B)$, то перед следующим фрагментом создается пробел в B пикселей без перехода на следующую строку; B - символьное число.

4. (геомредактор A_1 A_2 A_3) - A_1 есть описание чертежа в формате оператора "геомредактор"; A_2, A_3 - верхняя и нижняя строки рамки чертежа.

Информационные элементы набора b, определяющего режим работы текстформульного редактора, бывают следующих типов:

1. (начало A) - A есть символьный номер первой строки для прорисовки (по умолчанию 0).

2. (изменение A) - A есть индикатор изменений текстформульной структуры: $A = 1$ - были изменения, $A = 0$ - не было изменений (анализируется после выхода из оператора "текстформ"для принятия решения о сохранении изменений в файлах).

3. (высотаполосы A) - прорисовки не происходит; A становится равно высоте полосы, необходимой для прорисовки.

4. "выход" - выход из процедуры сразу после прорисовки.

5. (конец A) - A есть нижняя строка полосы прорисовки (для аналиаз после выхода из оператора).

6. (цветпункта A) - A есть цвет символов для прорисовки.

7. "решение выход из оператора при нажатии любой клавиши, кроме клавиш прокрутки.

8. (видео A_1 A_2 A_3) - указатель на экранную структуру A_1 (см. ниже), символьный номер A_2 первой свободной строки для прорисовки и номер A_3 полосы экранной структуры A_1, с которой начинается прорисовка. Экранная структура определяет размещение на экране фактически прорисовываемой части текстформульного массива. Она создается автоматически самим текстформульным редактором, однако ее можно сохранить и использовать, для ускорения повторной прорисовки, при следующем обращении к текстформульному редактору. Экранная структура представляет собой набор $(P_1 \dots P_n)$ наборов P_i, представляющих собой экранные полосы. Каждое P_i имеет вид $(B_1\ B_2\ B_3\ B_4)$, где B_1 - символьный номер первой строки полосы, B_2 - высота полосы в пикселях (символьное число), B_3 - символьный номер несущей строки полосы (на ней размещаются буквы, если из-за выступающих вверх частей формул, расположенных на той же строке, приходится смещать текстовую часть вниз); B_4 - набор $(C_1 \dots C_k)$ элементов полосы. Каждое C_i имеет один из следующих видов:

   а) Текстовый элемент - набор (слово $D_1\ D_2\ D_3\ D_4\ D_5$), где D_1, D_2 - ссылки на начало и конец в массиве текстов, D_3,D_4 - символьные номера левого и правого столбцов элемента; D_5 - вхождение левого края соответствующего элемента текстформульной структуры (к одному и тому же элементу текстформульной структуры может относиться несколько информационных элементов экранной структуры, размещенных на соседних строках).

   б) Скобочный элемент - набор (терм $D_1\ D_2\ D_3\ D_4$), где D_1 - терм, прорисовываемый в скобочной записи; D_2, D_3 - левый и правый столбцы элемента; D_4 - вхождение левого края соответствующего информационного элемента текстформульной структуры.

   в) Формульный элемент - набор (формредактор $D_1\ D_2\ D_3\ D_4\ D_5$), где D_1 - терм, прорисовываемый формульным редактором; D_2 - дерево указателей формульного редактора для D_1; D_3, D_4 - левый и правый столбцы элемента; D_5 - вхождение левого края соответствующего элемента текстформульной структуры.

   г) Элемент форматирования - набор (пусто $D_1\ D_2\ D_3$), где D_1, D_2 - левый и правый столбцы пустой зоны полосы; D_3 - вхождение левого края соответствующего элемента текстформульной структуры.

   д) Геометрический элемент - набор (геомредактор $D_1\ D_2\ D_3\ D_4$), где D_1 - структура данных описания чертежа; D_2, D_3 - левый и правый столбцы чертежа; D_4 - вхождение левого края соответствующего элемента текстформульной структуры.

В информационном блоке текстформульная структура сохраняется в виде текстформульной ветви - цепочки указателей - списков, переходы между которыми осуществляются по метке "продолжение". Каждый такой указатель - список хранит несколько последовательных элементов текстформульной структуры. Переходы от него к логическим либо текстовым терминалам с этими элементами происходят по меткам "1","2", ... Если логический терминал хранит единственный терм, не начинающийся с логического символа "фикс", то этот терм определяет элемент (терм ...) текстформульной структуры; если он хранит терм "фикс(...)", то операнды данного терма образуют установку на форматирование текста - элемент (позиция ...). Если логический терминал хранит более одного терма, то он содержит описание чертежа.

Оператор "записьтекста(a b c)"позволяет записывать текстформульную структуру c в виде текстформульной ветви активного информационного блока, к которой от указателя -списка a имеется переход по метке b. Если из a по метке b ранее выходила другая ветвь, то она удаляется. Оператор "чтениетекста(a b c)"получает ссылку на указатель-список a активного информационного блока и метку b, по которой от него имеется переход к текстформульной ветви. Переменной c присваивается соответствующая текстформульная структура. Используемые к этому моменту фрагменты массива текстов сохраняются.

Выйти на обработчик команд текстформульного редактора можно через пункт "Вспомогательные процедуры интерфейсов" - "Текстформульный редактор" - "Цикл обращений к клавиатуре" оглавления программ. Приведем основные его команды:

1. Завершение редактирования.
2. Отмена редактирования.
3. Добавление текстового фрагмента в конце массива.
4. Вставка текстового фрагмента перед выделенным элементом текстформульной структуры.
5. Выбор элемента текстформульной структуры левой кнопкой мыши.
6. Изменение элемента, выбранного правой кнопкой мыши.
7. Перестановка двух элементов.
8. Удаление выделенного элемента текстформульной структуры.
9. Добавление терма формульным редактором в конце массива.
10. Вставка терма формульным редактором перед выделенным элементом текстформульной структуры.
11. Добавление терма текстовым редактором в конце массива.
12. Вставка терма текстовым редактором перед выделенным элементом текстформульной структуры.
13. Ввод абзаца либо пропуск строки в конце массива.
14. Ввод абзаца либо пропуск строки перед выделенным элементом.
15. Удаление пробелов в конце массива.
16. Удаление пробелов перед выделенным элементом.
17. Просмотр справочной информации о текстформульном редакторе (Ctr-F1).
18. Переход в режим вставки пробелов.
19. Вставка пробелов.
20. Ввод чертежа в конце массива.

21. Ввод чертежа перед выделенным элементом.
22. Сдвиг вверх на одну полосу.
23. Сдвиг вниз на одну полосу.
24. Прокрутка вниз.
25. Прокрутка вверх.

9.12 Геометрический редактор

Интерфейс геометрического редактора был описан в разделе, посвященном редактированию задач. Этот редактор предназначен для построения простейших чертежей в задачах по планиметрии. Он реализуется оператором "геомредактор(t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8)". Здесь t_1, t_2, t_3, t_4 - координаты рамки редактирования (столбец и строка верхнего левого угла; столбец и строка правого нижнего угла); t_5, t_6 - цвет фона и цвет линий. t_7 - структура данных чертежа, представляющая собой пару (A_1, A_2). A_1 есть набор пятерок $(xa_1a_2b_1b_2)$, указывающих координаты (a_1, a_2) различных точек x чертежа и координаты (b_1, b_2) позиций, на которых размещаются обозначающие точки переменные. x - переменная, обозначающая точку; a_1, a_2, b_1, b_2 - символьные номера. В координате сначала идет номер столбца, затем - номер строки. A_2 есть набор информационных элементов, определяющих прорисовку. Типы этих элементов таковы: (точка x) - прорисовка точки x; (отрезок x_1 x_j) - прорисовка отрезка с концами x_i, x_j; (прямая x_i x_j) - прорисовка прямой, проходящей через точки x_i, x_j (до границ рамки); (окружность x_i a) - прорисовка окружности с центром в точке x_i и радиусом a. Здесь a - символьный номер, определяющий величину радиуса в пикселях. Оператор "геомредактор"сначала выполняет прорисовку чертежа; если $t_8 = 0$, то на этом его работа завершается. Если же $t_8 = 1$, то после прорисовки реализуется интерфейс редактирования чертежа. При создании нового чертежа вводится структура данных с пустыми списками A_1, A_2.

Оператор "блокчертежа($na_1 \ldots a_m$)"представляет собой семейство вспомогательных подоператоров для работы со структурой данных чертежа. Приведем здесь несколько наиболее важных подоператоров.

При $n = 1$ значением a_1 служит структура данных чертежа; выходная переменная a_2 выдает номер нижней строки, занятой чертежом.

При $n = 2$ реализуется коррекция структуры данных чертежа, соответствующая его параллельному переносу. Значением a_1 здесь служит набор (геомредактор A_1 A_2 A_3), обычно используемый в качестве комментария к посылкам задачи, указывающего на сопровождающий задачу чертеж. A_1 - структура данных чертежа; A_2, A_3 - символьные номера верхней и нижней строк чертежа. Сдвиг чертежа по горизонтали определяется парой $a_2 = (b_1, b_2)$; в случае $b_1 = 0$ происходит сдвиг влево, при $b_1 = 1$ - сдвиг вправо; b_2 - величина сдвига (символьное число). Сдвиг по вертикали определяется парой $a_3 = (c_1, c_2)$. $c_1 = 0$ - сдвиг вверх, $c_1 = 1$ - вниз; c_2 - величина сдвига.

При $n = 5$ значением a_1 является структура данных чертежа; переменной a_2 присваивается символьный номер верхней строки чертежа.

Чтобы выбирать на чертеже позицию для размещения буквенного обозначения заданной точки, служит оператор "выборпозиции(t_1 t_2 t_3 t_4 t_5 t_6)". У него t_1 - структура

данных чертежа; t_2, t_3 - столбец и строка для положения новой точки на чертеже. Позиция для прорисовки обозначения этой точки выбирается исходя из соображений наименьшего наложения этого обозначения на другие элементы чертежа. Переменным t_4, t_5 присваиваются столбец и строка для прорисовки; t_6 - оценка степени наложения. При определении последней используется оператор "оценкапозиции(t_1 t_2 t_3 t_4 t_5 t_6)". У него t_1 - структура данных чертежа; t_2, t_3 - координаты для прорисовки обозначения точки; t_4, t_5 - координаты точки. Переменной t_6 присваивается оценка степени наложения.

Для сохранения чертежа в информационном блоке он перекодируется в набор термов, который и заносится в некоторый логический терминал. Используются термы следующих типов: "переменная(x $a_1a_2b_1b_2$)" - код пятерки ($xa_1a_2b_1b_2$) из A_1; "точка(x)"; "отрезок(x_ix_j)"; "прямая(x_ix_j)"; "окружность(x_ir)" - коды элементов A_2; "начало(n_1)"; "конец(n_2)"; "левыйкрай(n_3)"; "правыйкрай(n_4)" - указатели на верхнюю, нижнюю, левую и правую границы рамки чертежа. n_1, n_2, n_3, n_4 суть символьные номера; указатели левой и правой границ могут отсутствовать. Кроме того, могут использоваться некоторые дополнительные пометки (например, логический символ "чертеж" - указатель на то, что чертеж был построен решателем самостоятельно).

Оператор "регистрациячертежа(a b c d)"выполняет запись чертежа в активый информационный блок. У него a - указатель-список данного блока; b - структура данных чертежа; c - пара (верхняя строка рамки чертежа - нижняя строка рамки); d - метка, по которой описание чертежа заносится в указатель a. Оператор "чтениечертежа(a b c d)"получает в качестве входных данных указатель-список a активного информационного блока и метку b, по которой от a имеется переход к логическому терминалу с описанием чертежа. Переменной c присваивается структура данных чертежа, переменной d - пара (верхняя строка рамки - нижняя строка рамки) либо (если в логическом терминале имелись элементы "левыйкрай(...)", "правыйкрай(...)") четверка (верхняя строка - левый столбец - правый столбец - нижняя строка).

Выйти на начальную точку обработчика команд геометрического редактора можно через пункт "Вспомогательные процедуры интерфейсов" - "Геометрический редактор" - "Обработчик команд" - "Исходная точка". Приведем основные команды редактора:

1. Завершение редактирования.
2. Отмена редактирования.
3. Переход в режим ввода точек.
4. Сброс текущего режима.
5. Переход в режим проведения отрезков.
6. Переход в режим проведения окружностей.
7. Переход в режим проведения перпендикуляра.
8. Переход в режим проведения кривой.
9. Переход в режим выделения линий.
10. Переход в режим перемещения.

11. Переход в режим блокировки прорисовки отдельных точек или их обозначений.
12. Просмотр справочной информации о командах редактора.
13. Переход в режим коррекции рамки.
14. Сдвиг точки клавишами курсора.
15. Удаление выделенной точки кривой.
16. Вставка новой точки кривой до или после выделенной точки.
17. Ввод обозначения линии.
18. Отмена всех блокировок прорисовки точек и букв.
19. Обработка команд мыши.

9.13 Операторы просмотра списков

Для просмотра и (частично) редактирования списка объекта произвольных типов используется оператор "просмотрсписка(a)". У него a - структура данных просмотра - набор информационных элементов, определяющих просматриваемый список и режим просмотра; типы этих элементов приводятся ниже. Помимо прорисовки термов (в формульной либо скобочной записи), происходит прорисовка отдельных логических символов, наборов, вхождений в наборы и термы, чисел в формате с плавающей запятой. Каждый элемент просматриваемого списка выдается после своего номера начиная с отдельной строки. Если элемент - логический символ, после номера идет закрывающая скобка, иначе - точка. При прорисовке набора те его элементы, которые суть логические символы и символы переменных, изображаются непосредственно; прочие элементы обозначаются одной из букв: "т" - терм, "в" - вхождение, "н" - набор. При прорисовке вхождения в набор либо терм происходит выделение разряда набора либо подтерма по заданному вхождению голубым цветом. Число в формате с плавающей запятой выдается как терм "набор(M P)"; M, P - десятичные записи, где обычно P равно 0. Число, определяемое таким термом, есть произведение M на десять в степени P. Номер текущего просматриваемого элемента выделен голубым цветом; смена этого элемента происходит с помощью клавиш "курсор вверх" - "курсор вниз"и (постраничная прокрутка) "Page Up" - "Page Down". Переход к первому либо последнему пункту части списка, прорисованной на экране - "Ctr-курсор вверх"и "Ctr-курсор вниз". Оператор имеет два буфера - для единичного выбранного элемента либо группы элементов. Занесение текущего элемента в первый буфер (информационный элемент (терм T)в структуре данных просмотра) с исключением из него ранее имевшегося элемента происходит при нажатии "Ctr - ы"; занесение текущего элемента во второй буфер (информационный элемент (буфер B)) либо исключение из него - при нажатии "б". Номера элементов, занесенных во второй буфер, выделяются красным цветом. Для перехода к общему формульному режиму просмотра (выбран по умолчанию) нажимается "м", для перехода к скобочному режиму - "с". Выход из просмотра осуществляется следующими способами: (а) нажатием клавиши "курсор вправо"на текущем пункте - в этом случае оператор истинен; (б) нажатием клавиш "Esc"либо "пробел"либо "курсор влево в этом случае оператор ложен. Оператор предоставляет возможность изменять просматриваемый

список. Если нужно добавить в конце списка новый терм формульным редактором, нажимается клавиша "Enter"; если это нужно сделать текстовым редактором, клавиша "т". При добавлении в конце списка набора логических символов и термов также используется текстовый редактор, но здесь должна быть нажата клавиша "н". Чтобы добавить в конце списка элемент T из буфера (терм T), служит клавиша "ы". Аналогичные операции, выполняемые не в конце списка, а перед текущим его элементом, инициируются клавишами "Ф", "Т", "Н", "Ы". Если нужно изменить текущий элемент, используются клавиши "Ctr-ф"(формульный редактор), "Ctr-т", "Ctr-н"(текстовый редактор для термов и наборов). Исключение текущего элемента списка - "Ctr-Del". Для просмотра соответствия обозначений использованных в списке переменных при прорисовке их формульным и текстовым редакторами нажимается клавиша "х". По завершении просмотра и редактирования, информация о выбранных элементах (в том числе о последней текущей позиции списка) и об имевших место изменениях извлекается из соответствующих информационных элементов структуры данных просмотра. Перечислим типы этих элементов:

1. (набор A) - A есть просматриваемый набор;

2. (позиция N) - N есть символьный номер текущего просматриваемого разряда набора A (нумерация начинается с 1);

3. (область h_1 h_2) - h_1, h_2 суть символьные номера верхней и нижней строк полосы, отведенной для просмотра списка;

4. (видео A) - A есть структура данных для прорисовки элементов набора, представляющая собой набор наборов ($A_1 \dots A_5$, где A_1 - символьный номер элемента, A_2 - строка, с которой происходит выдача либо (для начального отрезка списка термов, не отображенного на экране) переменная x1; A_3 - высота полосы в пикселях; A_4 - элемент просматриваемого набора; A_5 - 0 в случае выдачи A_4 текстовым редактором и корень дерева указателей в случае выдачи A_4 формульным редактором. Элемент, представляющий собой вхождение, выделяется синим цветом; при выдаче логического символа либо переменной после номера ставится тире. Вспомогательные элементы "вариант($i_1 \dots i_n$)" и "альтернатива($i_1 \dots i_n$)" используются для прорисовки подзаголовков "Случай $i_1 \dots i_n$" и "Альтернатива $i_1 \dots i_n$".

5. "текстредактор" - термы и переменные выдаются на экран при помощи текстового редактора;

6. "сквознойпросмотр" - выход по нажатии клавиши "курсор вправо"осуществляется только на позициях списка, являющихся наборами либо вхождениями в наборы;

7. "выход" - немедленный выход после прорисовки начала списка, без обращения к клавиатуре;

8. "номера" - блокировка прорисовки номеров пунктов;

9. (исследовать A) - просматривается список посылок задачи A.

10. (терм T) - буфер для сохранения выбранного терма; при нажатии клавиши "курсор вправо"на текущем терме происходит замена T на этот терм.

11. "обл" - при нажатии клавиши "пробел"выход из процедуры происходит по значению "истина".

12. "текстоваязадача" - выход по нажатии клавиши "Enter";

13. (примечание K) - K есть набор примечаний к просматриваемому набору;

14. (см S) - S есть набор сопровождающих термы основного набора объектов. При удалении элемента основного набора такое же удаление выполняются в наборе S. Если элемент набора S - набор, содержащий логический символ "плюс", то при прорисовке на экране после номера соответствующего пункта ставится знак +; если элемент набора S - набор, содержащий логический символ "минус", то после этого номера ставится знак −. Если элемент набора S - набор, содержащий логический символ "см", то после номера ставится знак вопроса; если элемент набора S - набор, содержащий логический символ "См", то после номера ставится восклицательный знак.

15. (буфер B) - в накопителе B содержится список номеров выделенных при просмотре списка элементов.

16. (следствия A) - просматривается результирующий список вывода теоремы, ссылка на узел которой есть A.

17. (приемы A) - A есть набор термов "прием(B_1 B_2 B_3)" - ссылок на приемы, соответствующие просматриваемым термам.

18. (теоремы A) - A есть список термов "теорема(B_1 B_2)" - ссылок на просматриваемые теоремы.

19. "продолжение" - блокировка выхода по нажатии клавиш "курсор влево" либо "пробел".

20. (конец A) - при нажатии End происходит выход по "истина", причем A заменяется на 1.

21. смкадр - выход при нажатии Enter. Выход по "курсор вправо" блокируется. При нажатии "ф" - переход в отладчик ЛОСа. При выходе по Esc либо End вводится комментарий "обрыв" к посылкам исходной задачи.

22. (комментарии A) - A есть список наборов комментариев (термов, логических символов либо переменных), сопровождающих термы просматриваемого списка. При нажатии "курсор вправо" на текущем пункте - выход в просмотр пары (терм - сопровождающие его комментарии). Для возвращения - курсор влево.

23. (текст A) - A есть набор указателей на те элементы посматриваемого набора, которые суть ссылки на текстовые терминалы текущего информационного блока. Если указатель равен 0, то элемент набора выдается стандартным образом; если же он равен 1, то вместо элемента (т.е. пары логических символов, образующей ссылку на терминал) прорисовывается текст его терминала.

24. (титр A_1 A_2) - A_2 есть набор пар (начало в массиве текстов - конец в массиве текстов) ссылок на те подзаголовки, которыми сопровождаются элементы просматриваемого набора. Подзаголовок прорисовывается сразу после номера

элемента и завершается двоеточием. A_1 - индикатор изменения подзаголовков: 0 - не было изменения, 1 - было.

25. (команда A) - элемент используется для передачи информации A, необходимой для завершения обработки текущей команды во внешней процедуре. Обычно запоминает специальную нажатую клавишу.

26. (команды A) - при нажатии любой клавиши, код которой принадлежит списку A, происходит выход из просмотра по значению "истина", причем код K нажатой клавиши регистрируется в элементе (команда K).

Обязательными при обращении к оператору являются элементы типов 1-3, к которым должен быть добавлен также набор (видео пустоеслово).

Оператор "просмотрсписка"не позволяет просматривать компоненты наборов своего списка. Чтобы такой просмотр стал возможен, применяется оператор "сквозной-просмотр(a)". У него a - та же структура данных просмотра, что и у оператора "просмотрсписка". Интерфейс отличается от интерфейса последнего оператора лишь тем, что по нажатии на текущем пункте клавиши "курсор вправо", если этот пункт представлял собой набор либо вхождение в набор, происходит переход к просмотру этого набора. Если текущий пункт не являлся набором, то нажатие клавиши "курсор вправо"игнорируется. Для возвращения в просмотр внешнего набора нажимается любая из клавиш "курсор влево", "Esc", "пробел", "End". Обработчик команд оператора "просмотрсписка" может быть найден в разделе "Вспомогательные процедуры интерфейсов" "Процедура ПРОСМОТРСПИСКА" - "Цикл обращений к клавиатуре".

9.14 Блоки диалога

9.14.1 Диалоговые блоки

Для организации специализированных диалогов, в том числе с использованием логических структур данных, можно создавать диалоговые блоки. Эти блоки служат для отображения различной статистической информации, выбора параметров, прорисовки и редактирования термов и логических символов. Ввиду специфических функциональных особенностей, они реализованы внутренними средствами логической системы и не используют стандартных диалоговых средств. Диалоговый блок обычно занимает весь экран и состоит из группы прямоугольников, называемых элементами блока диалога. Каждый элемент блока диалога сопровождается группой информационных элементов, уточняющих списоб его использования. Активизация элемента диалога предпринимается нажатием левой клавиши мыши в его области либо нажатием клавиши клавиатуры, код которой является меткой элемента (обычно название такой клавиши указывается в прямоугольнике элемента либо в примыкающих к нему прямоугольниках). Предусмотрен простой интерфейс, позволяющий создавать и редактировать блоки диалога. При обращении к диалоговому блоку создается структура данных диалога - набор информационных элементов для заполнения его окон при первоначальной прорисовке и для сохранения введенной в процессе диалога информации.

Прежде всего, перечислим типы информационных элементов, характеризующих использование элемента блока диалога (каждый такой элемент - логический символ либо терм):

1. "метка(A)" логический символ A является меткой элемента.

2. "См(A)" прямоугольная рамка элемента прорисовывается в цвете A. В случае белого цвета (A = "шестнадцать") рамка не видна.

3. "формблок" при активизации элемента инициируется набор терма формульным редактором. По окончании набора в структуре данных диалога находится информационный элемент "формблок A B", где A - метка элемента диалога, у которого B заменяется на набранный терм.

4. "текстредактор" при активизации элемента инициируется вход в текстовый редактор. Начало B используемого отрезка буфера текстов при этом берется из информационного элемента "текстредактор A B"структуры данных диалога; A - метка элемента диалога.

5. "терм" при активизации элемента инициируется набор терма текстовым редактором. По окончании набора в структуре данных диалога находится информационный элемент "терм A B", где A - метка элемента диалога, у которого B заменяется на набранный терм.

6. "логсимвол" при активизации элемента инициируется набор логического символа текстовым редактором. По окончании набора в структуре данных диалога находится информационный элемент "логсимвол A B", где A - метка элемента диалога, у которого B заменяется на набранный логический символ.

7. "новыйсимвол" при активизации элемента инициируется набор логического символа текстовым редактором. Если символа с таким названием не было, то он вводится. По окончании набора в структуре данных диалога находится информационный элемент "новыйсимвол A B", где A - метка элемента диалога, у которого B заменяется на набранный логический символ.

8. "серия" при активизации элемента инициируется набор группы термов текстовым редактором. По окончании набора в структуре данных диалога находится информационный элемент "серия A B", где A - метка элемента диалога, у которого B заменяется на список набранных термов.

Шаблоны используемых в системе диалоговых блоков хранятся в 5-м информационном блоке. Для хранения отдельного шаблона используется узел A статьи какого-либо логического символа, так что пара (логический символ - номер узла) является ссылкой на этот шаблон. Из узла A по меткам "команды"и "нормистина" (первоначально - логический символ "блокдиалога", который впоследствии был переименован в название вспомогательной процедуры) имеется переход к указателю - списку B, из которого по меткам $1, 2, \ldots$ - переходы к указателям-спискам $C_1, C_2, \ldots$, представляющим элементы блока диалога. Из C_i по метке "область переход к логическому терминалу, хранящему четверку логических символов D_1, D_2, D_3, D_4. D_1, D_2 суть символьные номера столбца и строки верхнего левого угла прямоугольника, занимаемого на экране элементом; D_3, D_4 - символьные номера столбца и строки правого нижнего угла. По метке "слово"(может отсутствовать) из C_i - переход к текстовому терминалу, содержимое которого выводится в области элемента. По метке "нормистина"из C_i - переход к логическому терминалу, содержащему информационные элементы, характеризующие использование элемента диалога (типы их приведены выше).

В том же 5-м информационном блоке содержится оглавление шаблонов диалоговых блоков; переход к его корню от корневого каталога 5-го информационного блока осуществляется по меткам "оглавление", "нормистина"). Концевые логические терминалы данного оглавления хранят термы "прием($A_1 A_2$ нормистина)", где A_2 - номер узла статьи символа A_1, хранящего описание шаблона.

Для инициализации диалога по уже имеющемуся шаблону диалогового блока используется оператор "началодиалога(a b c)". У него b - номер узла статьи логического символа a, хранящего описание шаблона блока. Оператор осуществляет прорисовку диалогового блока с незаполненными (кроме текстов, введенных непосредственно при создании блока) элементами и возвращает структуру данных диалога c, используемую, в частности, как накопитель результатов диалога. Приведем перечень типов элементов в этой структуре данных:

1. (формблок A B) - B есть терм, прорисованный формульным редактором внутри элемента диалога с меткой A.
2. (терм A B) - B есть терм, прорисованный текстовым редактором внутри элемента с меткой A.
3. (логсимвол A B) - B есть логический символ, прорисованный внутри элемента с меткой A.
4. (текстредактор A B) - B есть исходная позиция в буфере текстов, начиная с которой размещается текст, вводимый внутри элемента с меткой A.
5. (прямоугольник A B) - B есть четверка символьных номеров, определяющая границы прямоугольника для элемента диалога с меткой A.
6. (новыйсимвол A B) - B есть новый логический символ, введенный в элементе с меткой A.
7. (серия A B) - B есть набор термов, введенных с помощью текстового редактора в элементе диалога с меткой A.

В тех случаях, когда параметр B изменяется при диалоге, он изначально инициализируется какой-либо переменной.

После того, как оператор "началодиалога"прорисовал бланк блока диалога, происходит заполнение необходимым содержимым прямоугольников его элементов. Для этого служит оператор "рис(a b c)". У него a - структура данных диалога; b - метка элемента диалога. Осуществляется прорисовка внутри прямоугольника этого элемента информации, определяемой указателем прорисовки c. Используются указатели прорисовки следующих типов:

1. (формблок A) - прорисовка терма A формульным редактором;
2. (терм A) - прорисовка терма A в скобочной записи;
3. (логсимвол A) - прорисовка логического символа A;
4. (текстредактор A B C) - прорисовка содержимого буфера текстов начиная с позиции A до позиции B. C - символьный номер строки, с которой начинается прорисовка (относительно верхней части прямоугольника элемента диалога);

5. "пустоеслово расчистка прямоугольника элемента;

6. (серия A) - прорисовка набора термов A в скобочной записи.

Для создания нового шаблона блока диалога служит оператор "новыйдиалог (a b)". Перед обращением к нему должен быть создан узел статьи некоторого логического символа a в 5-м информационном блоке, в котором предполагается сохранить шаблон, и из этого узла должен быть создан переход по метке "команды"к новому указателю - списку. b - номер данного узла. Оператор реализует интерфейс редактирования шаблона; обращение к нему возможно из редактора программ ЛОСа по нажатию Ctr-д. Тогда появляется оглавление блоков диалога. Чтобы изменить уже созданный блок диалога, нужно выбрать соответствующий концевой пункт данного оглавления и нажать "курсор вправо". Для создания нового блока диалога достаточно создать новый концевой пункт оглавления, нажать на нем "курсор вправо" и перейти к редактированию блока. При этом выбор a, b и создание узла выполняется автоматически. Чтобы получить ссылку (a, b) на просматриваемый блок диалога, нажимается клавиша "и".

Перечислим операции, применяемые при редактировании шаблона диалога.

Для создания нового элемента диалога нажимается "с"(кир.). Затем левая клавиша мыши нажимается сначала в позиции левого верхнего угла прямоугольника, затем - в позиции правого нижнего угла.

Для удаления элемента диалога нажимается "у", затем левая клавиша мыши нажимается в области удаляемого элемента.

Для ввода текста внутри прямоугольника элемента нажимается "т" и левая клавиша мыши нажимается в области элемента. Тогда появляется курсор текстового редактора, позволяющий ввести текст. По завершении редактирования ("Enter") текст сохраняется, иначе, по "Esc" - редактирование отменяется.

Для редактирования списка сопровождающих элемент диалога информационных элементов, уточняющих его использование, нажимается "п" и левая клавиша мыши нажимается в поле элемента. Тогда шаблон диалога исчезает, а в верхнем левом угла экрана появляется курсор текстового редактора (если уже имелись сопровождающие информационные элементы, то они прорисовываются). По завершении редактирование изображение шаблона восстанавливается.

Чтобы передвинуть рамку прямоугольника элемента, левая клавиша мыши нажимается на одной из его сторон, и далее удерживанием клавиши курсора можно добиться медленного перемещения этой стороны (для вертикальных сторон нажимаются клавиши "курсор вправо - влево", для горизонтальных - "курсор вверх- вниз"). Пока выбранная сторона не будет сдвинута хотя бы на один пиксел, прочие команды редактором игнорируются.

Для перемещения всего элемента нужно нажать левую клавишу мыши в области этого элемента, затем нажать "д", после чего использовать клавиши курсора.

Выход из редактирования - по "Esc"; при этом результаты редактирования сохраняются.

Глава 10

Общелогические приемы, реализованные на ЛОСе

Для обучения и оптимизации решателя, а тем более для извлечения из него информации о возможных подходах к самообучению, необходимо хотя бы общее представление о тех приемах, которые уже имеются, а также о выработанных стандартах технической обработки приемов. Данный том посвящен систематическому описанию накопленного в решателе многообразия приемов, которое одновременно может играть как роль справочника по решателю, так и роль учебника по программированию решателей. Не все предлагаемые технические решения бесспорны, и описание их во многих случаях следует рассматривать лишь как постановку задачи на поиск более точного объяснения. Не следует забывать, что предлагаемая версия решателя лишь открывает поле исследований по логической динамике, представляя собой что-то вроде чернового наброска, иллюстрирующего созданные инструментальные возможности. Предстоит очень большая работа по оптимизации имеющихся конструктивных решений, в процессе которой практически все содержимое, вероятно, будет полностью переделано. С другой стороны, созданные процедуры, все же, работают на многих тысячах задач, архитектура их подсказана обучающим материалом, и любые крупные изменения архитектуры должны быть хорошо с этим материалом согласованы. Степени свободы по варьированию решателя могут оказаться во многих случаях не такими уж и большими.

Описание базы приемов мы начнем с общелогических приемов, применение которых возможно в произвольных предметных областях. Подавляющее большинство приемов решателя реализованы на ГЕНОЛОГе. В принципе, возможно такое развитие этого языка, при котором все приемы были бы переведены на него. Однако, в настоящее время часть приемов реализованы непосредственно на ЛОСе. Главным образом, это общелогические приемы, где оказалось проще написать программу вручную, чем создавать новые компоненты ГЕНОЛОГа для почти одноразового применения. Ниже перечисляются основные реализованные на ЛОСе общелогические приемы, которые могут быть найдены в разделе "Приемы решателя" - "Общие приемы" оглавления программ. Технические подробности программ приемов опускаются. Эти программы легко находятся в концевых пунктах указанного раздела, так как названия пунктов примерно соответствуют подзаголовкам данного текста.

10.1 Приемы задач на описание

Попытка применения приема обычно начинается с усмотрения в задаче некоторого ключевого для данного приема логического символа. Такой принцип позволяет равномерно распределить приемы по различным логическим символам и обеспечить отсечение всех приемов, не связанных с контекстом задачи. В процессе обучения выяснилось, что число приемов, применение которых невозможно связать с каким-либо ключевым понятием, возникающим в задаче, крайне невелико. Эти приемы распределены по четырем типам задач - на описание, преобразование, доказательство и исследование. Попытка применения приема начинается здесь с рассмотрения логического символа, являющегося типом задачи, и обращения к программе этого символа.

В данном подразделе перечислим приемы, отнесенные к задачам на описание. Их программы собраны в ветви фрагментов программ логического символа "описать". Переходить к начальным точкам программ отдельных приемов следует через концевые пункты оглавления программ.

Учет о.д.з.

Для сокращения формулировок задач принимаются специальные соглашения об опускании условий, восстанавливаемых в процессе решения по умолчанию. Так, обычно опускаются сопровождающие условия, обеспечивающие осмысленность выражений и утверждений основных условий. Эти условия называются условиями на область допустимых значений; сокращенно - о.д.з. Чаще всего с условиями на о.д.з. приходится сталкиваться в задачах по элементарной алгебре. Например, если решается уравнение

$$\frac{2}{2+\sqrt{4-x^2}} - \frac{1}{2-\sqrt{4-x^2}} = \frac{1}{x},$$

то перед началом преобразований восстанавливаются условия, выражающие неотрицательность выражений под радикалами и отличие знаменателей от 0: $0 \le 4-x^2$, $2-\sqrt{4-x^2} \ne 0$, $x \ne 0$. К этой же категории восстанавливаемых по умолчанию условий относится и утверждение "x- число".

Чтобы решатель автоматически присоединял к условиям и посылкам задачи утверждения, задающие о.д.з., имеется специальный прием, точнее, по одному приему для задач каждого из четырех типов. Они устроены совершенно одинаково - проверяется наличие у задачи цели "одз"; если эта цель есть, то она удаляется (для предотвращения повторных применений приема), и происходит обращение к вспомогательной процедуре "одз", входным данным которой является текущая задача. Последняя процедура и преобразует задачу, пополняя условия и посылки. Заметим, что цель "одз" присутствует в целевых установках почти всех задач, создаваемых в задачнике. В случае задачи на доказательство, не имеющей никаких целей, символ "одз" регистрируется в комментариях, где он обрабатывается аналогичным образом.

Использовать цель "одз" во вспомогательных задачах, создаваемых приемами решателя, следует достаточно осторожно, так как ввод дополнительных условий на о.д.з., строго говоря, не является эквивалентным логическим переходом. Это, скорее, часть интерфейса системы. Однако, в некоторых случаях решатель может сам отбрасывать имеющиеся у него сопровождающие условия на о.д.з., руководствуясь теми или иными принципами восстановления их "по умолчанию". Например, это происходит при

выдаче ответа. Лишь для такого восстановления и нужно применять обращения к вспомогательным задачам, имеющим цель "одз".

Информацию, необходимую для определения условий на область допустимых значений, процедура "одз" получает при помощи справочника "одз". Этот справочник имеет своими входными данными следующие значения: x1 - задача; x2,x3,x4 - координата вхождения в нее первого символа выражения или утверждения A. Символом обращения к справочнику служит заголовок терма A. Результатом обращения является набор условий, при которых A "имеет смысл". В особых случаях приемы справочника реализованы на ЛОСе, но основная их часть задана на ГЕНОЛОГе. Теорема приема справочника "одз" имеет вид "длялюбого($x_1 \dots x_n$ эквивалентно(определено($f\ x_1\ \dots\ x_n$)и($B_1 \dots B_m$)))", где $B_1, \dots, B_m$ - условия, при которых $f(x_1 \dots x_m)$ является осмысленным утверждением либо выражением. Например, в случае дроби имеем теорему: "длялюбого($x_1\ x_2$ эквивалентно(определено(дробь $x_1\ x_2$)и(число(x_1)число(x_2)не(равно(x_2 0)))))".

Отметим, что на той же самой теореме приема основан еще один справочник - "типданных". Этот справочник определяет допустимый тип значений операндов операции либо отношения. С его помощью решатель автоматически присоединяет к задаче указания на типы значений переменных и сохраняет их в задачнике еще до обращения к решению задачи - сразу же по окончании ее ручного ввода. Поэтому, хотя справочник "одз" и указывает на типы значений операндов, они обычно уже имеются в наличии и процедурой "одз" к задаче не добавляются.

Приведем краткое описание действий, выполняемых процедурой "одз". Сначала просматриваются все условия и посылки задачи, и для каждого вхождения v символа операции предпринимается обращение к справочнику "одз". Если справочник определяет непустой список утверждений, выражающих условия на область допустимых значений операндов данной операции, то просматриваются не являющиеся посылками задачи элементы F этого списка. Отбрасываются утверждения без свободных переменных, а также утверждения вида $P(g(\dots))$, где P - название такого типа объектов, что операция g принимает только значения этого типа. Принимается решение, рассматривать ли F как новую посылку (условие) или размещать его внутри того же терма, к которому относится v. Последнее делается, если v расположено внутри квантора, описателя или дизъюнкции. Для квантора общности принимается решение об отнесении F к консеквенту либо антецеденту. По итогам просмотра условий и посылок задачи составляется список S четверок ($A_1\ A_2\ A_3\ A_4$), где A_1 - указатель посылки либо условия (0 или 1); A_2 - указатель регистрации сопровождающих условий в качестве новых термов задачи (0) либо вхождение точки регистрации их в старый терм; A_3 - вхождение того терма задачи, к которому относится сопровождаемое вхождение v; A_4 - список сопровождающих утверждений, регистрируемых согласно адресу A_1, A_2.

Список S играет роль плана регистрации в задаче новых утверждений, обеспечивающих сопровождение по о.д.з. Перед реализацией данного плана выполняется упрощение найденных утверждений. Составляющие список четверки ($A_1\ A_2\ A_3\ A_4$) последовательно просматриваются; сначала - относящиеся к посылкам, затем - к условиям. В каждом из случаев просмотр начинается с четверки, относящейся к утверждениям, регистрируемым непосредственно в списке посылок либо условий. Такое упорядочение возникло с целью подготовить к нужному моменту те сопровождающие утверждения, которые понадобятся для упрощения других сопровождающих утверждений.

Далее составляется список P тех посылок или условий задачи, которые образуют контекст, относительно которого рассматриваются все утверждения списка A_4.

Чтобы упрощать относительно P указанные утверждения, применяется нормализатор "нормодз". Этот нормализатор реализован на ГЕНОЛОГе; его можно найти в разделе "Логические приемы" - "Общие приемы" оглавления базы приемов. Здесь собраны самые простые упрощающие преобразования типичных условий на о.д.з. - например, преобразование отрицания равенства нулю произведения в конъюнкцию отрицаний равенства нулю сомножителей; замена отрицания равенства нулю степени на отрицание равенства нулю ее основания; замена нестрогого неравенства на строгое при наличии в посылках отрицания равенства; отбрасывание заведомо строго положительных либо строго отрицательных сомножителей в условиях положительности либо отрицательности (разумеется, с необходимой перестановкой операндов), и т.п. Необходимость упрощать сопровождающие условия до регистрации их в задаче, где они все равно были бы упрощены, продиктована соображениями оптимизации. Цепочка мелких преобразований, выполняемых при сканировании задачи, требует обычно в десятки раз больше времени, чем реализация этих же преобразований за один шаг какой-либо вспомогательной процедурой, применяемой вне сканирования задачи. В данном случае имеется хороший повод сгруппировать приемы, упрощающие условия на о.д.з., в одном пакете, так как неупрощенная версия условий обычно порождает длинные цепочки применений этих приемов, а число самих приемов невелико.

Сопровождающие утверждения, регистрируемые в заданной точке, сначала упрощаются относительно контекста P независимо друг от друга. После этого реализуется цикл повторных упрощений, где контекстом преобразований каждого сопровождающего утверждения служит список P, пополненный остальными сопровождающими утверждениями. Преобразования выполняются до тех пор, пока происходят какие-либо изменения. После этого измененный список регистрируется в качестве элемента A_4 текущей просматриваемой четверки.

По завершении упрощения сопровождающих утверждений предпринимается регистрация их в задаче. Сначала рассматриваются те утверждения, которые размещаются во внутренних логических контекстах уже имеющихся посылок либо условий, затем - утверждения, регистрируемые непосредственно в виде новых условий или посылок.

Программа процедуры "одз" может быть найдена в разделе оглавления программ "Приемы решателя" - "Общие процедуры, используемые в приемах" - "Процедура ОДЗ". Программная переменная x2 играет роль накопителя списка S. Программу приема, обеспечивающего учет о.д.з. для задач на описание, можно найти в разделе "Приемы решателя" - "Общие приемы" - "Задачи на описание" - "Учет о.д.з. в задаче на описание". Этот прием применяется на нулевом уровне сканирования.

Создание комментариев, определяющих схему сопровождения по о.д.з.

В процессе решения задачи могут возникать новые выражения, для которых условия на о.д.з. не будут непосредственно усматриваться из контекста так, как они усматриваются для исходных выражений после применения предыдущего приема. Вместе с тем, условия на о.д.з. часто проверяются различными приемами перед выполнением преобразований. Чтобы из-за невозможности быстро проверить эти условия не

заблокировать необходимые приемы, в решателе предусмотрена автоматическая коррекция сопровождения по о.д.з. Те немногие основные процедуры, которые фактически преобразуют задачу (например, процедура "заменавхождения"), определяют условия на о.д.з. для новых выражений, проверяют их и регистрируют в контексте задачи. Чтобы облегчить им работу, иногда в приеме содержатся прямые указания на вывод утверждений, сопровождающих по о.д.з. новые выражения. Однако, сопровождающие по о.д.з. утверждения нужно защищать от применения к ним каких-бы то ни было, даже самых простых, преобразований - ведь любое такое преобразование увеличивает "расстояние" между сопровождающим утверждением и сопровождаемым термом, что приводит к замедлению проверки по о.д.з. или даже к отказу при этой проверке. Для такой защиты предусмотрен специальный комментарий (сопровождение A). В случае задачи на исследование он является комментарием к списку посылок, иначе - к самой задаче. В наборе A данного комментария перечисляются всевозможные пары $(M\ t)$, где t - выражение, для которого необходимо сопровождение по о.д.з.; M - множество утверждений, из которых следует выполнение условий на о.д.з. для t. Предполагается, что эти утверждения встречаются в задаче в явном виде - как посылки, условия, конъюнктивные члены конъюнкций, антецеденты кванторных импликаций и т.п. До тех пор, пока утверждение используется как сопровождающее по о.д.з., попытки изменить или удалить его будут автоматически блокироваться. Впрочем, приему предоставляется возможность отменять такую блокировку, принимая на себя всю ответственность за последствия. Обычно это делается для преобразований, которые синхронно будут изменять и сопровождаемый, и сопровождающий термы.

Комментарий (сопровождение ...) создается в самом начале решения задачи, сразу после обращения к процедуре "одз". Для этого служат приемы, которые легко найти в пунктах "Ввод комментария СОПРОВОЖДЕНИЕ" разделов оглавления программ, относящихся к общим приемам задач на описание, преобразование, и т.д. Приемы обращаются к процедуре "сопровождтерм", которой передается ссылка на конкретную посылку либо условие P. Просматриваются вхождения в P неоднобуквенных подтермов t, и с помощью справочника "одз" находятся сопровождающие утверждения, отличные от указателей типов объектов. Если множество Q таких утверждений непусто, то предпринимается попытка усмотреть их истинность из контекста вхождения t с помощью проверочных операторов. Эта попытка определяет множество M утверждений контекста, следствием которых служит Q, и далее пара $(M\ t)$ регистрируется в комментарии (сопровождение ...).

Учет информации об исходных посылках, использованных при решении задачи

Прежде чем продолжить описание приемов, остановимся на одном важном элементе структуры данных задачи. Во многих случаях бывает нужно не только получить ответ, но и распознать, какие именно исходные посылки задачи фактически были использованы при его получении. Чтобы отслеживать использование исходных посылок, применяются комментарии (выводимо A), автоматически создаваемые и корректируемые практически всеми приемами решателя. Различаются два типа комментария (выводимо A). Во-первых, комментарий ко всей задаче (кроме задач на исследование). В этом случае A представляет собой список исходных посылок задачи, использованных при преобразованиях ее исходных условий к текущему виду.

Во-вторых, комментарий к отдельной посылке f. В этом случае A - список всех исходных посылок задачи, использованных при получении f. Таким образом, отсутствие комментария (выводимо ...) означает, что f - исходная посылка.

Если прием обращается к пакету продукций, то последний выдает не только ответ, но и список посылок, использованных при его получении. Этот список в случае проверочного оператора или синтезатора является значением одной из выходных переменных пакета; в случае нормализатора для накопления данного списка используется комментарий того же вида (выводимо A), что и выше. Отсутствие такого комментария может нарушить работу нормализатора, и его следует вводить даже в тех случаях, когда список A далее не используется.

Для создания и коррекции комментариев (выводимо ...) служит процедура "учетпосылок(x1 x2 x3 x4)". У нее x1 - текущая задача; x2 - список использованных приемом утверждений текущего контекста. Значение x4 либо равно 0, и тогда значением x3 является вхождение выведенной либо преобразованной приемом посылки, либо равно 1, и тогда x3 игнорируется. Выполняется создание либо коррекция комментария (выводимо ...): в первом случае - к посылке x3, во втором - к задаче x1. Процедура отбирает те элементы списка x2, которые являются посылками задачи x1. Затем, с помощью комментариев (выводимо ...) к этим посылкам, определяется множество M исходных посылок, на которые опирается текущий прием. После этого остается лишь пополнить старый комментарий (выводимо ...) элементами списка M либо создать новый такой комментарий. Обращаться к процедуре "учетпосылок" следует до того, как прием фактически изменил посылку x3. Заметим, что при использовании ГЕНОЛОГа обращения к этой процедуре создаются компилятором автоматически, без ввода в описание приема каких-либо специальных указателей.

Противоположные посылки

Если задача на описание имеет две посылки, одна из которых является отрицанием другой, то ее список посылок противоречив, и можно считать ответом на задачу логическую константу "истина", представленную в формате терма. Прием, который отслеживает указанную ситуацию, активизируется на нулевом уровне сканирования. Он просматривает все посылки, имеющие вес 0 (т.е. только что измененные либо занесенные посылки), и для каждой выясняет, содержится ли ее отрицание в списке посылок. Так как обоснованием ответа "истина" служат только найденные две противоположные посылки, то прием сбрасывает ранее имевшуюся в комментарии (выводимо ...) информацию об исходных посылках, использованных при решении, и переустанавливает ее по указанным двум посылкам.

Идентичные посылки

Если задача на описание имеет две идентичные посылки, то одна из них отбрасывается, а все комментарии отброшенной посылки добавляются к комментариям оставшейся. Прием применяется на уровне 0; он просматривает все посылки задачи, имеющие вес 0, и проверяет наличие идентичной посылки только для них. Это приводит к тому, что обычно отбрасывается та посылка, которая появилась позднее.

Условие, имеющееся в посылках

Если условие задачи, отличное от константы "истина", содержится в посылках, то оно отбрасывается, а в комментарии (выводимо ...) регистрируется равная ему посылка. Исключение составляют только дизъюнктивные условия, сопровождаемые комментарием "разборслучаев", т.е. введенные специально для рассмотрения подслучаев. Заметим, что невозможно удалить вообще все условия задачи на описание: когда остается только одно условие, то вместо удаления предпринимается замена его на константу "истина".

Идентичные условия

При усмотрении совпадающих условий задачи на описание одно из них отбрасывается вместе со своими комментариями. Схема поиска совпадения та же, что для посылок - просматриваются все условия F веса 0, и если имеется другое условие, равное F, то удаляется F. Уровень срабатывания приема нулевой.

Посылка является частью условия

Если посылка A имеет вхождение в условие B, причем это вхождение не расположено в области действия кванторов либо описателей по свободным переменным утверждения A, то данное вхождение заменяется в условии B на константу "истина". Прием просматривает все посылки веса 0, и для каждой такой посылки - все условия. Уровень его срабатывания равен 0.

Для изменения условия прием использует процедуру "заменавхождения". Эта процедура относится к числу немногих процедур, применяемых приемами для преобразования задачи. Коротко о ней рассказывалось в первой книге, однако для развития решателя необходимо более детальное представление о ее устройстве. Мы рассмотрим эту процедуру в следующей главе, наряду с рядом других общих процедур, используемых приемами. Локализация всего многообразия действий решателя по изменению контекста внутри немногих процедур позволяет вводить дополнительные механизмы управления и сопровождения. В отличие от логики принятия решений, рассредоточенной по конкретным приемам, эти механизмы позволяют отследить или заблокировать действия решателя из каких-либо совсем общих соображений, а также автоматически дополнить основные действия приема рядом сопутствующих действий. Они существенно усложняют программы "преобразующих" процедур, составляя главную часть их объема и трудоемкости.

Упрощение посылок

Если задача имеет невырожденные посылки, которые до этого не анализировались, то имеет смысл в начале решения попытаться упростить эти посылки, привести их к стандартному для решателя виду. Таким образом может сделаться явной та или иная замаскированная вначале информация.

Прием, предпринимающий упрощение посылок задачи на описание, активизируется при текущем уровне 2 либо 3. Сначала он рассматривает целевую установку. Если из нее ясно, что посылки уже были ранее упрощены, или что их анализ целесообразно отложить до обращения к подзадаче, то дальнейшие действия блокируются.

Например, не упрощаются посылки задачи, имеющей цель "редакция", так как в ней происходит обработка найденного ответа, а посылки должны были анализироваться до этого. Затем просматриваются все посылки A веса 2. Если задача не имеет цели "прямойответ", то из просмотра исключаются посылки с квантором существования в заголовке, так как они будут преобразованы путем ввода вспомогательных объектов. Не рассматриваются также элементарные посылки, имеющие лишь однобуквенные подвыражения. Попытка упрощения посылки выполняется однократно - уже рассмотренные посылки снабжаются комментарием "упростить". Для упрощения используется обращение к вспомогательной задаче на преобразование. Условием этой задачи служит текущая посылка A, а посылками - все оставшиеся посылки задачи на описание. Если результат упрощения B отличен от A, то процедура "заменавхождения" заменяет посылку A на B.

Анализ объединенного списка условий и посылок

Ранее уже говорилось, что многие задачи на описание решаются с помощью вывода следствий из объединенного списка посылок и условий. Этот объединенный список хранится в задаче на исследование A, связываемой с задачей на описание Z и называемой ее блоком анализа. Он представляет собой список посылок задачи A. Обычно новая задача на описание не имеет блока анализа. Та позиция ее структуры данных, которая хранит ссылку на блок анализа (последний разряд набора), вначале заполняется либо пустым словом, либо, если блок анализа нежелательно создавать вообще, нулем. В процессе рассмотрения задачи решатель создает блок анализа самостоятельно. Он может делать несколько попыток развития данного блока - с возрастающей последовательностью уровней обращения, привлекая каждый следующий раз все более и более мощные средства.

Для первоначального ввода блока анализа и организации обращений к нему служит один и тот же прием. Если задача на описание Z имеет цель "исследовать", означающую, что ответом на нее являются все представляющие интерес утверждения, выведенные в блоке анализа, то уровень активизации приема равен 1. В противном случае он активизируется не ранее уровня 4.

Прием создает вспомогательную задачу на исследование A, регистрируя ее как блок анализа задачи Z, либо берет уже созданный ранее блок анализа A. Он обращается к решению задачи A, определяя максимальный уровень обращения в зависимости от контекста. От контекста же зависит уровень срабатывания самого приема. Если при рассмотрении блока анализа оказываются найдены следствия, требующие переосмысления на уровне внешней задачи Z, то они передаются в список условий последней, и работа с задачей A обрывается. Обрыв решения задачи A и возвращение к задаче Z инициируется оператором ЛОСа "ответ(A)".

Опишем действия приема несколько более подробно. После проверки того, что последний разряд набора, представляющего текущую задачу на описание Z, отличен от 0, а цели и комментарии этой задачи не отменяют рассмотрения блока анализа, составляется список всех логических символов, входящих в условия задачи Z. По нему определяется список S разделов, к которым относятся данные символы. Здесь используется процедура "разделы", обращающаяся к справочнику "раздел". Последний справочник, а также справочник "содержание" регулярно пополняются по мере ввода новых понятий при обучении решателя.

Знание разделов, к которым относится задача Z, позволяет определить как величину текущего уровня сканирования, на котором следует обращаться к блоку анализа, так и максимальный уровень обращения к нему. Для этого служит справочник "вход". Ему сообщаются название раздела, задача Z и текущий уровень сканирования. Если прием справочника находит целесообразным рассмотрение блока анализа, то выдает ненулевую величину максимального уровня обращения. Находится максимум M таких величин (в программе приема - значение x7). При наличии у Z целей "известно" либо "исследовать", которые непосредственно указывают на использование блока анализа как главного средства решения задачи, значение M сразу полагается наибольшим - равным 16.

Если блок анализа A еще не создан, то он создается как задача на исследование, имеющая своим списком посылок объединение списков условий и посылок задачи Z. Веса всех посылок полагаются равными 0; условия f задачи Z сопровождаются в этом списке комментариями (условие f (f)). Комментарий (условие g K) к посылке f впоследствии будет отслеживать возможные изменения этой посылки, указывая, что условие g задачи Z является следствием f и прочих посылок задачи A, перечисляемых наряду с f в списке K. В качестве общих комментариев к списку посылок задачи A вводятся: (описать B), где список B перечисляет все условия задачи Z, перенесенные в A; (вход C), где C - обратная ссылка из A на задачу Z, а также "обращение". Заметим, что в качестве ссылки C здесь берется не сама задача Z, а вхождение первого разряда представляющего ее набора. Это - мера предосторожности против возникновения циклических ссылок: ведь из Z имеется ссылка непосредственно на A. Циклические ссылки в зоне задач недопустимы, так как они могут приводить к зависанию процедуры ЛОСа, сравнивающей два набора. Комментарий "обращение" инициирует выдачу на экран в режиме просмотра решения информации об обращении к блоку анализа A. В список целей задачи A переносятся из списка целей задачи Z набор (неизвестные ...), перечисляющий неизвестные, а также ряд других целей, уточняющих режим вывода следствий.

Если задача Z имела цель (известно $a_1 \ldots a_n$), перечисляющую переменные $a_1, \ldots, a_n$, через которые должен быть выражен ответ, то она не переносится в задачу A; вместо нее задаче A передается цель "известно". В этой ситуации все переменные посылок задачи Z, не упомянутые среди $a_1, \ldots, a_n$, становятся дополнительными неизвестными задачи A. С другой стороны, неизвестные задачи Z, которые уже оказались неявно выражены ее условиями через $a_1, \ldots, a_n$, исключаются из неизвестных задачи A.

Если задача на описание Z представляет собой подслучай, возникший ранее при рассмотрении некоторой дизъюнктивной посылки, то задача A дополняется целью "контроль". Эта цель блокирует поспешную выдачу ответа, найденного в A для рассматриваемого подслучая, так как сам подслучай может оказаться невозможным. Лишь по достижении уровня 3, когда попытки усмотрения противоречивости списка посылок задачи A окажутся безуспешными, ответ будет выдан.

Задаче A передается ряд специальных комментариев задачи Z, например, комментарий (геомредактор ...), определяющий текущий чертеж.

По завершении пополнения структуры данных задачи A, она регистрируется в качестве блока анализа задачи Z, и предпринимается обращение к ее решению до максимального уровня M. Окончание рассмотрения задачи A возможно по двум причинам - либо по исчерпании допустимого уровня, либо при обрыве, вызванном действия-

ми какого-либо приема, обратившегося к оператору ЛОСа "ответ(A)". В последнем случае тот же прием модифицирует и задачу Z, так что она уже содержит в себе найденный при рассмотрении A результат. Поэтому далее возобновляется сканирование задачи Z; если она была изменена, то текущий уровень сканирования предварительно понижается до 0. Чтобы предотвратить повторные попытки обращения к блоку анализа с одним и тем же максимальным уровнем M, используются комментарии (вход M) к задаче Z.

Если на момент принятия решения о рассмотрения блока анализа A этот блок уже был создан, то предпринимается пополнение списка посылок задачи A теми условиями задачи Z которые ранее не были перенесены в A. Далее, как и выше, происходит обращение к задаче A с максимальным уровнем M.

Разбиение условий на независимые группы

Если множество содержащих неизвестные условий задачи на описание Z разбивается на такие подгруппы $M_1, \ldots, M_n$, что утверждения из разных подгрупп не имеют общей неизвестной, то можно перейти к решению независимых задач $Z_1, \ldots, Z_n$ со списками условий $M_1, \ldots, M_n$. К каждому из этих списков добавляются также все условия задачи Z, вообще не содержащие неизвестных. Ответ на задачу Z будет представлять собой объединение ответов на указанные задачи. Прием, выполняющий данные действия, нетрудно найти по оглавлению программ. Его программа начинается с контрольной точки "прием(13)" в программе символа "описать". Уровень срабатывания приема равен 2.

Во-первых, предпринимается проверка целесообразности разбиения списка условий на независимые группы. Например, оно не выполняется для задач с целью "редакция" или "известно ...", а также для задач, имеющих комментарий (контекст ...). Последний комментарий возникает при исключении части неизвестных задачи, для которых было найдено явное их описание через другие неизвестные. При редактировании ответа в каком-либо отдельном подслучае из комментария извлекаются утверждения, связывающие ранее исключенные неизвестные с оставшимися. Эти утверждения присоединяются к прочим условиям задачи, и упрощение относится ко всему списку утверждений, определяющих подслучай. Таким образом, при последующей склейке подслучаев в общий ответ не нужно специально учитывать ранее исключенные неизвестные. Разбиение на компоненты связности нарушило бы данный режим, из-за чего оно и блокируется.

Далее определяется разбиение списка содержащих неизвестные условий задачи Z на компоненты связности M_i по зависимости от общей неизвестной. Находится также список M_0 всех условий, не содержащих неизвестных. Составляется список S пар ($M_i \cup M_0$ - множество неизвестных, входящих в утверждения из M_i). В программе приема этот список присвоен переменной x6. Если число компонент более одной, то они последовательно просматриваются, и для каждой из них создается задача Z_i. Комментарии к условиям задачи Z_i, а также комментарии ко всей этой задаче по умолчанию берутся из Z. При необходимости они могут быть скорректированы с помощью справочника "копияфайла". Это же относится и к целям задачи, где используется другой справочник - "сокращнеизв". Для отображения (при трассировке) на экране факта обращения к задаче Z_i, она снабжается комментарием "обращение". Предпринимается обращение к решению задачи Z_i, причем максимальный уровень -

тот же, что у задачи Z. Если получен отказ, то и на задачу Z выдается отказ. Иначе - список конъюнктивных членов ответа задачи Z_i и список использованных при получении этого ответа исходных посылок задачи Z регистрируются в специальном накопителе (R_1, R_2). В программе приема роль этого накопителя играет переменная x7.

После решения всех задач Z_i проверяется наличие цели "упростить" у задачи Z. Если она есть, то создается вспомогательная задача на описание Z', условия которой суть все элементы накопителя R_1, а цели - дополненные символом "редакция" цели задачи Z. Эта задача решается; в комментарии (выводимо ...) задачи Z регистрируются все исходные посылки, использованные в процессе решения $Z_1, \ldots, Z_n, Z'$, и выдается ответ задачи Z'. Если же цели "упростить" у задачи Z не было, то в качестве ответа выдается конъюнкция утверждений списка R_1.

Непосредственный подбор значений неизвестных

Если в задаче на описание достаточно получить лишь пример значений неизвестных, удовлетворяющих ее условиям, то можно проверить, не содержат ли посылки группу утверждений, непосредственно дающих такой пример, т.е. получающихся из условий подстановкой вместо неизвестных каких-то конкретных выражений. При этом следует учитывать, что искомые посылки могут идентифицироваться с условиями лишь после применения к ним каких-либо простых преобразований, например, перестановки операндов в коммутативных операциях и отношениях. Иногда достаточно лишь частичной идентификации условий с посылками, позволяющей однозначно определить предполагаемые значения неизвестных, после чего легко усматривается истинность оставшихся условий.

Такого рода сравнение условий с посылками нужно проводить быстро, не доводя дело до сколь-нибудь значительного перебора, так как случаи, в которых оно вообще оказывается результативным, достаточно редки. С другой стороны, задачи, где ответ все же усматривается указанным способом, относятся к категории "очевидных", и данный прием должен применяться на малых уровнях.

В решателе имеется несколько приемов, выполняющих попытку непосредственного сопоставления условий с посылками для усмотрения ответа. Прежде всего, эта попытка предпринимается на уровнях 1 и 3. Проверяется, что все условия задачи суть элементарные утверждения, и для подбора значений неизвестных используется процедура "подборзначений". Она находит условие, заголовок которого (с учетом внешнего отрицания) встречается в посылках наименьшее число раз. Последовательно рассматриваются варианты идентификации этого условия с посылками; для идентификации оставшихся условий предпринимаются рекурсивные обращения к той же самой процедуре. Если удалось подобрать значения неизвестных, то перед выдачей ответа проверяется наличие комментария (контекст ...). Он хранит ту часть ответа, которая связана с ранее исключенными (выраженными через оставшиеся) неизвестными. Эта часть присоединяется к текущему ответу, причем в нее подставляются значения найденных неизвестных.

На уровне 4 предпринимается еще одна попытка подбора значений неизвестных. В ней учитывается возможность наличия условий без неизвестных. Для таких условий выполняются попытки усмотрения их истинности с помощью вспомогательных задач на доказательство, решаемых с максимальным уровнем 4. В случае успеха используется процедура "подборнеизвестных", идентифицирующая с посылками остальные

условия. Чтобы все эти действия выполнялись быстро, введен ограничитель трудоемкости, по исчерпании которого применение приема обрываются.

Еще одна попытка подбора, по существу совпадающая с предпринимавшейся на уровнях 1 и 3, реализуется на уровне 5.

Наконец, упомянем еще один прием, несколько отличающийся по своему назначению от указанных выше. Если задача на описание имеет цели "пример", "полный", причем все ее неизвестные - несущественные, т.е. содержатся в цели (параметры ...), то она фактически эквивалентна задаче на доказательство существования. Предположим, что удалось подобрать такие значения ее неизвестных, для которых все условия, кроме одного, отобразились в некоторые посылки. Обозначим A результат подстановки в оставшееся условие найденных значений неизвестных; пусть этот результат не содержит неизвестных. Тогда можно решать задачу путем разбора случаев, присоединяя к ее посылкам утверждение $A \vee \neg A$. Первый подслучай очевиден - необходимые значения неизвестных существуют. Поэтому остается только рассмотрение второго подслучая. Однако, технически проще избежать явного рассмотрения случаев, сразу же добавляя к списку посылок задачи утверждение $\neg A$. Конечно, это повлияет на итоговые значения неизвестных - они должны были бы определяться условными выражениями, в зависимости от истинности A. Но так как неизвестные несущественны, данное обстоятельство не играет никакой роли. Прием, выполняющий описанный "вывод" новой посылки $\neg A$, активизируется на уровне 4. Для подбора неизвестных он использует процедуру "частичнподбор".

Выражение одной неизвестной через другие

Если задача на описание имеет более одной неизвестной, то можно сначала попытаться, выделив какую-то одну неизвестную x и временно считая все остальные неизвестные известными, начать решение относительно x. Получив ответ и исключив из него ту часть, которая определяет допустимые значения x, далее можно вернуться к решению задачи относительно оставшихся неизвестных. По завершении последнего процесса остается объединить ответы и упростить результат. Эта схема часто применяется в математике; классический пример - решение систем линейных уравнений методом Гаусса. Однако, даже в элементарной алгебре она не является универсальной. Легко привести примеры, в которых после выражения одной неизвестной через другие система уравнений становится чрезмерно громоздкой, и задача заводится в тупик. При обучении решателя прием, реализующий данную схему, накопил в себе настолько существенные ограничения, что в некоторых разделах оказался почти отключенным. Например, в элементарной алгебре он применяется лишь для случаев, когда число уравнений системы меньше числа неизвестных. Разумеется, если система имеет одно или несколько линейных уравнений, то решатель попытается использовать их для исключения неизвестных. Однако, эти действия выполняются не общим приемом, а специальными реализованными на ГЕНОЛОГе приемами, относящимися именно к линейным уравнениям. Даже они имеют множество дополнительных ограничений. После того, как в задаче появляется уравнение вида $x = t$, явно выражающее неизвестную x через прочие неизвестные, неизвестная x исключается приемом, закрепленным за символом "равно" (см. ниже).

Впрочем, указанный выше прием неплохо работает в других разделах; например, при решении систем уравнений для множеств.

Перейдем к более подробному описанию действий приема. Программу его можно найти в пункте "Решение задачи на описание с несколькими неизвестными путем выражения одной из неизвестных через остальные" оглавления программ.

Сначала предпринимается анализ целевой установки приема. Проверяется отсутствие цели "редакция", означающей, что задача уже решена и выполняется редактирование ответа; отсутствие цели "пример"; наличие цели "полный"; отсутствие комментария "блокнеизвестных", блокирующего повторную попытку применения приема; отсутствие дизъюнктивных условий, для которых будут разбираться подслучаи, и т.п.

Если из целевой установки и общего вида условий не усматривается нецелесообразность применения приема, то находится список S всех разделов, к которым относятся встречающиеся в задаче логические символы. Просматриваются входящие в S символы f, и для окончательного принятия решения о применении приема выполняются обращения к справочнику "описать" на этих символах. Здесь же уточняется уровень применения приема - 2 либо 3. Если попытка признается целесообразной, то справочник выдает список всех неизвестных x, для которых ее следует выполнять. Неизвестные в этом списке упорядочены по убыванию приоритета. Берется первая неизвестная x, и для нее создается вспомогательная задача Z'. Она получена из текущей задачи Z преобразованием целей, соответствующим сохранению единственной неизвестной x. Такое преобразование осуществляется с помощью справочника "сокращнеизв". Задача Z' снабжается комментарием (сокращнеизв Z^* X), где Z^* - исходная задача со многими неизвестными, совпадающая с Z либо (если уже выполнялось исключение неизвестных) извлекаемая из комментария (контекст Z^* ...); X - список неизвестных, временно рассматриваемых как известные. Хотя вспомогательная задача Z' изначально ориентирована на разрешение условий только относительно x, в процессе ее решения комментарий (сокращнеизв ...) будет учтен таким образом, чтобы получить сразу ответ на задачу Z. Как именно это происходит, опишем ниже. Здесь же заметим, что прием обращается к решению задачи Z'. Если на нее получается ответ, отличный от символа "отказ", то он выдается как ответ на Z. Иначе - вводится комментарий "блокнеизвестных" и выполняется откат к просмотру следующего символа f.

В процессе решения задачи Z' может происходить разбор случаев, и комментарий (сокращнеизв ...) будет передаваться задачам, соответствующим отдельным подслучаям. После получения окончательного ответа для x в некоторой такой задаче Z'', решатель обращается к процедуре "редакторответа", выполняющей завершающую обработку ответа. Здесь и реализуется учет комментария (сокращнеизв Z^* X). Программу, выполняющую этот учет, можно найти в пункте "Приемы решателя" - "Общие процедуры, используемые в приемах" - "Процедура РЕДАКТОРОТВЕТА" - "Учет неизвестных, которые временно рассматривались как известные" оглавления программ. Ответ представляется в виде $A(x)$ & B, где B - все утверждения без неизвестной x. Создается вспомогательная задача Z''' для разрешения относительно оставшихся неизвестных X условий, полученных добавлением к B условия существования $\exists_x A(x)$, если оно не очевидно. В программе эта задача присваивается переменной x10. Информация о фрагменте ответа $A(x)$ передается задаче Z'' через комментарий (контекст ...), который может также содержать информацию о других ранее исключенных неизвестных. После этого предпринимается решение задачи Z'''. Если на нее получен "отказ", то выдается отказ на Z''. Так как при редактировании ответов отдельных подслучаев, возникающих в Z''', будет выполняться присоедине-

ние и упрощение фрагментов ответа, сохраненых в комментарии (контекст ...), то получение ответа R на Z''' одновременно будет означать получение ответа и на Z (разумеется, лишь для текущего подслучая, определяемого задачей Z''). В этой ситуации будет предпринято перенесение тех комментариев задачи Z''', которые представляют ценность для внешних задач, в задачу Z'', и выдан ответ R задачи Z''. Обработка комментариев выполняется здесь справочником "преобразование".

Попытка исключения несущественной неизвестной

У задачи на описание, имеющей цель (параметры $x_1 \dots x_n$), неизвестные $x_1, \dots x_n$ являются несущественными - они не обязаны входить в ответ. Если усматривается, что существование значения какой-либо несущественной неизвестной x, для которого истинны все содержащие ее условия $A_1, \dots, A_n$, является следствием прочих условий, то данная неизвестная и все условия на нее могут быть из задачи исключены. Несущественную неизвестную x можно исключить и другим путем - если усмотреть, что утверждение $\exists_x(A_1 \ \& \ \dots \ \& \ A_n)$ эквивалентно какому-то простому бесквантворному утверждению B и заменить $A_1, \dots, A_n$ на B. В простейших случаях такое усмотрение выполняется приемами, ориентированными на конкретные ситуации. Например, если все содержащие несущественную неизвестную x условия суть $a < x$, $x < b$, "x-число", то они заменяются на $a < b$.

Если задача не требует получения полного ответа - либо имеет цель "пример", либо не имеет цели "полный", то предусмотрен общий прием исключения несущественной неизвестной (см. пункт "Попытка исключения несущественной неизвестной путем решения вспомогательной задачи на описание с единственной неизвестной"). Он проверяет, что число неизвестных задачи не менее 2, и выбирает какую-либо несущественную неизвестную x. Для блокировки повторных рассмотрений проверяется отсутствие комментария (свертка x), который затем сразу же вводится. Создается вспомогательная задача Z', полученная из текущей задачи Z перенесением в посылки всех не содержащих x условий и выбором x в качестве единственной неизвестной, причем несущественной. Она решается с умеренным ограничением на допустимую трудоемкость. Если получен ответ B, отличный от символа "отказ", то проверяется отсутствие x в этом ответе и происходит регистрация всех его конъюнктивных членов в условиях задачи Z вместо всех старых условий, содержавших x. Переменная x исключается из неизвестных, и веса условий задачи Z полагаются равными 0.

Упрощение выражений при редактировании ответа

После того, как усмотрен ответ задачи на описание, обычно применяется процедура "редакторответа", выполняющая упрощение ответа в соответствии с целевой установкой задачи. Подробнее эта процедура будет рассмотрена позднее. Пока же отметим, что для упрощения ответа используется вспомогательная задача на описание, имеющая цель "редакция". В ней ответ определяется уже без обращения к процедуре "редакторответа", по исчерпании отведенных для решения средств.

При решении задачи, имеющей цель "редакция", предпринимается попытка упростить подвыражения, не содержащие неизвестных. Это важно, так как после перехода к утверждениям, образующим требуемое описание значений неизвестных, решение задачи сразу прекращается, и относительно известных параметров указанные утверждения могут оказаться совершенно необработанными. Упрощение подвыражения,

содержащего только известные параметры, выполняется в два этапа: сначала (на текущем уровне 2) с помощью вспомогательных задач на преобразование; затем (на уровне 3) с помощью процедуры "свертка", обеспечивающей переход к возможно более короткой записи. Эта процедура обращается к специальным пакетным нормализаторам, заголовки которых получаются добавлением префикса "упрощ" к названию корневой операции. Они легко находятся по оглавлению базы приемов. В качестве примера приведем прием пакета "упрощумножение", преобразующий произведение $a^c b^c$ к виду $(ab)^c$. Заметим, что этот переход нужен лишь для сжатия ответа; в процессе решения задачи обычно применяется обратное преобразование, позволяющее получать одночлены стандартного вида.

К программе приема, выполняющего упрощение известных подвыражений при редактировании ответа, можно перейти через оглавление программ. Уровни срабатывания приема - 2 и 3. После анализа целевой установки задачи (в частности, проверки наличия цели "редакция") принимается решение о применении приема. Просматриваются условия F, не используемые для сопровождения по о.д.з. и не имеющие комментария (параметры U), указывающего, что на текущем уровне U известные подвыражения F уже упрощались. Проверяется отсутствие более короткого условия, подходящего для упрощения известных подвыражений и еще не помеченного данным комментарием. Указанный комментарий к F вводится. Составляется список S максимальных известных подвыражений утверждения F. Так как в ответе задачи с несколькими неизвестными встречаются утверждения, выражающие одну неизвестную через другие, локально как бы известные, то при составлении списка S это учитывается, и в него могут попасть выражения с "побочными" для F неизвестными. Если встречается ассоциативно-коммутативная операция, то в S отбирается ее фрагмент, образованный всеми известными операндами. Из S исключаются однобуквенные выражения и десятичные числа. Для уровня 2 отбрасываются также подвыражения ранее упрощенных выражений A. Последние распознаются по комментариям (упрощение A). В программе приема список S присвоен переменной x9. Далее выполняется просмотр элементов t списка S. Если текущий уровень равен 2, то организуется обращение к вспомогательной задаче на упрощение t относительно контекста утверждения F. Для результата T создается комментарий (упрощение T). Если T отличается от t, то выполняется замена t на T. Она относится не только к F, но сразу ко всем содержащим t термам задачи. После выполнения замены просмотр списка S продолжается. Действия в случае уровня 3 аналогичны, но вместо обращения к вспомогательной задаче предпринимается обращение к процедуре "свертка". По окончании просмотра списка S - откат к началу цикла сканирования задачи с обнулением текущего уровня (оператор "пересмотр").

Выдача ответа

В большинстве случаев ответ задачи на описание выдается приемами ГЕНОЛОГа, усматривающими в списке условий явное описание для единственной неизвестной. Перечислим те особые случаи, для которых предусмотрен общий прием выдачи ответа, реализованный на ЛОСе. Перейти к их программам можно через раздел "Приемы решателя" - "Общие приемы" - "Задачи на описание" - "Выдача ответа" оглавления программ.

1. Если задача имеет цель "редакция" и достигнут ее максимальный уровень, то проверяется, не противоречит ли целям задачи вид ее списка условий. Для

этого используется справочник "редакция", обращения к которому происходят на символах - заголовках целей. Если противоречия не усмотрено, в качестве ответа выдается конъюнкция условий задачи.

2. Если задача имеет единственное условие $P(x)$, где x - неизвестная, а P - название одного из основных типов объектов ("число", "множество", "функция", и т.п.), то на текущем уровне 0 выдается ответ. Если задача не имеет цели "редакция", то выдача ответа осуществляется процедурой "редакторответа", иначе - ответ $P(x)$ выдается непосредственно. При наличии цели "пример" прием блокируется, так как в этом случае нужно указать конкретное значение x. Обращение к процедуре "редакторответа" при отсутствии цели "редакция" необходимо не для упрощения и без того "минимального" ответа, а для учета информации о внешем контексте, которая, возможно, потребует присоединения к ответу утверждений, характеризующих ранее исключенные неизвестные.

3. Если задача имеет цель "стоп", то ответ на нее выдается на текущем уровне 0 и представляет собой конъюнкцию условий. Такая ситуация складывается, если задача нужна только для получения списка утверждений, пополненного ограничениями на о.д.з.

4. Если все условия задачи оказались не содержащими неизвестных, то в качестве ответа выдается конъюнкция этих условий, возможно, предварительно упрощенная. Уровень срабатывания приема равен 0. Проверяется, что задача не имеет целей "редакция" и "независит ..." (последний случай обрабатывается другим приемом). Отбрасывается случай задачи без неизвестных, имеющей цель "явное", если только ее список условий не состоит из единственной логической константы "истина" или "ложь". Если задача имеет цели "полный", "пример", то предпринимается попытка усмотреть истинность всех ее условий, и при неудаче выдача ответа блокируется. В прочих случаях, в зависимости от наличия цели "упростить", либо происходит обращение к процедуре "редакторответа", либо сразу выдается конъюнкция условий. Особо анализируется случай наличия в условиях константы "ложь" - тогда она и выдается в качестве ответа.

5. Если задача имеет одну из целей "мощность", "исследовать", "попыткаспуска", то ответ на нее выдается по достижении максимального уровня. Для этого используется процедура "редакторответа".

6. Если задача имеет цель "свобоперанд", то ответ на нее выдается сразу же, как только в ее условиях пропадают кванторы. Прием, отслеживающий это событие, применяется на уровне 0. Ответом служит конъюнкция условий.

7. Если задача имеет комментарий "ответ", то сразу же (на уровне 0) в качестве ответа выдается конъюнкция условий. Комментарий обычно используется для передачи уже найденного ответа из вспомогательной задачи во внешнюю задачу. Для этого вспомогательная задача должна изменить список условий внешней задачи и передать ей указанный комментарий.

8. Если задача имеет цель "перечисление", то она решается в режиме перечисления частичных ответов, накапливаемых в наборе A комментария (ответзадачи A). Регистрация этих ответов выполняется процедурами "попыткаспуска", "попыткапараметризации", используемых приемами при попытках решить

задачу в частных случаях. По достижении максимального уровня составляется дизъюнкция утверждений накопителя A, на которую заменяются условия задачи и которая выдается в качестве ответа. При пустом списке A выдается отказ.

9. Для задачи, имеющей только цели "независит" и "прямойответ", ответ выдается по достижении максимального уровня, если ее условия не содержат запрещенных переменных. Аналогичные действия выполняются для задачи, имеющей цели "длялюбого" и "независит", однако уже на нулевом текущем уровне.

10. Предусмотрен специальный прием для выдачи ответа на дифференциальные уравнения, если производные неизвестных функций исключены, но явно разрешить полученные соотношения относительно этих функций не удалось.

11. Совсем особый случай - задачи с целью "вычисление". Такие задачи предназначены для создания программы ЛОСа, находящей значения неизвестных по заданным значениям известных параметров. Сначала задача решается как обычная задача на описание, причем в списке ее условий складывается схема вычислений, понятная компилятору ГЕНОЛОГа. Она представляет собой некоторую совокупность утверждений, последовательно определяющих новые значения через ранее определенные. В этом отношении она ничем не отличается от списка антецедентов обрабатываемой компилятором ГЕНОЛОГа теоремы приема. Поэтому, по завершении решения "обычной" задачи на описание, остается лишь обратиться к компилятору для получения исходной программы. Подробнее все эти действия будут рассматриваться в главе, посвященной вычислительным задачам. Пока ограничимся указанием на пункт "Компиляция ответа задачи на вычисление" рассматриваемого подраздела оглавления программ, через который можно выйти на начало программы приема, обращающегося к компилятору ГЕНОЛОГа по завершении подготовки схемы вычислений. Прием активизируется на максимальном уровне, т.е. по исчерпании средств, отведенных для обработки указанной схемы. Собственно обращение к компилятору выполняется процедурой "вычисление", которая регистрирует созданную программу в блоке программ как программу справочника "вычисление", закрепленную за некоторым (выбранным наугад) ключевым логическим символом s. Если задача не имела известных параметров, то вычисления выполняются немедленно, путем обращения к справочнику "вычисление" на символе s. По итогам формируется список равенств, указывающих найденные значения неизвестных. Условия задачи заменяются на эти равенства, и конъюнкция их выдается как ответ. Если же задача имела известные параметры, то ответом на нее служит однобуквенный терм "программа". Однако, перед выдачей данного ответа на экран произойдет обращение к интерфейсу, предоставляющему возможность вводить нужные значения параметров и вычислять для них значения неизвестных. Ответ будет прорисован лишь после выхода из этого интерфейса. Заметим, что в обоих случаях созданная программа справочника "вычисление" сохраняется, и дальнейшие обращения к ней можно выполнять без повторного запуска решения задачи.

Задача на проверку истинности

Задача на установление истинности либо ложности утверждения оформляется как задача на описание, имеющая своим условием это утверждение и не имеющая неизвестных. Такая задача сопровождается целью "проверка". Сначала она решается в режиме эквивалентных преобразований условия - в простейших случаях это уже может привести к получению константы "истина" либо "ложь". Однако, по достижении уровней 7 и 10 предпринимаются явные попытки доказать либо опровергнуть утверждение с помощью вспомогательной задачи на доказательство. На уровне 7 при обращении к вспомогательной задаче устанавливается лимит трудоемкости, в несколько раз меньший лимита, устанавливаемого на уровне 10.

Задача с целью "замещение"

В некоторых ситуациях бывает необходимо найти переформулировку группы утверждений относительно заданного контекста, при который были бы использованы только переменные заданного списка. Такие ситауции возникают, например, при решении задач на поиск экстремальных значений параметров, допускающих некоторый "сценарий" - геометрический чертеж, физический процесс и т.п. Здесь необходимо сначала перевести качественные условия, описывающие допустимость сценария, на язык соотношений для числовых параметров, а затем уже применять традиционные методы анализа.

Задача на описание, в которой нужно преобразовать условия к заданному списку переменных, снабжается целью "замещение". Неизвестными ее служат все переменные условий, не относящиеся к этому списку. Чтобы было возможно задать связи неизвестных с известными, допускается вхождение первых не только в условия, но и в посылки задачи.

Прием, обращающийся к задаче с целью "замещение" для отыскания экстремальных значений, будет рассмотрен в другом разделе. Здесь же рассмотрим лишь прием, который предпринимает попытку выразить атомарные числовые подвыражения условий, содержащие неизвестные, через известные числовые параметры. К программе его можно перейти из пункта "Попытка вычисления неизвестных подвыражений в задаче с целью "замещение" " оглавления программ.

Сначала проверяется отсутствие комментария "замещение", блокирующего повторную попытку применения приема, и этот комментарий сразу же вводится. Затем просматриваются условия F, причем сначала - все равенства, а затем уже остальные условия. Находится список S атомарных числовых параметров условия F, содержащих неизвестные и отличных от переменных. В программе этот список присвоен переменной x10. Если список непуст и состоит из выражений $t_1, \ldots, t_n$, то выбираются новые переменные $x_1, \ldots, x_n$. Находится список $p_1, \ldots, p_n$ всех известных числовых переменных, встречающихся в контексте условия F. Решается вспомогательная задача на описание Z', условия которой суть равенства $x_1 = t_1, \ldots, x_n = t_n$ и указатели типа значения "число(x_1)", ... , "число(x_n)". Посылками ее служат все посылки текущей задачи Z, к которым присоединены не содержащие выражений $t_1, \ldots, t_n$ остальные условия задачи Z. Задача Z' имеет цели (неизвестные $x_1 \ldots x_n$), (известно $p_1 \ldots p_n$), "полный", "явное", "прямойответ". Если найден ответ R задачи Z', определяющий выражения $r_1, \ldots, r_n$ для $x_1, \ldots, x_n$, то во всех условиях задачи Z

предпринимается замена выражений t_i на r_i. Далее - продолжение просмотра условий F.

Переход к новой несущественной неизвестной

Если задача на описание имеет такую несущественную неизвестную x, которая встречается (за исключением условия $P(x)$, определяющего тип значения x) только внутри заданного терма t вида $f(\ldots\ x\ \ldots)$, и этот терм имеет не менее двух вхождений в условия, то имеет смысл сам терм t обозначить новой вспомогательной несущественной неизвестной, а старую неизвестную x не рассматривать вообще. Этот прием реализован для частного случая, когда x и t принимают числовые значения. Его программу можно найти в пункте "Переход к новой несущественной неизвестной" оглавления программ. Несущественная неизвестная x обозначена в программе посредством x7, а вхождение терма t - x12. На роль вспомогательной несущественной неизвестной сначала выбирается новая переменная y (см. переменную x15), и решается задача на описание Z', имеющая условия "число(y)", $\exists_x(y = t)$. Посылками ее служат все посылки текущей задачи Z, а также все ее условия, не содержащие x. Задача Z' имеет неизвестную y и дополнительную цель "см", блокирующую попытки повторного применения данного приема при ее решении (что могло бы привести к зацикливанию). Если на задачу Z' получен ответ R, не содержащий переменной x, то выполняется преобразование задачи Z. При этом преобразовании оказывается удобным сохранить для новой несущественной неизвестной старое обозначение x. Вхождения терма t в условия задачи Z заменяются на x; переменная y в утверждении R заменяется на x, и отличные от "число(x)" конъюнктивные члены данного утверждения заносятся в условия задачи Z.

Учет комментария "внимание"

Если какой-либо прием не срабатывает из-за того, что не удается убедиться в истинности некоторого утверждения A, то он может установить режим слежения за появлением этого утверждения при последующих преобразованиях задачи (например, при выводе следствий). Для этого создается комментарий (внимание S), список S которого перечисляет тройки (A T u). Здесь T - терм задачи (условие либо посылка), вес которого должен быть уменьшен до величины u при обнаружении утверждения A. Предполагается, что уменьшение веса вызовет повторную попытку применения приема, которая может оказаться уже успешной. Если комментарий относится к списку посылок задачи, то отслеживается только появление утверждения A в списке посылок; если он относится ко всей задаче, то - появление утверждения A в списке условий. Чтобы установка на слежение была введена, в описание приема на ГЕНОЛОГе добавляется указатель "См(i)", где i - номер антецедента теоремы, для которого вводится слежение. Заметим, что хотя режим использования комментариев "внимание" и был введен в процессе обучения, однако впоследствии надобность в нем отпала, и сейчас он не востребован.

Программы приемов, учитывающих комментарий "внимание", находятся через пункты "Учет комментария посылок ВНИМАНИЕ", "Учет комментария ВНИМАНИЕ" оглавления программ. Эти приемы просматривают все посылки либо условия задачи, имеющие вес 0, и сравнивают их с утверждениями, за которыми введено слежение. При совпадении реализуются уменьшение веса и отключение слежения за найденным утверждением.

Учет комментария "Случай"

Иногда разбор случаев в задаче нужно организовать по той причине, что этого требует какой-либо вспомогательный пакетный оператор. Необходимость разбора случаев усматривается в процессе применения последнего, однако прямых способов передать информацию о необходимости разбора случаев внешней задачи не предусмотрено - оператор просто оказывается ложным либо выдает отказ, и на этом все заканчивается. Поэтому для передачи информации приходится использовать комментарий (Случай A) к внешней задаче, создаваемый специальным приемом пакетного оператора. Здесь A - дизъюнкция, определяющая разбор случаев. Прием задачи на описание, учитывающий данный комментарий, срабатывает на любом уровне, как только этот комментарий появляется. Он исключает комментарий, регистрирует утверждение A в списке условий, и сопровождает его комментарием "разборслучаев". Прием используется крайне редко.

Усмотрение принадлежности элемента классу

В заключение рассмотрим несколько нетипичный прием. Хотя он и относится к частному логическому символу - описателю "класс", программу его пришлось закрепить за типом задачи "описать".

Если задача имеет посылку вида $\text{класс}_x(A(x)\ \&\ldots) = y$, где $A(x)$ - элементарное утверждение, содержащее все переменные связывающей приставки x, то при появлении посылок вида $A(p)$ предпринимается проверка истинности остальных утверждений под описателем "класс", после чего выводится следствие $p \in y$. Этот прием можно было бы реализовать на ГЕНОЛОГе и предпринимать попытки применения его при обнаружении символа "класс". Однако, тогда будут игнорироваться новые утверждения $A(p)$, появляющиеся после попытки применения приема. Так как априори ничего не известно о понятиях, которые могут встретиться в утверждениях $A(p)$, пришлось создать общий прием, активизируемый на символе "описать" при текущем уровне 4. Он просматривает элементарные посылки веса 4 - кандидаты на роль $A(p)$, ищет для них подходящее равенство, определяющее класс, выполняет необходимые проверки, и выводит следствие.

10.2 Приемы задач на преобразование

Учет о.д.з.

Ввод утверждений, формулирующих условия на о.д.з., для задач на преобразование происходит так же, как и для задач на описание. На текущем уровне 0 проверяется наличие цели "одз"; эта цель удаляется, и предпринимается обращение к процедуре "одз".

Создание комментариев, определяющих схему сопровождения по о.д.з.

Прием - такой же, как для задач на описание.

Идентичные посылки задачи

Прием, устраняющий дублирование в посылках, такой же, как в случае задач на описание.

Понижение веса посылки

При создании вспомогательной задачи на преобразование веса посылок часто берутся большими, чем максимальный уровень обращения (например, равными "двадцать"). Это нужно для ускорения решения, чтобы не тратить время на повторное рассмотрение посылок, образующих уже "знакомый" контекст какой-то внешней задачи. Однако, такое замораживание посылок все же приходится частично отменять, используя специальные приемы. В частности, нежелательно блокировать рассмотрение равенств $A = B$, у которых заменяемая часть A встречается в преобразуемом терме t. Они, будучи заблаговременно ориентированы нужным образом, определяют стандартизацию обозначений одного и того же объекта, так что решателю стоило бы заменить в t выражение A на B. Чтобы это произошло, используется прием, который по достижения максимального уровня просматривает посылки $A = B$, вес которых больше текущего уровня, причем A входит в условие. Веса таких посылок заменяются на 0, и реализуется повторное рассмотрение задачи.

Упрощение посылок

Прием, упрощающий посылки, аналогичен соответствующему приему задач на описание. Проверяется, что посылка не была получена при выводе следствий и ранее не упрощалась. Для ее упрощения используется вспомогательная задача на преобразование, решаемая с достаточно сильным ограничением на трудоемкость (всего 50000 шагов работы интерпретатора). Таким образом, реализуется лишь самое быстрое поверхностное упрощение посылок.

Завершающее редактирование ответа

По исчерпании средств, отпущенных для решения задачи на преобразование, предпринимается попытка компактной переформулировки ее условия с помощью процедуры "свертка". Она инициируется для текущего уровня, равного максимальному уровню. Предварительно анализируется целевая установка, чтобы не применять прием там, где он может нарушить требуемый вид ответа. Для блокировки повторного применения приема служит комментарий "длина". Если максимальный уровень задачи меньше 7, то прием применяется лишь при наличии цели "учетрезультата", т.е. при завершающем редактировании ответа задачи на описание, имеющей цель (известно ...).

Выдача ответа

Ответ задачи на преобразование выдается приемами, сгруппированными в разделе "Задачи на преобразование" - "Выдача ответа" оглавления программ. Обычно это происходит на максимальном уровне, при исчерпании отведенных для решения

задачи средств; см. пункт "Выдача ответа задачи на преобразование - общий случай". Здесь проверяется, что условие задачи удовлетворяет всем требованиям, накладываемым на ответ целевой установкой. Прежде всего, отбрасываются случаи, когда условие содержит введенную при решении вспомогательную переменную x, указанную в комментарии (вспомпараметр x). Учет остальных целей происходит с помощью справочника "преобразовать", которому последовательно предъявляются тройки (текущая цель - задача - ее условие). Символом обращения к справочнику служит заголовок цели. Если ни одна из целей не блокирует выдачи ответа, то в качестве ответа берется условие задачи.

Заметим, что контроль пригодности условия выполняется в начале цикла сканирования, соответствующего максимальному уровню. Это означает, что в случае выдачи ответа другие приемы, которые могли бы сработать на максимальном уровне, применены не будут. Если нужно обеспечить применение каких-либо конкретных приемов, например, при упрощении выражения, то максимальный уровень обращения к задаче следует выбирать хотя бы на единицу большим их уровня срабатывания.

В специальных случаях ответ может быть выдан сразу же, на нулевом текущем уровне:

1. Если задача имеет цель (символ A), означающую немедленную выдачу ответа при получении заголовка A преобразуемого терма;

2. Если преобразуется не выражение, а утверждение, которое совпало с одной из посылок задачи. Тогда сразу выдается ответ "истина";

3. Если задача имеет цель (упрощение A), изначающую немедленную выдачу ответа при получении терма, длина которого меньше длины терма A;

4. Если задача имеет цель (заголовок $A_1 \dots A_n$) и не имеет других целей (кроме, быть может, цели с заголовком "декомпозиция"), причем получен терм с заголовком A_i; $i \in \{1, \dots, n\}$;

5. Если задача имеет цель "деление", означающую немедленную выдачу ответа при устранении дробного вида преобразуемого терма.

Заметим, что перечисленные особые случаи встречаются в работе системы исключительно редко.

Учет цели "заголовок"

Если задача имеет цель (заголовок $A_1 \dots A_n$), указывающую на необходимость преобразования условия к виду терма с заголовком - элементом списка $\{A_1$, $\dots, A_n\}$, то предпринимается попытка воспользоваться нормализаторами преобразования к заданным заголовкам. Эти нормализаторы находятся с помощью справочника "нормзаголовок". Прием применяется в процедурах, связанных с базой теорем.

Учет комментария посылок "внимание"

Прием - такой же, как для задачи на описание.

Приближенное вычисление константного выражения

Если задача имеет цель "выч", то ее условие представляет собой константный терм, и нужно найти его приближенное численное значение, ориентируясь на уровень точности, обеспечиваемый вычислениями с помощью математического сопроцессора в формате чисел "с плавающей запятой". В этой ситуации применяется либо процедура "вычпрог", составляющая программу вычислений, реализуемую далее оператором "Выч(программа ...)", либо процедура "вычконст", осуществляющая рекурсивное вычисление значения. Первая возможность несколько быстрее, и вторая используется только в случаях, когда условие содержит операции, не реализуемые командами программы оператора "Выч(программа ...)".

10.3 Приемы задач на доказательство

Учет о.д.з.

Выполняется так же, как в предыдущих случаях.

Создание комментариев, определяющих схему сопровождения по о.д.з.

Так же, как в предыдущих случаях.

Идентичные посылки задачи

Так же, как ранее.

Противоположные посылки задачи

Если при просмотре посылок A, имеющих вес 0, обнаруживается, что отрицание A тоже имеется в посылках, то ранее введенные комментарии (выводимо ...) удаляются; регистрируется информация о том, что ответ извлечен из найденных противоположных посылок, и выдается ответ "истина".

Совпадение условия с одной из посылок

Если на текущем уровне 0 усматривается, что условие совпадает с некоторой посылкой, то выдается ответ "истина". Предварительно в комментарии (выводимо ...) регистрируется информация об использовании этой посылки.

Отрицание конъюнктивного члена условия входит в посылки

Если условие задачи имеет вид $A_1\&\ldots\&A_n$, причем отрицание некоторого A_i содержится в посылках, то условие задачи заменяется на константу "ложь". Уровень срабатывания приема равен 0.

Упрощение посылок

Прием аналогичен приему, упрощающему посылки задачи на описание. Уровень применения его равен 2. После проверки того, что условие задачи не имеет заголовка "и" либо "существует" (иначе будут применяться другие, более приоритетные приемы) начинается просмотр посылок A. Отбираются те посылки, для которых целесообразно предпринимать попытку их упрощения. Она реализуется с помощью вспомогательной задачи на преобразование.

Доказательство от противного

Если прямое доказательство не удается довести до конца с помощью приемов малых уровней, то на уровне 5 либо 7 предпринимается попытка доказательства от противного. Уровень 5 относится к случаям, когда правдоподобно извлечение достаточно информативных следствий из отрицания условия A: либо A имеет вид "не(равно(x t))", где x - переменная, не входящая в t, либо A имеет вид равенства двух нечисловых переменных, либо A содержит выражение "значение(f t)"для константного t, а в посылках имеется кванторная импликация для f. В прочих случаях уровень срабатывания равен 7. Применение приема блокируется, если доказывается шаг индукции. Кроме того, оно блокируется, если A не имеет вида "не(равно(x t))", а задача содержит некоторые конкретные понятия ("расстояние", "угол", и т.п.).

Разумеется, все это - сугубо эвристические ограничители. Для рассмотренного обучающего материала они достаточны, так как отсекают большинство заведомо бесполезных попыток доказательства от противного. Однако, эти ограничители ни в коей мере не должны рассматриваться как нечто завершенное. Прием доказательства от противного - чрезвычайно общий метод, и для хорошего управления им понадобится накопление целой базы эвристических фильтров, учитывающих особенности конкретных ситуаций для различных предметных областей. Возможно, по мере развития этой базы придется распределить фильтры по программам вспомогательного справочника.

Если попытка доказательства от противного не отвергнута, то вводится комментарий "противоречие", блокирующий повторную попытку, и создается вспомогательная задача на исследование Z'. Ее список посылок образован всеми посылками задачи на доказательство, к которым добавлено отрицание условия A. Задача Z' имеет единственную цель "противоречие". Происходит обращение к решению задачи Z' с максимальным уровнем 7. Если после этого в посылках задачи Z' обнаруживается константа "ложь", то на текущую задачу Z выдается ответ "истина".

Использование замороженного равенства в посылках

Для ускоренной проверки часто используются задачи, у которых веса посылок больше, чем максимальный уровень. Обычно они сопровождаются комментарием "извлекается". Однако, у таких задач оказывается заблокированной стандартизация обозначений одного и того же объекта, использующая равенства из посылок. Это может привести к выдаче отказа даже в очевидных случаях. Поэтому предусмотрен специальный прием, который при наличии цели "извлекается" проверяет наличие посылок вида $A_1 = A_2$, где A_1 входит в условие задачи. Для этих посылок предпринимается замена в условии всех вхождений A_1 на A_2. Веса посылок не изменяются. Уровень

срабатывания приема равен 1; для блокировки его повторного применения служит комментарий "результподст".

Упрощение условия

На достаточно высоком уровне сканирования (шестом) предпринимается попытка обратиться к вспомогательной задаче на преобразование для упрощения условия. Прием блокируется, если условие имеет своим заголовком конъюнкцию, дизъюнкцию либо квантор, либо не содержит более чем однобуквенных выражений. Имеется ряд других фильтров, ограничивающих применение приема. Для блокировки повторных попыток упрощения служит комментарий "упростить".

Развертка - свертка неэлементарных посылок

Если посылка задачи содержит кванторы и описатели, то иногда ее удается существенным образом упростить, используя сначала полную расшифровку входящих в нее понятий по определениям, а затем, после общей стандартизации результата расшифровки, обратную свертку "по определениям". Прием, выполняющий такие действия, активизируется на 6-м уровне. Для расшифровки "по определениям" служит обращение к вспомогательной задаче на описание, условие которой образовано выделенной неэлементарной посылкой, а цели суть "редакция", "развертка", "полный", "прямойответ". Она решается с небольшим уровнем обращения - 4. Свертка выполняется аналогичной задачей, в отсутствии цели "развертка". Для блокировки повторов служит комментарий к посылке "стандтерм".

Выбор новой неизвестной при получении известного условия

В некоторых предметных областях для доказательства утверждения удобно бывает выделить среди переменных подмножества условно "известных" и "неизвестных". Выделенные неизвестные регистрируются в комментарии к задаче (неизвестные $x_1 \ldots x_n$); известные - в комментарии (известно $a_1 \ldots a_m$). При этом начинают работать приемы, предпринимающие попытки выразить "неизвестные" через "известные". После подстановки таких выражений в условие задачи оно обычно становится очевидным. Данная схема применяется, например, при доказательстве геометрических соотношений. В качестве неизвестной здесь естественно выбирать какую-либо числовую переменную, входящую в условие, а в качестве известных - прочие числовые переменные. Если в процессе преобразований (например, выражения неизвестной через известные) условие перестает содержать неизвестные, то выбирается новая неизвестная. Прием, выполняющий такой выбор, активизируется на уровне 7. В качестве новой неизвестной он выбирает произвольный параметр, указанный в комментарии (известно ...) и входящий в условие. Этот параметр переносится из списка (известно ...) в список (неизвестные ...). Веса посылок и условия при этом понижаются до 0.

Учет комментария посылок "внимание"

То же, что для задач на описание и преобразование.

Ввод комментария "раздел"

Для различных целей может понадобиться список максимальных разделов, к которым относятся встречающиеся в задаче понятия. Такой список A создается в начале решения задачи (при текущем уровне 0) и сохраняется в комментарии (раздел A).

10.4 Приемы задач на исследование

Учет о.д.з.

Выполняется так же, как в предыдущих случаях.

Создание комментариев, определяющих схему сопровождения по о.д.з.

Так же, как в предыдущих случаях.

Идентичные посылки задачи

Так же, как ранее.

Противоположные посылки задачи

Если некоторая посылка задачи, имеющая вес 0, оказывается совпадающей с отрицанием другой посылки, то она заменяется на константу "ложь".

Удаление чрезмерно больших посылок

Чрезмерно громоздкие посылки, возникающие при выводе следствий, чаще всего оказываются бесполезны для решения задачи. При этом они сильно замедляют действия системы. Поэтому желательно иметь прием, который бы исключал такие посылки. Однако, необходимо блокировать отбрасывание посылок, несущих уникальную и необходимую в задаче информацию. Например, не следует удалять посылки вида $x = t$, где x - неизвестная внешней задачи на описание, t - известное выражение. Заметим, что иногда эти посылки и оказываются рекордно длинными, например, если ответ по необходимости громоздок. Здесь возникает непростая проблема накопления эвристических ограничителей приема. На текущий момент накоплена какая-то, видимо, далеко не идеальная система таких ограничителей. Подробнее можно ознакомиться с ними по программе приема (см. пункт "Удаление чрезмерно больших посылок в задачах, имеющих цель ИЗВЕСТНО" оглавления программ). Посылка считается чрезмерно длинной, если в ней более 130 символов. В специальных случаях эта константа увеличивается до 300. Блокируется отбрасывание дизъюнкций, снабженных комментарием "разборслучаев", а также конъюнкций (впрочем, последнее отменяется, если неизвестные задачи частично определены). Блокируется также отбрасывание посылок, возникших с самого начала решения задачи.

Усмотрение ответа внешней задачи на описание

При решении задачи на исследование Z', являющейся блоком анализа задачи на описание Z, предпринимается попытка получить такую систему следствий, которая явно указала бы искомые значения неизвестных, и таким образом могла бы быть взята (с необходимыми коррекциями) в качестве ответа задачи Z. Усмотрение того, что данная система следствий или ее фрагмент уже получены, обычно выполняется приемами символа "равно". Однако, в одном специальном случае для усмотрения ответа понадобился общий прием задач на исследование. Этот случай связан с решением геометрических задач на построение или похожих на них задач. Процедура решения заключается в том, что предпринимается некоторая изначальная фиксация нескольких точек как "известных" - для устранения степеней свободы, допускаемых параллельными сдвигами, поворотами и т.п. Далее происходит анализ чертежа, в процессе которого постепенно доопределяются (с помощью допустимых средств построения) прочие точки. На каждом шаге доопределения возникают одно или несколько утверждений, рассматриваемых как выражение новой точки через старые. После этого данная точка уже считается известной. Утверждения, определяющие фиксацию точки A, помечаются комментариями (найдено A). В некоторый момент оказывается, что все неизвестные в задаче Z точки определились. Тогда и должен сработать указанный выше прием усмотрения ответа. Уровень срабатывания его равен 1; программу можно найти в пункте "Попытка усмотрения ответа внешней задачи на описание при определении всех ее неизвестных" оглавления программ. Прежде всего, прием анализирует целевую установку и проверяет непересечение списков неизвестных задач Z и Z'. Последнее означает, что все неизвестные задачи Z определены. Составляется список S пар (переменная - утверждения, определяющие ее значение согласно комментариям "найдено"). Здесь могут встречаться не только неизвестные задачи Z, но и вспомогательные переменные, через которые будут выражаться эти неизвестные. В программе список S присваивается переменной x7. Пары списка S переупорядочиваются так, чтобы новые переменные определялись только с использованием ранее определенных; это дает список S'. В программе он присвоен переменной x8. Далее по S' находится список R всех утверждений, используемых, прямо или косвенно, для определения неизвестных задачи Z. В программе список R присвоен переменной x9. Составляется список P всех известных посылок задачи Z', не содержащих переменных, определенных списком S, пополненный всеми известными условиями задачи Z. После этого предпринимается попытка доказать, что условия задачи Z являются следствиями ее посылок, объединенных со списками R и P. Для задач на построение этот шаг означает попытку доказать, что найденная при анализе чертежа последовательность действий приводит к желаемым результатам. Использованные при доказательстве утверждения списка P добавляются, после упрощения их вспомогательными задачами на описание, к списку R. Последний и рассматривается как ответ. Чтобы зарегистрировать его в таком качестве, прием предпринимает замену всех условий внешней задачи Z на утверждения списка R, а список неизвестных задачи Z пополняет теми вспомогательными переменными, которые вошли в R. Затем создается комментарий "ответ" к задаче Z, и происходит возвращение к ней от задачи Z'.

Учет комментария посылок "внимание"

То же, что для задач на описание, преобразование и доказательство.

Пополнение накопителя "полныепосылки"

Если задача на описание имеет своим условием кванторную импликацию $\forall_x(A_1 \& \ldots \& A_n \rightarrow A_0)$, то обычно предпринимается попытка найти ее бескванторную эквивалентную переформулировку. В простейших случаях этого удается добиться с помощью кванторных определений либо за счет явного разрешения подкванторных утверждений относительно переменных связывающей приставки x. В более сложных случаях приходится применять процедуру накопления такого списка бескванторных следствий кванторной импликации, которые оказались бы достаточны для истинности этой импликации. Это - хорошо известная процедура усиления необходимого условия, пока оно не станет достаточным. Для ее реализации создан специальный прием символа "длялюбого", который будет подробнее описан впоследствии. Вывод следствий осуществляется в рамках вспомогательной задачи на исследование, имеющей цель "длялюбого". Этой задаче передаются в качестве посылок все антецеденты A_i и все конъюнктивные члены консеквента A_0. Предварительно проверяется реализуемость антецедентов.

Накопление бескванторных следствий кванторной импликации происходит в комментарии (полныепосылки S). Его обеспечивает общий прием задач на исследование, просматривающий вновь выведенные посылки P. Если такая посылка не содержит переменных связывающей приставки x, то она непосредственно регистрируется в S. Иначе - рассматривается кванторная импликация $\forall_x(A_1 \& \ldots \& A_n \rightarrow P)$, и предпринимается попытка преобразовать ее в бескванторное утверждение с помощью вспомогательной задачи на описание. Если это удается, результат тоже регистрируется в списке S.

10.5 Приемы символа "и"

Переходим к приемам, активизируемым при просмотре входящих в условия и посылки задачи логических символов. Эти приемы, в основном делятся на три группы - простые общелогические упрощения утверждений, применяемые почти без ограничений; приемы, планирующие общий ход действий по решению задачи (разбор случаев; исключение неизвестной, явно выраженной через другие неизвестные, и т.п.), и приемы, осуществляющие вывод следствий. Некоторые приемы имеют достаточно общий характер, в то время как оценить из общих соображений целесообразность их применения бывает затруднительно. Тогда приходится прибегать к специальным "решающим" справочникам, подлежащим пополнению в процессе обучения системы и оценивающим целесообразность применения данного общего приема в зависимости от тех понятий, которые встретились в конкретном контексте его срабатывания.

Начнем с символа "и". Группу его реализованных на ЛОСе приемов можно найти в разделе оглавления программ "Приемы решателя" - "Общие приемы" - "Конъюнкция". Все эти приемы крайне просты.

Конъюнкция с противоположными членами

Прием применяется на уровне 0. Он производит попарные сравнения различных операндов конъюнкции. Если обнаруживаются операнды, один из которых является отрицанием другого, то конъюнкция заменяется на константу "ложь".

Конъюнкция с повторяющимися членами

Прием аналогичен предыдущему, но проверяет совпадение различных операндов. При обнаружении совпадения один из операндов отбрасывается. В случае, когда операндов было всего два, отбрасывается также символ конъюнкции.

Конъюнктивное условие задачи на доказательство

Если условие задачи на доказательство имеет вид $A_1 \& \ldots \& A_n$, то обычно эта задача разбивается на n независимо решаемых задач, имеющих условия $A_1, \ldots, A_n$. Уровень срабатывания приема равен 2 - чтобы в простейших случаях успели сработать приемы, устраняющие конъюнкцию прочими средствами. Блокировка приема была ведена лишь для геометрических задач на доказательство, которые оказалось целесообразнее решать без разбиения условия. Комментарии исходной задачи передаются подзадачам после обработки их справочником "конъюнктчлен". Список посылок каждой подзадачи представляет собой независимую копию списка посылок исходной задачи. Информация об использовании посылок извлекается из каждой подзадачи после ее решения, и объединенный список передается исходной задаче перед выдачей ответа "истина" на нее.

Конъюнктивное условие задачи на описание

Если задача на описание имеет своим условием утверждение $A_1 \& \ldots \& A_n$, то это условие заменяется на A_1, а утверждения $A_2, \ldots, A_n$ присоединяются в качестве ноых условий. Комментарии к исходному условию передаются измененному и новым условиям после обработки их справочником "и". Веса этих условий полагаются равными 0. Прием применяется без ограничений, и уровень срабатывания его равен 0.

Конъюнктивная посылка

Если утверждение $A_1 \& \ldots \& A_n$ представляет собой посылку задачи, то она разбивается на отдельные конъюнктивные члены. Это происходит так же, как в предыдущем приеме, причем используется тот же самый справочник "и". Для задач на исследование, имеющих цель "текстоваязадача", работа приема несколько модифицирована - посылки $A_2, \ldots, A_n$ добавляются не в начале списка посылок, как в общем случае, а непосредственно после измененной посылки. Это делается для сохранения той информации, которая определяется порядком слов в анализируемом тексте, например, информации о временной последовательности событий. Впрочем, для сохранения последней предусмотрены и другие средства.

10.6 Приемы символа "или"

Количество приемов в этом разделе значительно больше, чем в предыдущем, и некоторые из них уже достаточно сложны.

Простейшая стандартизация

Начнем с серии простых приемов, осуществляющих упрощение дизъюнкции.

1. Устранение повторных дизъюнктивных членов. Если какие-то два операнда A_i, A_j дизъюнкции $A_1 \vee \ldots \vee A_n$ совпадают, то один из них удаляется. При $n = 2$ удаляется также знак дизъюнкции. Уровень срабатывания приема равен 0.

2. Дизъюнкция с противоположными членами. Если какой-либо операнд дизъюнкции совпадает с отрицанием другого операнда, то дизъюнкция заменяется на константу "истина". Исключение составляют дизъюнктивные условия и посылки, снабженные комментарием "разборслучаев". Обычно они имеют противоположные операнды, но исключать их не следует, так как дизъюнкция должна инициировать срабатывание приема, выполняющего разбор случаев. Уровень срабатывания приема равен 0.

3. Преобразование дизъюнкции, если в контексте имеется дизъюнктивный член или его отрицание. Если в контексте, относительно которого рассматривается дизъюнкция $A_1 \vee \ldots \vee A_n$, имеется утверждение A_i, то эта дизъюнкция заменяется на константу "истина". Если в контексте имеется отрицание утверждения A_i, то дизъюнкция заменяется на $A_1 \vee \ldots\ A_{i-1} \vee A_{i+1} \vee \ldots \vee A_n$. Уровни срабатывания приема - 0 либо 7. Уровень 7 введен для повторного контроля, так как возможность применения приема зависит от появления в контексте новых утверждений.

4. Анализ конъюнктивных членов дизъюнктивного члена. Если дизъюнктивный член A_i имеет вид $B_1 \,\&\, \ldots \,\& B_m$, причем некоторое B_i встречается в контексте дизъюнкции, то оно исключается из A_i. Если в контексте встречается отрицание утверждения B_i, то исключается дизъюнктивный член A_i. Оба приема срабатывают на уровне 0.

5. Усмотрение в контексте дизъюнкции, поглощающей данную. Если в контексте имеется дизъюнкция, множество дизъюнктивных членов которой является подмножеством множества $A_1, \ldots, A_n$ дизъюнктивных членов текущей дизъюнкции, то последняя заменяется на константу "истина". Уровни срабатывания приема равны 0 либо 7.

6. Вынесение за скобку общей части двух дизъюнктивных членов. Если два дизъюнктивных члена A_i, A_j представляют собой конъюнкции, имеющие непустое множество одинаковых членов, т.е. представимые в виде $B\&C$ и $B\&D$, то они объединяются в один новый дизъюнктивный член $B\&(C \vee D)$. Имеются небольшие ограничения на применение данного приема - исключение составляют случай задачи на преобразование, имеющей цель "днф", а также случай редактирования ответа задачи на описание, когда C содержит явное выражение некоторой неизвестной, встречающейся в B. Уровень срабатывания приема равен 1.

7. Поглощение дизъюнктивного члена. Если некоторый дизъюнктивный член A_i имеет своим конъюнктивным членом дизъюнктивный член A_j, то A_i отбрасывается. Уровень срабатывания приема равен 1.

8. Использование оператора "блокдизъюнкции" для упрощения дизъюнктивного условия в редактируемом ответе задачи на описание. Оператор "блокдизъюнкции" включает в себя три простейших преобразования: исключение вложенных дизъюнкций и вложенных конъюнкций, а также вынесение за скобку общей части двух дизъюнктивных членов. Прием применяется на текущем уровне 0. Для блокировки повторных попыток обработки им того же самого условия используется комментарий "блокдизъюнкции".

9. Использование преобразования $A \vee (B\&(A \vee C)) = A \vee (B\&C)$. Прием применяется на уровне 1.

Как легко заметить, предложенный список упрощающих дизъюнкцию приемов не претендует на полноту. Однако, увеличение этого списка из каких-либо общих соображений, в особенности для малых уровней срабатывания, могло бы повлечь за собой неоправданное замедление работы решателя. Поэтому данная коллекция включает в себя лишь то, что фактически было востребовано на проработанном обучающем материале, и дальнейшее пополнение ее должно происходить с учетом целесообразности временных затрат решателя на попытки применения новых приемов.

Дизъюнктивное условие задачи на доказательство

Если задача на доказательство имеет условие $A_1 \vee \ldots \vee A_n$, то она сводится к задаче на доказательство какого-то одного утверждения A_i, с присоединением к посылкам отрицаний утверждений $A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n$. Чтобы выбрать нужное A_i, предварительно выполняется переупорядочение дизъюнктивных членов: во-первых, предпочтение отдается тому из двух утверждений, которое не имеет заголовка "существует"; во-вторых, тому, которое не имеет заголовка "не"; в-третьих, более короткому. Комментарии во вспомогательную задачу переносятся после обработки их справочником "или". Если на нее получен ответ "истина", то во внешнюю задачу переносится информация об использованных посылках, и затем выдается ответ "истина". Уровень срабатывания приема выбран равным 2, чтобы в случаях, когда это возможно, успевали сработать приемы общей стандартизации, исключающие дизъюнкцию.

Разбор случаев по дизъюнктивному условию задачи на описание

Это один из наиболее важных общелогических приемов. Уровни, на которых может быть предпринятая попытка его применения, равны 1,2,3,5 и 7. Конкретное значение зависит от контекста. Если дизъюнктивное условие появляется на этапе редактирования ответа задачи на описание (т.е. при решении задачи на описание, имеющей цель "редакция"), то разбор случаев допускается в двух ситуациях - либо дизъюнкция не содержит неизвестных и помечена комментарием "разборслучаев", либо в каком-либо из дизъюнктивных членов явно указывается значение несущественной неизвестной. В начале программы приема предусмотрена еще одна проверка того, что какой-либо дизъюнктивный член совпадает с условием задачи. Если это так, то дизъюнкция заменяется на константу "истина". Хотя данная проверка уже предпринималась на уровне 0 другим приемом, но с тех пор (по достижении текущего уровня) могли измениться прочие условия задачи, и повторение ее иногда оказывается результативным.

Далее уточняется уровень срабатывания приема. Если условие выделено комментарием "разборслучаев", то он равен 1. Для условия, содержащего неизвестные, уровень обычно равен 3. Если неизвестных в дизъюнкции нет, то уровень берется равным 7 при редактировании ответа задачи на описание и 5 в противном случае. При наличии цели "пример" ограничения на уровень срабатывания снимаются.

Чтобы блокировать повторную попытку применения данного приема при данном уровне срабатывания U, используется комментарий к условию (или U). Если имеется несколько дизъюнктивных условий, то предпочтение отдается тому, которое в каждом своем подслучае дает явное значение для какой-либо неизвестной задачи.

Определяется максимальный уровень, с которым будут решаться задачи для отдельных подслучаев (см. переменную x7 в программе приема). Обычно он равен максимальному уровню текущей задачи, хотя при решении задачи на подбор примера возможны попытки рассмотрения подслучаев с малым уровнем, равным 3. Если решается задача на подбор примера, то из двух подслучаев сначала рассматривается тот, который задается более коротким утверждением.

Далее находится список $A_1, \ldots, A_n$ дизъюнктивных членов рассматриваемого условия, и начинается их последовательный просмотр. Пусть A_i - текущий просматриваемый подслучай. Если задача - на подбор примера и имеет цель (независит ...), причем A_i содержит какую-либо переменную данной цели, то рассмотрение этого подслучая отменяется. Если выбранное дизъюнктивное условие было помечено комментарием "Одз", то предпринимается попытка скорректировать его члены (с сохранением эквивалентности исходному условию), присоединив к ним те необходимые для сопровождения по о.д.з. утверждения, которые можно усмотреть из отрицания остальных членов.

Наконец, создается вспомогательная задача Z', полученная из текущей задачи Z заменой дизъюнктивного условия на утверждение A_i. Здесь используется оператор "спуск", заменяющий заданное условие на новое, а также копирующий блок анализа и корректирующий комментарии. Если дизъюнктивное условие не имело комментария "разборслучаев", то вхождения в условия задачи Z' ранее разобранных подслучаев заменяются на константу "ложь". Корректируется блок анализа задачи Z'. Из него удаляется рассматриваемое дизъюнктивное условие и понижаются до 0 веса ряда посылок - например, тех, которые имеют общую переменную с A_i. Далее предпринимается обращение к решению задачи Z'. Если на нее получен "отказ", а задача Z не имела цели "пример" и имела цель "полный", то на задачу Z тоже выдается "отказ". Если на задачу Z' получен ответ R_i, отличный от символа "отказ", то прежде всего предпринимается перенесение из списка посылок задачи Z' в список посылок задачи Z тех утверждений, которые могут впоследствии оказаться полезными - они помечаются в задаче Z' комментариями "облвхожд". Аналогичная работа проводится с комментариями - справочник "преобразование" отбирает и корректирует их для перенесения из Z' в Z. Если R_i - константа "истина", то она сразу выдается в качестве ответа. Для константы "ложь" выделен один специальный подслучай, когда ее тоже можно сразу же выдать в качестве ответа. В прочих ситуациях частичный ответ R_i регистрируется в накопителе ответа (его роль играет программная переменная x8). Для задач на подбор примера определяются условия Q_i на известные параметры, при которых R_i дает требуемый пример. Проверяется, не является ли дизъюнкция $Q_1, \ldots, Q_i$ следствием посылок задачи Z. Если это так, то уже можно выдавать полный ответ, и рассмотрение последующих подслучаев A_j обрывается.

По окончании рассмотрения всех A_i (с учетом указанной выше возможности обрыва) возникают списки ответов $R_1, \ldots, R_k$ для разобранных подслучаев. Если решается задача на поиск примера, то возникает также список условий $Q_1, \ldots, Q_k$ на известные параметры, относительно которых найдены соответствующие ответы. Первый список (в виде наборов конъюнктивных членов утверждений R_i) присвоен переменной x8; второй - переменной x9. Переменная x10 является индикатором охвата условиями $Q_1, \ldots, Q_k$ всего многообразия возможностей, предоставляемых посылками задачи Z: если все случаи охвачены, то она равна 0.

Прежде всего, в ситуации с задачей на поиск примера и при ненулевом x10, предпринимается попытка установить, что дизъюнкция утверждений $Q_1, \ldots, Q_k$ является следствием посылок задачи Z. Если это не так, то применение приема блокируется. Иначе - определяются общая не содержащая неизвестных часть M всех ответов $R_1, \ldots, R_k$, а также остатки $S_1, \ldots, S_k$ этих ответов. Определяется утверждение R вида $M\ \&(S_1 \vee \ldots \vee S_k))$, в котором выполняются тривиальные упрощения, соответствующие тождественной истинности или ложности каких - либо S_i. Если задача Z не имеет цели "упростить", то в качестве ответа на нее выдается R. Это же происходит еще в некоторых специальных случаях. Иначе - предпринимается обращение к нормализатору "нормили", который и дает окончательный ответ на задачу.

При данной схеме разбора случаев получается, что все завершающее редактирование дизъюнкции ответов, найденных в подслучаях, происходит только внутри нормализатора "нормили". Никаких преобразований этой дизъюнкции в рамках сканирования той или иной вспомогательной задачи не предусмотрено. Соответственно, нормализатор "нормили" оказывается исключительно большой процедурой, вбирающей в себя всю технику склейки дизъюнктивных логических конструкций, имеющуюся в различных предметных областях. Подробнее его содержимое будет рассмотрено ниже: общелогическая составляющая в данной главе, остальное - в главах, посвященных отдельным предметным областям.

Разбор случаев по дизъюнктивной посылке задачи на описание

Если задача на описание имеет своей посылкой дизъюнкцию $A_1 \vee \ldots \vee A_n$, то она может решаться путем разбора случаев по этой дизъюнкции. Уровни срабатывания приема, реализующего разбор случаев, равны 1, 2 либо 3. Уровень равен 1, если посылка выделена комментарием "разборслучаев"; равен 2, если задача имеет цель "пример", и равен 3 в прочих случаях. Если посылка не выделена комментарием "разборслучаев", то проверяется, что хотя бы один ее параметр входит в условия задачи. Далее начинается просмотр утверждений, определяющих подслучаи. Пусть A_i - текущее такое утверждение. Создается вспомогательная задача Z', полученная из Z заменой рассматриваемой посылки на A_i. В программе эта задача присвоена переменной x11. Комментарии к посылкам переносятся в Z' из Z при помощи справочника "вариант"; комментарии к условиям - при помощи справочника "компонента". Добавляется цель "текпосылка", имеющая чисто информационный характер - она означает, что данная вспомогательная задача на описание возникла при разборе случаев по дизъюнктивной посылке. Если задача Z имела блок анализа, то копия его передается задаче Z' после добавления посылки A_i. Веса посылок задачи Z', которые содержат условные выражения, имеющие своим подутверждением утверждение A_i с отброшенным внешним отрицанием, уменьшаются до 0.

Далее происходит обращение к решению задачи Z' с тем же максимальным уровнем,

какой имела задача Z. Если получается "отказ", то и на задачу Z выдается "отказ". Иначе - по завершении рассмотрения всех подслучаев возникает набор ответов $R_1, \ldots, R_n$. Если задача Z имеет цель (независит ...) и не имеет цели "полный" (что обычно выполняется при наличии цели "независит"), то в качестве ответа берется конъюнкция всех R_i. Быть может, в каких-то случаях это существенно сужает ответ, но зато исключает его зависимость от указанных в цели параметров. При отсутствии цели (независит ...) в качестве заготовки ответа рассматривается утверждение вида $(A_1 \& R_1) \vee \ldots \vee (A_n \& R_n)$. Оно упрощается с помошью вспомогательной задачи на описание, цели которой получены добавлением к целям задачи Z символа "редакция", и затем выдается как окончательный ответ.

Разбор случаев по дизъюнктивной посылке задачи на доказательство или на преобразование

Разбор случаев по дизъюнктивной посылке задачи на преобразование предпринимается лишь тогда, когда эта посылка выделена комментарием "разборслучаев". Уровень срабатывания приема при этом равен 0. Для задачи на доказательство уровень срабатывания равен 1 при наличии комментария посылки "разборслучаев", равен 2, если хотя бы одна из альтернатив не имеет вида отрицания равенства, и равен 5 в остальных случаях. Для задачи на преобразование, результат разбора случаев не всегда является окончательным ответом. В некоторых ситуациях он лишь замещает текущую версию преобразуемого условия. Тогда, с целью блокировки повторных применений приема, вводится комментарий посылки "или".

Обозначим для дальнейших рассмотрений дизъюнктивную посылку через $A_1 \vee \ldots \vee A_n$. Если решается задача на преобразование, имеющая цели (обозначение ...), то составляется список S всех переменных, упоминаемых в таких целях. Эти переменные суть вспомогательные обозначения, которые не должны входить в ответ задачи. Поэтому нежелательно их появление в альтернативах A_i, которые войдут в условные выражения для результата разбора случаев. Чтобы исключить такую возможность, рассматривается список "укороченных" альтернатив B_i, полученных из A_i отбрасыванием всех конъюнктивных членов, зависящих от S. Если все утверждения B_i отличны от константы "истина", причем усматривается их несовместность, то они заменяют в дальнейших преобразованиях утверждения A_i. Иначе применение приема блокируется.

Далее начинается последовательный просмотр дизъюнктивных членов посылки. Пусть A_i - текущий такой член. Вводится вспомогательная задача Z', полученная из текущей задачи Z заменой дизъюнктивной посылки на A_i. Если Z имеет комментарий "смпосылка", то веса всех посылок задачи Z' обнуливаются. Иначе на ноль заменяется только вес посылки A_i. Комментарии к посылкам и к задаче переносятся из Z в Z', соответственно, с помощью справочников "дизъюнктчлен" и "разборслучаев". Предпринимается понижение до нуля весов посылок, имеющих такие условные выражения, которые содержат A_i. После этого происходит обращение к решению задачи Z' с тем же максимальным уровнем, что и у задачи Z. Если получен "отказ", то в случае задачи на доказательство сразу выдается итоговый отказ. В случае задачи на преобразование получение отказа приводит лишь к обрыву попытки применения данного приема. Для задачи на преобразование заполняется накопитель частичных ответов $R_1, \ldots, R_n$, найденных в отдельных подслучаях. Значением программной переменной x6 становится пара, первым элементом которой служит данный накопи-

тель, а вторым - накопитель использованных во всех подслучаях посылок.

По окончании разбора случаев в задаче на доказательство выдается ответ "истина". Предварительно обеспечивается коррекция комментария (выводимо ...) к текущей задаче Z, чтобы он ссылался на все использованные при разборе подслучаев исходные посылки.

Если решалась задача на преобразование, то после получения списка частичных ответов $(R_1, \ldots, R_n)$ предпринимается попытка усмотреть возможность поглощения одних подслучаев другими. Пусть, например, некоторое утверждение A_i имело своими конъюнктивными членами равенства $x_1 = t_1, \ldots, x_m = t_m$, фиксирующие значения некоторых переменных $x_1, \ldots, x_m$. Если в другом утверждении A_j значения этих переменных не были фиксированы, причем удается усмотреть равенство выражения R_i результату подстановки выражений $t_1, \ldots, t_m$ в R_j вместо переменных $x_1, \ldots, x_m$, предполагая истинность A_i, то вместо R_i в подслучае A_i можно использовать R_j.

Далее строится условное выражение R вида "вариант(A_1 R_1 вариант(A_2 R_2 ... вариант(A_{n-1} R_{n-1} R_n) ...))". Если какие-либо подслучаи A_i в действительности оказались невозможными и в качестве R_i был получен символ "противоречие", то эти подслучаи при построении R отбрасываются. Чтобы получить окончательный результат, к R применяется нормализатор "упрощвариант". Исключение составляют задачи, имеющие цель "класс". Тогда нормализация отменяется, и вместо выдачи ответа выражение R замещает условие текущей задачи Z. Как и в случае задач на доказательство, предпринимается коррекция комментария (выводимо ...).

Преобразования дизъюнкции при редактировании ответа задачи на описание

При редактировании ответа задачи на описание могут встречаться дизъюнктивные условия, по которым не предпринимается разбор случаев. Для работы с ними предусмотрен ряд простых приемов, которые перечисляем ниже. Напомним, что этап редактирования ответа выделяется наличием цели "редакция".

1. Попытка усмотрения избыточности дизъюнкции отрицаний равенств. Если при редактировании ответа возникает условие вида $x_1 \neq t_1 \vee \ldots \vee x_n \neq t_n$, где переменные $x_1, \ldots, x_n$ не встречаются в термах $t_1, \ldots, t_n$, то просматриваются другие условия g, все свободные переменные которых содержатся в списке $x_1, \ldots, x_n$. Предпринимается попытка усмотреть из контекста утверждения g отрицание результата подстановки в g выражений $t_1, \ldots, x_n$ вместо переменных $x_1, \ldots, x_n$. Как только это удается, дизъюнктивное условие отбрасывается. Уровень срабатывания приема равен 2.

2. Перемещение вглубь дизъюнкции тех утверждений, для которых внутри дизъюнкции явным образом определяются значения их неизвестных. Если в условии задачи на описание, имеющей цель "редакция", встречается дизъюнкция D вида $(x = t)\&A \vee B_1 \ldots \vee B_n$, где x - неизвестная, то рассматриваются все расположенные над ней дизъюнкции D' того же условия (допуская совпадение D' и D). Пусть $D' = C_1 \vee \ldots \vee C_m$. Если D' является операндом конъюнкции K, причем существуют другие ее операнды, зависящие от x, то рассматривается конъюнкция K' всех таких операндов. Они исключаются из K и переносятся в D', которая приобретает вид $C_1\&K' \vee \ldots \vee C_m\&K'$. Аналогичное преобразование

выполняется, если D' - корневая дизъюнкция. Вместо операндов конъюнкции K тогда берутся все прочие условия задачи. Данный прием подготавливает возможность подстановки значения t вместо x в содержащие x утверждения. Он имеет два уровня срабатывания - 1 и 7.

3. Логическое упрощение. Если условие редактируемого ответа содержит дизъюнкцию вида $A \vee (B\&\neg A)$, где утверждение A содержит неизвестную, то эта дизъюнкция заменяется на $A \vee B$. Заметим, что в случае известного A данное преобразование уже нежелательно, так как устраняет явное указание на альтернативы, относительно которых даются фрагменты ответа. Еще один пример логической перегруппировки, выполняемой из соображений вынесения за скобки утверждений, не содержащих неизвестных, - использование тождества $(A\&B)\vee(C\&((A\&D)\vee E)) = (A\&(B\vee(C\&D)))\vee(C\&E)$. Здесь A - утверждение без неизвестных; C - утверждение, все конъюнктивные члены которого содержат неизвестные. Первый прием срабатывает на уровнях 1 и 7; второй - на уровне 2. Видимо, они не исчерпывают логических перегруппировок в дизъюнкциях, которые могли бы быть полезными при редактировании ответа. Появление именно этих преобразований в решателе объясняется просто тем, что они оказались необходимы в рассмотренных к текущему моменту примерах.

Склейка двух дизъюнкций с общими членами

Если две дизъюнкции $A \vee B$ и $A \vee C$ являются операндами некоторой конъюнкции, то они объединяются в одну дизъюнкцию $A \vee (B\&C)$. Здесь A включает в себя все общие члены дизъюнкций. Уровень срабатывания приема равен 2. Он применяется без каких-либо ограничений, как в условиях, так и в посылках задач.

Вынесение за знак дизъюнкции конъюнктивного члена одного из операндов

Если решается задача на описание с целью "и", то ответ ее должен иметь вид конъюнкции элементарных утверждений. В этом случае предпринимается попытка извлечения из дизъюнкции $A_1 \vee \ldots \vee A_n$ конъюнктивных составляющих. Предварительно проверяется, что задача имеет цель "редакция" и рассматриваемое ее условие, содержащее дизъюнкцию, не имеет неизвестных. Затем просматриваются утверждения A_i. Для текущего такого утверждения просматриваются все его конъюнктивные члены B, представляющие собой элементарные утверждения. Если существует пакетный проверочный оператор, который мог бы распознавать истинность B, то он применяется для усмотрения B в каждой из оставшихся альтернатив A_j. В случае успеха дизъюнкция заменяется на конъюнкцию утверждения B и остаточной дизъюнкции $A'_1 \vee \ldots \vee A'_n$, где A'_j получено из A_j отбрасыванием конъюнктивного члена B, если таковой имелся. Уровень срабатывания приема равен 4.

Раскрывание скобок в условии задачи на преобразование, имеющей цель "днф."

Если задача на преобразование имеет цель "днф", т.е. ее условием является некоторое утверждение, которое надо преобразовать к виду дизъюнктивной нормальной

формы относительно элементарных подутверждений, то применяется преобразование $A\&(B \vee C) = (A\&B) \vee (A\&C)$. Уровень срабатывания приема равен 2.

Дизъюнктивное условие, определяющее в одном из подслучаев явное значение известного параметра

Если условие задачи на описание не содержит неизвестных и имеет вид $(x = t\&A) \vee B_1 \vee \ldots \vee B_n$, где x - переменная, не входящая в терм t, то оно снабжается комментарием "разборслучаев". Вес условия при этом понижается до 0. Прием срабатывает на уровне 1.

Выход из блока анализа для разбора случаев по выведенной дизъюнкции

Если при выводе следствий в блоке анализа Z некоторой задачи на описание Z' возникает дизъюнкция $A_1 \vee \ldots \vee A_n$, содержащая неизвестные либо помеченная комментарием "разборслучаев", то на уровне 2 она инициирует разбор случаев. Предварительно проверяется отсутствие комментариев, блокирующих это действие, а также отсутствие в списке условий задачи Z' утверждений A_i либо дизъюнкции, полученной из данной отбрасыванием части членов.

Чтобы выполнить разбор случаев, сначала происходит перенесение из Z в Z' указанной дизъюнкции, а также, быть может, части сопровождающих ее посылок. Это обеспечивается процедурой "замещениеусловий", которая предпринимает попытку дополнить переносимые из Z в Z' утверждения таким образом, чтобы стало возможным отбросить какие-либо старые условия задачи Z', являющиеся следствиями новых. Процедура используется различными приемами, выполняющими перенесение во внешнюю задачу ответа, полученного в блоке анализа. Она будет описана ниже в специальном подразделе. Перенесенная в Z' дизъюнкция снабжается комментарием "разборслучаев". Затем прием осуществляет выход из блока анализа Z, используя для этого оператор "ответ(Z)". Дальнейшие действия будут выполняться другими приемами. Произойдут разбор случаев по дизъюнктивному условию задачи на описание и переходы к рассмотрению блока анализа в подслучаях A_i, так что прерванный вывод следствий будет продолжен для каждого такого подслучая по отдельности.

Если задача на исследование имеет цель "исследовать", то разбор случаев по выведенной дизъюнкции, снабженной комментарием "разборслучаев", инициируется быстрее - уже на нулевом уровне.

Нормализатор "нормили"

Основная часть приемов нормализатора "нормили" реализована на ГЕНОЛОГе. На ЛОСе реализованы лишь три второстепенных и притом весьма простых приема, которые и приводятся ниже.

1. Устранение отрицания равенства, имеющегося в некотором дизъюнктивном члене. Прием аналогичен приведенному выше приему сканирования задачи. Он отбрасывает отрицание равенства в дизъюнкции $(\neg(x = t)\&A(x) \vee B_1(x) \vee \ldots \vee B_n(x))$ после проверки того, что $A(t)$ вытекает из некоторого $B_i(t)$. Уровень срабатывания приема равен 4.

2. Переобозначение связанных переменных для отождествления кванторов. Если в преобразуемом терме обнаруживаются два отличающихся друг от друга квантора существования, имеющих одинаковую длину, то предпринимается попытка переобозначить связанные переменные так, чтобы кванторы отождествились. Уровень срабатывания приема равен 5. Он играет вспомогательную роль, подготавливая возможность последующих преобразований дизъюнкции.

3. Исключение модуля. Если преобразуемый терм содержит подвыражение вида $\{F(|A|), F(-|A|)\}$, то оно заменяется на $\{F(A), F(-A)\}$. Причина, по которой такой прием пришлось включить в нормализатор "нормили", состоит в том, что условие принадлежности неизвестной конечному списку обычно возникает при его работе после обработки дизъюнкции равенств. Дальнейшее упрощение списка за счет приемов сканирования задачи уже невозможно - ответ будет выдан сразу по окончании применения нормализатора. Поэтому приходится включать в него множество мелких приемов упрощения ответа, срабатывание которых становится достаточно вероятным из-за предшествующей деятельности данного нормализатора.

10.7 Приемы символа "не"

Эти приемы, в основном, крайне просты и обеспечивают лишь общую стандартизацию утверждений.

Двойное отрицание

Утверждение "не(не(A))" заменяется на A. Уровень срабатывания равен 0.

Отрицание дизъюнкции

Утверждение "не(или(A_1 ... A_n))" заменяется на "и(не(A_1) ... не(A_n))". Уровень срабатывания - 0.

Отрицание квантора общности

Утверждение "не(длялюбого($x_1 \dots x_n$ если $A_1 \dots A_n$ то A_0))" заменяется на "существует($x_1 \dots x_n$ и($A_1 \dots A_n$ не(A_0)))". Уровень срабатывания - 0.

Отрицание квантора существования

Утверждение "не(существует($x_1 \dots x_n$ и($A_1 \dots A_n$)))" заменяется на "длялюбого(x_1 ... x_n если $A_1 \dots A_{n-1}$ то не(A_n))". Уровень срабатывания равен 1. Заметим, что в этом преобразовании можно произвольно перегруппировывать утверждения между антецедентами и консеквентом. Прием относит к антецедентам первые конъюнктивные члены. Вообще говоря, это не всегда целесообразно. От способа группировки может зависеть, например, вывод следствий, основанный на использовании кванторной импликации, обратный вывод, и т.п. Поэтому имеются специальные приемы, обеспечивающие перегруппировку утверждений кванторной импликации между антецедентами и консеквентом в соответствии с требованиями контекста.

Отрицание эквивалентности

Если кванторная импликация имеет антецедент вида "не(эквивалентно(A B))", причем консеквент ее не имеет заголовка "эквивалентно", то происходит контрапозиция консеквента с данным антецедентом. В прочих случаях утверждение "не(эквивалентно(A B))" преобразуется к виду "эквивалентно(A не(B))". Уровень срабатывания - 1.

Отрицание конъюнкции

Если преобразовать отрицание конъюнкции "не($A_1 \& \ldots \& A_n$)" в дизъюнкцию отрицаний "не(A_1) $\vee \ldots \vee$ не(A_n)", то может оказаться, что утеряно сопровождение по о.д.з. для некоторых из дизъюнктивных членов "не(A_i)". Именно, это произойдет, если данное сопровождение обеспечивалось другими A_j. Поэтому преобразование применяется с некоторой модификацией. Сначала для каждого A_i проверяется, что все сопровождающие его по о.д.з. утверждения B_i, не вытекающие из контекста отрицания конъюнкции, суть следствия прочих A_j. Тогда при переходе к дизъюнкции отрицание A_i берется в конъюнкции с утверждениями B_i. Эквивалентность такого перехода основана на отсутствии циклов в отношении сопровождения по о.д.з. между различными утверждениями. Если же указанное выше условие не выполняется, то преобразование блокируется, причем вводится комментарий, предотвращающий повторные попытки его применения. Уровень срабатывания приема равен 2. В качестве примера можно привести преобразование утверждения $\neg(x/y = z \ \& \ \neg(y = 0))$. Второй конъюнктивный член является сопровождающим по о.д.з. для первого. Поэтому преобразованное утверждение будет иметь вид $(\neg(x/y = z) \ \& \ \neg(y = 0)) \vee (y = 0)$.

Отрицание условия либо посылки

Если для условия "не(A)" усматривается, что A - посылка либо другое условие, то первое условие заменяется на константу "ложь". Если "не(A)" - посылка, то проверяется наличие условия A, которое заменяется на "ложь". Уровень срабатывания обоих приемов равен 1.

Контрапозиция для условия задачи на доказательство

Если условие задачи на доказательство имеет вид "не(A)", а некоторая посылка - вид "не(B)", то условие заменяется на B, а посылка - на A. Для проверки целесообразности такой контрапозиции применяется справочник "не", обращение к которому происходит на символе, являющемся заголовком утверждения A.

10.8 Приемы символа "длялюбого"

Частота возникновения кванторных конструкций в задачах сильно зависит от рассматриваемой предметной области. Например, в геометрии они встречаются редко; в теоретических задачах по математическому анализу - часто. Типичный способ исключения кванторных конструкций - ввод определений, с помощью которых они сводятся к новым бескванторным утверждениям или выражениям. Так, для кванторной импликации $\forall_x(x \in A \rightarrow x \in B)$ вводится по определению обозначение $A \subseteq B$; для

выражения $\mathrm{set}_x(\exists_y(y \in A \ \& \ x = f(y)))$ - обозначение "образ(f,A)", и т.д. За счет наработки теорем, обслуживающих введенные новые понятия, во многих разделах удается избежать необходимости расшифровки по определениям, и сделать процессы решения задач почти "бескванторными". Однако, остается достаточное количество случаев, когда исключение кванторных конструкций невозможно, и для работы с ними приходится создавать множество специальных приемов. На долю квантора общности приходится наибольшее число таких приемов.

Простейшая стандартизация

Сначала перечислим приемы, срабатывающие на нулевом уровне.

1. Переобозначение связанных переменных. Если в задаче встречается квантор или описатель, то связанные переменные его немедленно переобозначаются так, чтобы они не имели свободных вхождений в утверждения, образующие внешний контекст для данного квантора или описателя. Например, если внешний контекст кванторного утверждения $\forall_x P(x)$ содержит равенство $x = 2$, то это утверждение будет заменено на $\forall_y P(y)$. Такое преобразование выполняется оператором "нормализациясвязок". Возникающая после переобозначений стандартизация гарантирует, что на уровнях, больших или равных 1, никакая связанная переменная не будет совпадать с обозначением объекта, уже имеющимся в текущем контексте.

2. Устранение из кванторной приставки дублирующих переменных. Если кванторная приставка имеет переменные, встречающиеся более одного раза, то повторные вхождения переменных из нее исключаются.

3. Устранение дублирующих антецедентов. Если имеются повторяющиеся антецеденты, то они исключаются.

4. Исключение из кванторной приставки переменных, не имеющих свободных вхождений в антецеденты и консеквент. Если некоторые переменные связывающей приставки избыточны - не имеют свободных вхождений ни в антецеденты, ни в консеквент, то они удаляются. В случае, когда связывающая приставка остается пустой, происходит замена кванторной импликации $\forall_{x_1 \ldots x_n}(A_1 \ \& \ldots \& \ A_n \to A_0)$ на дизъюнкцию $\neg A_1 \vee \ldots \vee \neg A_n \vee A_0$.

5. Устранение дизъюнкции в консеквенте. Если кванторная импликация имеет вид $\forall_{x_1 \ldots x_n}(A_1 \ \& \ldots \& \ A_n \to (B_1 \vee \ldots \vee B_m))$, то она преобразуется к виду $\forall_{x_1 \ldots x_n}(A_1 \ \& \ldots \& \ A_n \ \& \ \neg B_1 \ \& \ldots \ \& \ \neg B_{m-1} \to B_m)$. Исключение составляет случай редактирования теоремы, специально предназначенной для указания списка альтернатив.

6. Замена конъюнктивного антецедента на группу антецедентов. Если некоторый антецедент представляет собой конъюнкцию утверждений, то он исключается, а все его конъюнктивные члены становятся новыми антецедентами.

7. Усмотрение истинности кванторной импликации, у которой антецедент совпадает с консеквентом либо имеются два противоположных антецедента. В указанной ситуации кванторная импликация заменяется на константу "истина".

8. Исключение антецедента, отрицание которого совпадает с консеквентом.

9. Усмотрение антецедентов, указывающих несовместимые типы одной и той же переменной. Если кванторная импликация имеет антецеденты $P(x)$ и $Q(x)$, где P, Q - названия различных типов объектов, причем с помощью справочника "род" усматривается, что ни один из типов не является подтипом другого, то эта импликация заменяется на константу "истина".

Далее идут приемы стандартизации кванторных импликаций, уровень срабатывания которых больше нуля.

1. Устранение конъюнкции в консеквенте. Если кванторная импликация имеет вид $\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\&\, A_n \to (B_1 \,\&\ldots\&\, B_m))$, то ее можно преобразовать к виду конъюнкции кванторных импликаций

$$\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\&\, A_n \to B_1) \,\&\ldots\&\, \forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\&\, A_n \to B_m).$$

   Это преобразование, вообще говоря, полезно, так как упрощает кванторные конструкции, хотя и увеличивает их число. Однако, есть ряд ограничений на его применение. Во-первых, нецелесообразно применять его к условию задачи на доказательство, так как выгоднее сначала применить другое преобразование, исключающее кванторную импликацию за счет переброски антецедентов в список посылок (см. ниже). Во-вторых, при преобразовании условия задачи на описание нужно контролировать сохранение сопровождения по о.д.з. Если какое-то B_i обеспечивало о.д.з. для другого B_j, то их невыгодно разбивать по разным кванторным импликациям. Поэтому в данном случае преобразование модифицируется, аналогично преобразованию отрицания конъюнкции. Находятся все B_i, не используемые для сопровождения по о.д.з. других B_j, и утверждения $B_1, \ldots, B_m$ распределяются по группам, образованным указанными B_i и сопровождающими их по о.д.з. другими B_k. Эти группы могут пересекаться. Затем исходная импликация заменяется на конъюнкцию импликаций, соответствующих найденным группам. Уровень срабатывания приема равен 1.

2. Устранение дизъюнкции в антецеденте. Если кванторная импликация имеет вид $\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\&\, A_{i-1} \,\&(B_1 \vee \ldots \vee B_m) \,\&\, A_{i+1} \,\&\ldots\&\, A_n \to A_0)$, то она заменяется на конъюнкцию импликаций $\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\&\, A_{i-1} \,\&\, B_1 \,\&\, A_{i+1} \,\&\ldots\&\, A_n \to A_0) \,\&\ldots\&\, \forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\&\, A_{i-1} \,\&\, B_m \,\&\, A_{i+1} \,\&\, \ldots\&\, A_n \to A_0)$. Прием применяется без ограничений, и уровень срабатывания его равен 1.

3. Усмотрение вхождения антецедента в консеквент либо в другой антецедент.

   Если некоторый антецедент кванторной импликации входит в другой ее антецедент либо в консеквент, то это вхождение заменяется на константу "истина". Если антецедент имеет вид "не(A)", причем утверждение A входит в другой антецедент либо в консеквент, то вхождение этого утверждения заменяется на константу "ложь". Уровень срабатывания равен 1.

4. Усмотрение вхождения консеквента в антецедент. Если консеквент кванторной импликации входит в ее антецедент, то это вхождение заменяется на константу "ложь". Если консеквент имеет вид "не(A)", причем утверждение A входит в антецедент, то его вхождение заменяется на константу "истина". Уровень срабатывания равен 1.

5. Сдвоенные переменные кванторной приставки. Если кванторная импликация $\forall_{x_1 \ldots x_k}(A_1 \,\&\, \ldots \,\&\, A_n \to A_0)$ имеет такие две различные переменные x_i, x_j, которые повсюду встречаются вместе под одной и той же операцией f, кроме, быть может, антецедентов, указывающих тип значения этих переменных по отдельности, то предпринимается попытка перейти от пары переменных x_i, x_j к новой переменной, обозначающей $f(x_i, x_j)$. Для коммутативной и ассоциативной операции f проверяется, что как x_i, так и x_j присутствуют среди ее операндов лишь однократно. Для не коммутативной операции f проверяется, что она двуместная, причем операнды x_i, x_j встречаются везде в одном и том же порядке. Чтобы определить условия на допустимые значения новой переменной z, обозначающей $f(x_i, x_j)$, решается вспомогательная задача, условия которой суть $z = f(x_i x_j)$, $P(z)$, где P - тип значений операции f, а также все антецеденты, имеющие единственную переменную x_i либо единственную переменную x_j. Неизвестные задачи суть переменные x_i, x_j. На ответ навешивается квантор существования по x_i, x_j, и результат упрощается. Найденные условия на z берутся в качестве антецедентов, заменяющих ранее имевшиеся антецеденты, содержащие x_i, x_j по отдельности. Все сдвоенные вхождения x_i, x_j в антецеденты и консеквент заменяются на z; в связывающей приставке проводится такая же замена. Фактически прием даже не переходит к новой переменной z, а использует в качестве нее старую переменную x_i. Уровень срабатывания равен 2.

6. Лексикографическое упорядочение антецедентов. По завершении описанных выше преобразований общей стандартизации кванторной импликации предпринимается лексикографическое переупорядочение ее антецедентов. Уровень срабатывания приема равен 3. Такое переупорядочение может быть необходимым для усмотрения совпадения двух различных импликаций.

Декомпозиция кванторной импликации на независимые утверждения

Утверждения $A_1, \ldots, A_n, \neg A_0$ кванторной импликации $\forall_{x_1 \ldots x_k}(A_1 \,\&\, \ldots \,\&\, A_n \to A_0)$ могут распадаться на такие подмножества $B_1, \ldots, B_r$, что элементы различных подмножеств не зависят от общих переменных кванторной приставки $x_1, \ldots, x_k$. В этом случае становится возможной декомпозиция кванторной импликации в дизъюнкцию некоторых утверждений $C_1, \ldots, C_r$. Число r выбирается наибольшим возможным, $r > 1$, что соответствует выбору в качестве каждого B_i одной компоненты связности исходного списка утверждений для отношения зависимости двух утверждений от общей переменной кванторной приставки. C_i есть кванторная импликация, полученная из элементов подмножества B_i выбором в качестве консеквента отрицания одного из них, а в качестве антецедентов - оставшихся элементов. Кванторная приставка у C_i представляет собой подмножество K_i переменных кванторной приставки исходной импликации, встречающихся в B_i. Если K_i пустое, то B_i, очевидно, одноэлементно, и тогда C_i является отрицанием этого элемента. Прием применяется на уровнях 0 либо 2. Уровень 0 имеет место для задач на описание, имеющих цель "кванторная-свертка". Здесь необходимо успеть выполнить декомпозицию до того, как кванторная импликация будет преобразована "по определениям" в бесквантное утверждение. При наличии в кванторной импликации логических констант прием блокируется, так как целесообразно сначала от них избавиться.

Для получения подмножеств B_i используется процедура "компонента(x1 x2 x3)", при

обращении к которой задаются набор термов x2 и два списка переменных - x1 и x3. x1 указывает свободные переменные некоторого терма T набора x2. Процедура находит содержащую T компоненту связности в x2, состоящую их всех термов, к которым можно перейти от T по цепочке переходов между термами, зависящими от общей переменной списка x3.

Выбор консеквента в кванторной импликации C_i, вообще говоря, неоднозначен. Прием делает некоторый предварительный выбор, руководствуясь рядом эвристических соображений. В частности, он пытается сохранить старый консеквент, если таковой попал в C_i, и сохранить в антецедентах простейшие утверждения, обеспечивающие о.д.з. для более сложных утверждений. При неудачном выборе консеквента необходимые перестановки будут выполнены другими приемами.

Контрапозиции

Приведем несколько общих приемов, переставляющих местами консеквент и антецедент кванторной импликации (разумеется, переходя при этом к их отрицаниям). Такие преобразования называются контрапозициями.

1. Консеквент и антецедент начинаются с символа "не". В этом случае обычно выполняется контрапозиция, позволяющая отбросить оба отрицания. Однако, в ряде специальных случаев применение приема блокируется. Например, если консеквент имеет вид отрицания равенства, содержащего неизвестные, а антецедент неизвестных не содержит, либо если антецедент необходим для обеспечения сопровождения по о.д.з. Уровень срабатывания приема равен 0.

2. Консеквент имеет вид "не($P(x)$)", где x - переменная кванторной приставки, а P - логический символ. Если некоторый антецедент не имеет вида $Q(y)$, где y - переменная кванторной приставки, а Q - логический символ, то для него выполняется контрапозиция. Уровень срабатывания равен 1.

3. Консеквент имеет вид "не($P(x)$)", и никакой антецедент не имеет своим заголовком символа "не". Составляется список S антецедентов, пополненный утверждением $P(x)$. Находится подмножество R утверждений списка S, не используемых в качестве сопровождающих по о.д.з. для других утверждений данного списка. Если R состоит из единственного элемента Q, отличного от $P(x)$, то выполняется контрапозиция для антецедента Q. Если R более чем одноэлементно, то контрапозиция выполняется для такого его элемента Q, который либо является кванторной импликацией, в то время как $P(x)$ элементарно, либо имеет наибольшее число входящих в него переменных кванторной приставки, превосходящее такое число для $P(x)$.

4. Некоторый антецедент имеет вид "не(P)", а консеквент Q имеет тот же заголовок, что и утверждение P. Если список свободных переменных консеквента включается в список свободных переменных утверждения P, длина терма Q хотя бы в 1.2 раза меньше длины терма P, и антецедент "не(P)" не используется для сопровождения по о.д.з., то выполняется контрапозиция.

Слияние кванторных приставок

Если кванторная импликация имеет вид "длялюбого($x_1 \dots x_n$ длялюбого($y_1 \dots y_m$ если $A_1 \dots A_k$ то A_0))", то она преобразуется к виду "длялюбого($x_1 \dots x_n y_1 \dots y_m$ если $A_1 \dots A_k$ то A_0)". Уровень срабатывания равен 0.

Группировка кванторов наружу

На уровне 1 выполняется обобщающая предыдущий случай группировка кванторов наружу. Именно, если кванторная импликация имеет вид "длялюбого($x_1 \dots x_n$ если $B_1 \dots B_p$ то длялюбого($y_1 \dots y_m$ если $A_1 \dots A_k$ то A_0))", то она заменяется на "длялюбого($x_1 \dots x_n y_1 \dots y_m$ если $B_1 \dots B_p A_1 \dots A_k$ то A_0)". Аналогично, утверждение "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_{i-1}$ существует($y_1 \dots y_m$ B) $A_{i+1} \dots A_k$ то A_0)" заменяется на "длялюбого($x_1 \dots x_n y_1 \dots y_m$ если $A_1 \dots A_{i-1} B A_{i+1} \dots A_k$ то A_0)". В последнем случае требуется, чтобы рассматриваемый терм задачи не содержал выражения "отображение($y_1 \dots y_m$ B t)".

Исключение избыточных антецедентов

Несколько простых приемов позволяют отбрасывать антецеденты, не изменяющие области истинности кванторной импликации. Обычно такие приемы бывают нужны при выводе теорем.

1. Отбрасывается антецедент $P(x)$, где одноместный предикатный символ P определяет тип объекта, а связанная переменная x не встречается в других антецедентах и в консеквенте. Уровень срабатывания равен 0.

2. Если некоторая переменная кванторной приставки x не встречается в консеквенте, то рассматриваются все содержащие ее антецеденты $A_1, \dots, A_n$. Если из оставшихся антецедентов $B_1, \dots, B_m$ вытекает существование такого значения x, что все утверждения $A_1, \dots, A_n$ истинны, то последние отбрасываются из списка антецедентов. Для проверки здесь используется вспомогательная задача на описание, решаемая с сильным ограничением на допустимую трудоемкость. Прием применяется только при выводе теорем - для его активизации требуется цель "редуцирование". Уровень срабатывания равен 5.

3. Если антецедент кванторной импликации, возникающей при выводе теорем, имеет вид $\neg(x = t)$, где x - переменная кванторной приставки, не входящая в t, то проверяется, что теорема сохраняет истинность и в случае $x = t$. После этого антецедент отбрасывается. Уровень срабатывания равен 5.

Замена переменной кванторной приставки

Пусть кванторная импликация "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то A_0)" имеет переменную x_i своей кванторной приставки, которая встречается в консеквенте только внутри вхождений одного и того же выражения T, причем все содержащие x_i антецеденты не имеют переменных, отличных от x_i. В качестве T берется минимальное выражение, корневым операндом которого служит x_i, причем проверяется, что число вхождений T в консеквент не менее двух и что ни одна из свободных переменных T не связана никаким внешним квантором или описателем, находящимся в консеквенте.

В указанной ситуации предпринимается попытка заменить переменную x_i на новую переменную y, являющуюся вспомогательным обозначением для T. Чтобы найти условия на область изменения новой переменной, преобразуется к бесквантерному виду утверждение "существует(x_i и(Q $y = T$))", где Q - конъюнкция содержащих x_i антецедентов. Преобразование выполняется в два этапа - сначала решается задача на описание с условиями Q, $y = T$ и несущественной неизвестной x_i, затем упрощается квантор существования. Если результат упрощения имеет вид $P_1 \vee \ldots \vee P_k$, $k \geq 1$, то проверяется, что каждый содержащий y конъюнктивный член каждого P_j не содержит других переменных. Далее вводятся кванторные импликации C_j, $j = 1, \ldots, k$, полученные из исходной кванторной импликации заменой в консеквенте всех вхождений выражения T на y, заменой всех содержащих x_i антецедентов на P_j и заменой в кванторной приставке x_i на y. Наконец, предпринимается замена исходной кванторной импликации на конъюнкцию результатов упрощения кванторных импликаций $C_1, \ldots, C_k$. Уровень срабатывания равен 4.

Переформулировка кванторной импликации как условия пустоты класса

Если в одном и том же терме задачи присутствуют кванторная импликация "длялюбого($x_1 \ldots x_n$ если $A_1 \ldots A_m$ то A_0)" и выражение "отображение($x_1 \ldots x_n$ P t)", причем отрицание кванторной импликации совпадает, с точностью до простейших преобразований, с утверждением "существует($x_1 \ldots x_n$ P)", то кванторная импликация преобразуется к виду "равно(класс($x_1 \ldots x_n$ P)пусто)". Уровень срабатывания равен 4.

Импликативное условие задачи на доказательство

Если задача на доказательство Z имеет условие вида "длялюбого($x_1 \ldots x_n$ если A_1 $\ldots A_m$ то A_0)", то она сводится к новой задаче на доказательство Z', получаемой из текущей добавлением утверждений $A_1, \ldots, A_m$ к списку посылок и выбором утверждения A_0 в качестве условия. Комментарии к посылкам и комментарии к задаче переносятся из Z в Z' с помощью справочников "новаяпосылка", "длялюбого". Задача Z' решается с тем же максимальным уровнем, что и Z. Результат ее решения (ответ "истина" либо символ "отказ") выдается как ответ на задачу Z. Уровень срабатывания равен 3.

Контрапозиция кванторной импликации в посылках задачи на доказательство и ее условия, имеющего вид отрицания

Если задача на доказательство Z имеет единственную кванторную импликацию A в посылках, а условие ее имеет вид "не(B)", то она сводится к новой задаче Z', полученной заменой посылки A на B, а условия - на отрицание A. Целью такой контрапозиции является получение в условии задачи квантора существования для последующего перехода от задачи на доказательство к задаче на подбор примера. Уровень срабатывания приема равен 6.

Обратный вывод в задаче на доказательство

Если задача на доказательство имеет своей посылкой кванторную импликацию "длялюбого($x_1 \ldots x_n$ если $A_1 \ldots A_m$ то A_0)", у которой все переменные связывающей при-

ставки входят в консеквент A_0, а условием задачи служит элементарное утверждение B с тем же заголовком, что A_0, отличным от равенства, то проверяется, не является ли B результатом подстановки в A_0 некоторых термов $t_1, \ldots, t_n$ вместо переменных $x_1, \ldots, x_n$. Если такие термы найдены, то последовательно рассматриваются результаты B_i подстановки их вместо переменных кванторной приставки в антецеденты $A_1, \ldots, A_m$. Для каждого B_i предпринимается попытка усмотреть его истинность в контексте посылок текущей задачи, используя вспомогательную задачу на доказательство с замороженными посылками и с достаточно сильным ограничением на трудоемкость. Таким образом, реализуется цикл попыток быстрого усмотрения. В случае его успешного завершения на текущую задачу выдается ответ "истина". Уровень срабатывания приема равен 5.

Использование кванторной импликации, имеющей заголовком консеквента квантор существования, для доказательства существования

Если задача на доказательство имеет посылку вида "длялюбого($x_1 \ldots x_n$ если $A_1 \ldots A_m$ то существует($y_1 \ldots y_k$ и($B_1 \ldots B_r$)))", а условие ее имеет вид "существует($z_1 \ldots z_p$ и($C_1 \ldots C_q$))", то предпринимается попытка подобрать такую подстановку термов $s_1, \ldots, s_p$ вместо переменных $z_1, \ldots, z_p$ в утверждения $C_1, \ldots, C_q$, которая дает подмножество результатов подстановки некоторых термов $t_1, \ldots, t_n$ вместо переменных $x_1, \ldots, x_n$ в утверждения $B_1, \ldots, B_r$. Для этого используется процедура "унификация". В результате оказываются найдены конкретные значения переменных квантора существования - условия задачи, отождествляющие подкванторные утверждения с некоторыми конъюнктивными членами консеквента кванторной импликации. Далее остается только проверить истинность утверждений, получающихся при подстановке в антецеденты термов $t_1, \ldots, t_n$ вместо переменных $x_1, \ldots, x_n$. Это делается с помощью оператора "извлекается", решающего вспомогательную задачу на доказательство с ограниченными средствами, обеспечивающими попытку быстрого усмотрения. Уровень срабатывания данного приема равен 1. Увеличение его привело бы к тому, что прием оказался практически отключенным: на уровне 2 срабатывает прием, сводящий доказательство существования к задаче на подбор примера.

Разрешение подкванторных утверждений относительно переменных кванторной приставки

В некоторых случаях подкванторные утверждения допускают явное разрешение относительно переменных кванторной приставки как неизвестных. Обычно, вслед за этим удается вообще избавиться от данного квантора, переформулировав его в терминах элементарных утверждений. Для удобства кванторная импликация преобразуется к виду отрицания квантора существования, и рассматриваются подкванторные утверждения последнего. Например, если условие задачи на описание имеет вид $\forall_x(1 \leq x \;\&\; x \leq 3 \rightarrow x + a < 2)$, то можно решить неравенства $1 \leq x$, $x \leq 3$, $\neg(x + a < 2)$ относительно несущественной неизвестной x. В процессе решения будут исключаться явные указания интервалов допустимых значений x и сохраняться только условия на параметры, при которых эти значения существуют. Получив ответ $a \geq -1$, преобразуем исходную кванторную импликацию к виду $\neg\exists_x(a \geq -1)$, что далее дает бескванторное условие $a < -1$.

Приведенный прием часто бывает полезен в задачах по элементарной алгебре; в других разделах он обычно мало эффективен и приводит лишь к неоправданным

дополнительным затратам времени. Поэтому начинается прием с проверки серии эвристических фильтров, ограничивающих его активизацию.

Прием используется только для кванторных импликаций "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то A_0)", являющихся условиями задач на описание. Проверяется, что такая кванторная импликация не является частью ответа, задающей серию ограничений. Находится список утверждений $A_1, \dots, A_m, \neg A_0$, и решается вспомогательная задача на описание с данным списком условий. Посылками ее служат все утверждения из контекста рассматриваемой кванторной импликации; несущестенными неизвестными - переменные $x_1, \dots, x_n$. Если найден ответ B, то кванторная импликация заменяется на утверждение $\neg\exists_{x_1 \dots x_n} B$. Уровень срабатывания приема равен 3.

Фиксация значения связанной переменной через равенство в антецеденте

Кванторная импликация вида "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_{i-1}\ x_j = t\ A_{i+1} \dots A_m$ то A_0)", где выражение t не содержит x_j, за исключением крайне редких специальных случаев, преобразуется к виду "длялюбого($x_1 \dots x_{j-1} x_{j+1} \dots x_n$ если $B_1 \dots B_{i-1} B_{i+1} \dots B_m$ то B_0)". Здесь каждое B_k - результат подстановки выражения t вместо переменной x_j в утверждение A_k. Чтобы преобразование было корректным, связанные переменные утверждения A_k переобозначаются на переменные, отличные от свободных переменных терма t. Уровень срабатывания приема равен 0.

Фиксация значения связанной переменной через отрицание равенства в консеквенте

Этот прием аналогичен предыдущему. Он заменяет кванторную импликацию вида "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то не$(x_j = t)$)" на импликацию "длялюбого($x_1 \dots x_{j-1} x_{j+1} \dots x_n$ если $B_1 \dots B_{m-1}$ то не(B_m))". Уровень срабатывания его тоже равен 0.

Вывод следствий с помощью кванторной импликации

Для вывода следствий в посылках некоторой задачи можно использовать кванторную импликацию "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то A_0)", содержащуюся в тех же посылках. Имеется несколько приемов, осуществляющих такой вывод:

1. Вывод без контрапозиций. В случае задачи на доказательство уровень срабатывания приема равен 2, иначе он равен 3. Проверяется, что все утверждения $A_1, \dots, A_m, A_0$ элементарны (не содержат логических сязок, кроме, быть может, внешнего отрицания, и не имеют связанных переменных). Проверяется также отсутствие некоторых специальных комментариев, делающих использование данной кванторной импликаций для прямого вывода следствий нецелесообразным. После этого предпринимается обращение к процедуре "подборзначений", которая пытается подобрать такие термы $t_1, \dots, t_n$, подстановка которых в антецеденты дает (с точностью до простейших преобразований) некоторые посылки $B_1, \dots, B_n$. Так как целью вывода является получение утверждений, связывающих между собой переменные задачи, то проверяется, что указанные посылки имеют хотя бы одну свободную переменную. Проверяется, что каждая посылка B_i либо имеет свободную переменную, встречающуюся в условии задачи, либо относится к той же компоненте списка посылок, определяемой

отношением зависимости двух утверждений от общей переменной, что и рассматриваемая кванторная импликация. После этого определяется результат C подстановки термов $t_1, \ldots, t_n$ вместо переменных $x_1, \ldots, x_n$ в консеквент A_0. Предпринимается упрощение утверждения C, в котором участвуют нормализаторы общей стандартизации и (в случае равенства) оператор "стандравно". Те подтермы данного утверждения, которые уже встречаются в посылках задачи, упрощением не затрагиваются. Если в результате упрощения получено утверждение D, отличное от константы "истина", то выполняется обращение к оператору "фильтримпликации" для окончательного определения целесообразности вывода следствия D.

Здесь мы сталкиваемся с явлением, характерным для многих общих приемов. Ввиду их общности, иногда бывает затруднительно дать какие-то простые рекомендации относительно того, в каких случаях прием применять нужно, а в каких - не нужно. Чтобы обеспечить хорошее управление приемом, приходится вводить в него накопитель эвристических решающих правил, имеющих существенно меньший уровень общности, чем логическая схема выполняемых преобразований. Эти правила могут быть связаны с понятиями различных предметных областей, и тогда для объединения их в общую процедуру применяются справочники. После того, как ситуация конкретизируется, обычно удается найти достаточно хорошие мотивировки для разрешения либо блокировки срабатывания. В нашем случае несколько различных решающих правил, накопленных при обучении системы, сгруппированы в процедуре "фильтримпликации". Сразу заметим, что для тех разделов, по которым велось обучение решателя, кванторные импликации возникали в посылках исключительно редко. Поэтому обучающий материал для создания фильтров данного приема оказался небольшим. Это привело к тому, что длительное время достаточными являлись самые простые средства отсечения. Разрешалось выводить следствия, у которых новые подвыражения не превосходили по своей длине подвыражений, уже имеющихся в задаче. Лишь при переходе к рассмотрению простых "теоретических" задач по математическому анализу данный прием стал порождать чрезмерное количество бесполезных следствий, и процедура "фильтримпликации" пополнилась дополнительными ограничителями, относящимися к вводу функциональных переменных (т.е. термов вида "значение(f t)"). Эти ограничители достаточно грубые и впоследствии должны быть существенно уточнены.

Перейдем к перечислению решающих правил процедуры "фильтримпликации":

(a) Если утверждение D содержит подвыражение "значение(f t)", причем текущая задача на описание имеет цель "пример", и некоторое ее условие также содержит подвыражение вида "значение(f s)", не связанное внешним квантором либо описателем, а посылки такого вида нет, то принимается решение о выводе следствия D.

(b) Если предыдущая ситуация не имеет места, причем D содержит подвыражение "значение(f t)", где t имеет вхождение символа "значение", то вывод блокируется. Хотя и несколько искусственное, это ограничение должно восприниматься в контексте наличия других приемов, специально предназначенных для вывода следствий с подвыражениями типа "значение(...)". Поэтому для данного приема оставлены лишь простейшие ситуации вывода следствий, содержащих такие подвыражения.

(c) Если D содержит подвыражение "значение($f\ t$)", которое до этого не встречалось в задаче, то проверяется наличие такой функциональной переменной g, что выражения "значение(f ...)" входят в отличные от рассматриваемой кванторной импликации термы задачи только внутри выражений "значение(g ...)", причем хотя бы одно такое вхождение имеется. В этой ситуации вывод следствия D блокируется.

(d) Если D содержит подвыражение "значение(...)", не встречающееся в условиях и посылках текущей задачи на описание, имеющей цель "пример", причем условие из пункта a) не выполнено, то вывод блокируется.

(e) Если решается задача на описание, имеющая цель "пример", либо задача на доказательство, максимальный уровень которой более 6, причем длина каждого подвыражения утверждения D не превосходит длины хотя бы какого-то подвыражения посылки либо условия задачи, то принимается решение о выводе следствия D.

(f) Если каждое подвыражение утверждения D уже имеется в посылках и условиях задачи, то принимается решение о выводе следствия D.

(g) Если кванторная импликация снабжена комментарием "то", блокирующим ограничения процедуры "фильтримпликации", то принимается решение о выводе следствия D. Однако, после получения с помощью рассматриваемой кванторной импликации более 4 следствий, комментарий "то" к ней удаляется.

(h) Если решается задача на исследование, имеющая цель"известно", причем D содержит подвыражение специального вида ("значение(...)" либо числовой атом), содержащее неизвестные и уже имеющееся в некоторой посылке задачи, в которую входят также неизвестные внешней задачи на описание, то принимается решение о выводе следствия D.

Если процедура "фильтримпликации" принимает решение о выводе следствия D, то оно регистрируется в списке посылок. При этом веса всех посылок и условий, имеющих общую свободную переменную с D, уменьшаются до 1.

2. Вывод с контрапозициями.

   Уровень срабатывания приема равен 5. Здесь перед использованием кванторной импликации для вывода следствий разрешается выполнить контрапозицию какого-либо ее антецедента (поменять его местами с консеквентом, заменив оба утверждения на их отрицания). Технически это достигается путем составления списка $A_1, \ldots, A_m, \neg A_0$ и обращением к процедуре "подборпосылок", которая пытается идентифицировать с посылками задачи все утверждения списка, кроме какого-либо одного. На основе отрицания последнего и строится следствие D. Это делается аналогично предыдущему приему. Фильтрация выводимых утверждений D усиливается - пропускаются лишь те утверждения, все подвыражения которых уже встречались в задаче. Соответственно, обращение к процедуре "фильтримпликации" для данного приема становится не нужным. Уменьшение весов посылок и условий не предпринимается.

3. Вывод для получения утверждения с функциональным выражением, уже имеющимся в задаче. Если решается задача на исследование, посылка которой

содержит функциональное выражение "значение(f t)", зависящее от неизвестных внешней задачи на описание, то могут предприниматься попытки вывода следствий, содержащих данное функциональное выражение. Прием, выполняющий такие попытки, инициируется при обнаружении в консеквенте A_0 кванторной импликации подвыражения "значение(f s)", где f - переменная. Он проверяет, что все переменные кванторной приставки содержатся в выражении s, выбирает в некоторой элементарной посылке текущей задачи на исследование подвыражение "значение(f t)", где t зависит от неизвестных внешней задачи на описание, и находит такие термы $r_1, \ldots, r_n$, подстановка которых вместо переменных $x_1, \ldots, x_n$ в выражение s дает выражение t. Находятся результаты $B_1, \ldots B_m, B_0$ подстановки термов $r_1, \ldots, r_n$ вместо переменных $x_1, \ldots, x_n$ в антецеденты и консеквент кванторной импликации. Используя проверочные операторы, прием пытается установить истинность утверждений $B_1, \ldots B_n$ в контексте списка посылок текущей задачи. Здесь устанавливается достаточно сильное ограничение на допустимую трудоемкость. В случае успеха следствие B_0 регистрируется в посылках задачи. Уровень срабатывания приема равен 4.

4. Вывод равенства для числового атома. Если решается задача на исследование, и консеквент кванторной импликации представляет собой тождество $F(a_1 \ldots a_k) = t$, где F - некоторая числовая характеристика нечисловых объектов (длина, масса, и т.п.), то предпринимается попытка установить с его помощью значения данной характеристики, встречающиеся в прочих посылках задачи. Для любого вхождения в такую посылку подтерма $F(b_1 \ldots b_k)$, не связанного внешними кванторами и описателями, находится подстановка p вместо переменных кванторной приставки, переводящая $a_1, \ldots, a_k$ в $b_1, \ldots, b_k$. Проверяется, что истинность результатов применения p к антецедентам усматривается из посылок задачи с помощью проверочных операторов, и регистрируется следствие $F(b_1 \ldots b_k) = s$, где s - результат применения p к терму t. Уровень срабатывания приема равен 4.

5. Попытка вывести следствие, содержащее уже встречавшееся в задаче выражение с неизвестными. Если решается задача на исследование, причем длина кванторной приставки больше 1, то рассматривается некоторое более чем однобуквенное подвыражение t ее консеквента, заголовок которого не ассоциативен и отличен от символа "значение". Это выражение берется максимальным в том смысле, что является непосредственным операндом некоторого утверждения. Проверяется, что пересечение S свободных переменных выражения t с кванторной приставкой непусто и что непуст список R его свободных переменных, не входящих в кванторную приставку. Затем ищется такая элементарная посылка F, которая имеет неизвестное подвыражение T с тем же заголовком, что у выражения t. Проверяется, что все переменные списка R содержатся в T и что посылка F не имеет вида $T = a$, где a известно. Находятся такие термы Q, подстановка которых в t вместо переменных списка S дает выражение T. Определяются результаты $C_1, \ldots, C_m$ применения данной подстановки к антецедентам $A_1, \ldots, A_m$. Находится список D всех переменных кванторной приставки $x_1, \ldots, x_n$, входящих в утверждения $C_1, \ldots, C_m$, и проверяется, что он непуст. Решается вспомогательная задача на описание Z', условиями которой служит утверждения $C_1, \ldots, C_m$, посылками - все посылки текущей задачи на исследование, а неизвестными - переменные списка D. Эта задача имеет также цели "полный", "явное", "прямойответ", "и", "упростить". При обращении к

ней устанавливается достаточно сильное ограничение на трудоемкость. Если в результате решения возникает система равенств, явно определяющих значения переменных списка D, то находится результат применения к консеквенту A_0 объединенной подстановки вместо переменных списков S, D. Этот результат регистрируется приемом как новое следствие, выведенное с помощью кванторной импликации. Он содержит ранее встречавшееся в задаче неизвестное выражение T и может оказаться полезным для дальнейшего. Уровень срабатывания приема равен 5.

Использование кванторной посылки задачи на описание для усмотрения истинности условия

Если решается задача на описание, в которой не требуется получить полный ответ, то кванторная импликация K в ее посылках может быть использована для усмотрения истинности элементарного условия F. Проверяется, что F является результатом подстановки p в консеквент импликации K некоторых термов вместо переменных кванторной приставки. После этого предпринимается цикл проверок истинности результатов применения той же подстановки p к антецедентам импликации. Для проверок используются проверочные операторы. В случае успеха условие F заменяется на константу "истина". Уровень срабатывания приема равен 5.

Использование равенства функциональной переменной константе

Если кванторная импликация имеет консеквент вида $f(t) = p$, где выражение t содержит переменные кванторной приставки, а выражение p - не содержит этих переменных, то предпринимается попытка заменить на p все представимые в виде $f(t)$ подвыражения, к контексту которых относится кванторная импликация. Перед каждой заменой проверяется истинность антецедентов в соответствующем подслучае. Уровень срабатывания приема равен 2.

Импликативное условие задачи на описание, не требующей полного ответа

Для устранения кванторных импликаций в условиях задачи на описание нет каких-то простых общих приемов. В задачах по элементарной алгебре они обычно исключаются за счет разрешения подкванторных утверждений относительно переменных кванторной приставки. Однако, в других разделах (например, в математическом анализе) такое разрешение редко бывает возможным. Чтобы избавиться от кванторной импликации, иногда удается использовать новые понятия, имеющие кванторные определения. Целесообразность перехода к таким понятиям целиком зависит от того, насколько развит аппарат решения задач, сформулированных на их языке.

Новые возможности для работы с кванторными импликациями в условиях задачи на описание появляются, если не требуется получить полный ответ. Тогда можно применять различные разновидности обратного вывода, подбирая такие бескванторные дополнительные условия задачи, при наличии которых консеквент импликативного условия становится следствием его антецедентов. Приведем несколько приемов, реализующих этот обратный вывод. Рассматриваемое импликативное условие обозначаем далее посредством $\forall_{x_1 \ldots x_n}(A_1 \,\&\, \ldots \,\&\, A_m \to A_0)$.

1. Попытка отождествления консеквента с антецедентом либо получения двух противоположных антецедентов. Пусть текущая задача на описание Z не имеет цели "полный" либо имеет цель "пример", причем консеквент A_0 и некоторый антецедент A_i элементарны. Если одно из утверждений A_0, A_i содержит неизвестные, а другое нет, причем то из них, которое "известно", представляет собой (с точностью до простейших преобразований) результат подстановки в "неизвестное" не содержащих переменных кванторной приставки термов $t_1, \ldots, t_k$ вместо неизвестных $y_1, \ldots, y_k$, то решается вспомогательная задача Z', полученная из Z заменой импликативного условия на $y_1 = t_1 \,\&\ldots\&\, y_k = t_k$. В случае успеха найденный ответ выдается как ответ на задачу Z.

   Аналогичным образом, рассматриваются два антецедента A_i, A_j, один из которых содержит неизвестные, а другой - нет. Если отрицание "известного" представляет собой результат подстановки в "неизвестное" не зависящих от переменных кванторной приставки термов $t_1, \ldots, t_k$ вместо неизвестных $y_1, \ldots, y_k$, то далее, как и выше, рассматривается вспомогательная задача Z'.

   Оба случая объединены в один прием, уровень срабатывания которого равен 9.

2. Исключение импликативного условия без неизвестных в антецедентах с помощью вспомогательной задачи на описание, имеющей цель "независит". Пусть все антецеденты $A_1, \ldots, A_m$ не содержат неизвестных текущей задачи на описание Z, а консеквент A_0 содержит. Если эта задача не имеет цели "полный" либо имеет цель "пример", то создается вспомогательная задача Z', посылками которой становятся все посылки задачи Z, пополненные утверждениями $A_1, \ldots, A_m$. Условия ее получены заменой рассматриваемой кванторной импликации на группу конъюнктивных членов консеквента A_0. Цели задачи Z' воспроизводят цели задачи Z, с добавлением цели (независит ...), перечисляющей переменные $x_1, \ldots, x_n$ кванторной приставки. При необходимости перед созданием задачи Z' предпринимается переобозначение этих переменных. Требование независимости ответа R задачи Z' от переменных кванторной приставки, как легко видеть, гарантирует, что рассматриваемая кванторная импликация будет являться следствием R и посылок задачи Z. Поэтому R выдается в качестве ответа на задачу Z. Уровень срабатывания приема равен 6.

3. Попытка обратного вывода с помощью кванторной посылки. Как и выше, предполагаем, что текущая задача на описание не имеет цели "полный" либо имеет цель "пример". Пусть консеквент кванторной импликации имеет вхождения выражений вида "значение(f t)", где f - переменная, не входящая в кванторную приставку. Рассматривается список S всех таких переменных f. Находится такая импликативная посылка $\forall_{y_1 \ldots y_p}(B_1 \,\&\ldots\&\, B_r \to B_0)$, что консеквент B_0 содержит все переменные $y_1, \ldots, y_p$ и имеет вхождение такой переменной списка S, которая не встречается в антецедентах $B_1, \ldots B_r$. Если существуют термы $t_1, \ldots, t_p$, подстановка которых в B_0 вместо переменных $y_1, \ldots, y_p$ дает консеквент A_0, то вводится вспомогательная задача Z', полученная из текущей заменой в рассматриваемом импликативном условии консеквента на конъюнкцию результатов подстановки $t_1, \ldots, t_p$ вместо $y_1 \ldots, y_p$ в антецеденты $B_1, \ldots, B_r$. Если на задачу Z' найден ответ, то он выдается как ответ на текущую задачу. Уровень срабатывания приема равен 6.

4. Использование вспомогательной задачи на описание, имеющей цель "длялюбого". Ранее уже рассматривался прием, вводящий для исключения имплика-

тивного условия вспомогательную задачу Z' на описание с целью "независит". Посылками ее служили все утверждения из контекста импликативного условия (т.е. посылки и остальные условия), а также антецеденты; условиями являлись конъюнктивные члены консеквента. При получении ответа R, не зависящего от переменных кванторной приставки, можно было предпринять попытку замены импликативного условия на этот ответ. Часто бывает необходима более общая схема рассуждений, при которой по ходу решения задачи Z' разрешается вводить вспомогательные объекты, существование которых, вообще говоря, не вытекает из ее исходных посылок, но накладывает не слишком сильные дополнительные ограничения на параметры текущей задачи Z. Эти ограничения (разумеется, не зависящие от переменных кванторной приставки) присоединяются по окончании решения к ответу R. Чтобы указать на данный режим решения, задача Z' снабжается целью "длялюбого".

Прием, вводящий задачу Z', сначала находит список S всех свободных переменных f импликативного условия, имеющих вхождение выражения $f(t)$, некоторые переменные которого связаны внешними кванторами и описателями. Проверяется, что все переменные списка S появляются только в консеквенте импликативного условия. Это - эвристический фильтр, отсекающий ситуации, не типичные для задач, в которых прием был использован. Далее, как указано выше, создается задача Z'. Она имеет лишь три цели - "прямойответ", "длялюбого" и "независит $x_1 \ldots x_n$", указывающей на переменные кванторной приставки.

В процессе решения данной задачи создаются комментарии (блокпосылок Q_1 Q_2), регистрирующие ввод вспомогательных объектов. Здесь Q_1 - набор троек (X_1 X_2 P), соответствующих дополнительным посылкам "существует(X B)", вводящим вспомогательные объекты X. Связывающая приставка X разбивается на две части - X_1 и X_2. Переменные первой части считаются не зависящими от переменных кванторной приставки $x_1, \ldots, x_n$ рассматриваемого импликативного условия; переменные второй части - зависящими от них. P - набор конъюнктивных членов утверждения B. Квантор существования в дополнительной посылке устраняется специальным приемом, после чего все переменные списка X становятся обозначениями некоторых объектов, рассматриваемых в задаче Z'. Решение о том, какие переменные следует относить к X_1, а какие - к X_2, принимается на основе эвристических соображений. Логическая корректность последующих преобразований основана на учете этого решения. Q_2 есть набор конъюнктивных членов результата упрощения утверждения, выражающего, что для любых удовлетворяющих антецедентам $A_1, \ldots, A_m$ значений переменных $x_1, \ldots, x_n$ существуют такие значения переменных из списков X_2, представленных в Q_1, что истинна конъюнкция всех утверждений P. Обычно вводятся такие вспомогательные объекты, чтобы утверждения в Q_2 не имели вида кванторных импликаций. Задача Z' может иметь несколько различных комментариев (блокпосылок ...), относящихся к несвязанным между собой группам вспомогательных объектов.

По окончании решения задачи Z' составляется объединенный список D конъюнктивных членов ее ответа и всех утверждений, содержащихся в списках Q_2 комментариев "блокпосылок". На конъюнкцию элементов списка D навешивается квантор существования по всем переменным списков X_1. Пусть H - результирующее утверждение. Как нетрудно видеть, рассматриваемое импликативное условие является следствием этого утверждения относительно своего контек-

ста, и далее можно предпринять попытку решения вспомогательной задачи Z'', полученной из текущей задачи заменой импликативного условия на H.

В приеме предусмотрено особое рассмотрение случая, когда все неизвестные текущей задачи являются несущественными. Например, так бывает при сведении к задаче на описание задачи на доказательство существования. Здесь предпринимается попытка решения указанной выше задачи Z'' с ослабленными целями, разрешающими выдачу ответа M просто по исчерпании средств (как конъюнкцию текущих условий). В случае, когда M оказалось константой "истина", выдается ответ "истина". Иначе - рассматривается отрицание результата навешивания на утверждение M квантора существования по всем входящим в него неизвестным, которое регистрируется в списке посылок текущей задачи. Смысл такого преобразования состоит в отсечении ситуации, для которой задачу удалось решить, так что дальнейшие действия относятся к альтернативным ситуациям.

Уровень срабатывания приема равен 7.

В качестве примера применения данного приема рассмотрим задачу на доказательство равенства предела суммы двух сходящихся числовых последовательностей сумме их пределов. После несложных преобразований приходим к задаче на описание, имеющей следующие посылки:

последовательность(f, R), последовательность(g, R), $\forall_\epsilon(0 < \epsilon \ \& \ \epsilon -$ число $\rightarrow n(\epsilon) -$ натуральное$)$, $\forall_{\epsilon k}(0 < \epsilon \ \& \ \epsilon -$ число $\& \ k -$ натуральное $\& \ n(\epsilon) \leq k \rightarrow |a - f(k)| < \epsilon)$, $\forall_\epsilon(0 < \epsilon \ \& \ \epsilon -$ число $\rightarrow m(\epsilon) -$ натуральное$)$, $\forall_{\epsilon k}(0 < \epsilon \ \& \ \epsilon -$ число $\& \ k -$ натуральное $\& \ m(\epsilon) \leq k \rightarrow |b - g(k)| < \epsilon)$, $0 < c$, $c -$ число.

Условиями задачи служат кванторная импликация $\forall_N(N -$ натуральное $\& \ p \leq N \rightarrow |a + b - f(N) - g(N)| < c$ и утверждение "p - натуральное". Задача имеет единственную несущественную неизвестную p.

Переходя к вспомогательной задаче Z', добавляем к указанным выше посылкам утверждения "p-натуральное", "N-натуральное" и $p \leq N$. Условием этой задачи служит единственное утверждение $|a + b - f(N) - g(N)| < c$.

Так как ответ задачи не должен зависеть от переменной N, необходимо как-то исключить из условия выражения $f(N)$ и $g(N)$. Сделать это можно только с помощью информации, содержащейся в импликативных посылках для f и g. Однако, чтобы воспользоваться данными посылками, нужно обеспечить истинность их антецедентов. Например, в случае f нужно рассмотреть такое число ϵ_1, что истинны утверждения $0 < \epsilon_1$, $\epsilon_1 -$ число, $n(\epsilon_1) \leq N$. Вообще говоря, ниоткуда не следует существование этого числа. Однако, рассмотрение его необходимо, и соответствующий прием (мы рассмотрим его позднее) введет указанные утверждения в качестве дополнительных посылок, сопроводив их комментарием (блокпосылок Q_1 Q_2). Здесь Q_1 состоит из единственной тройки (ϵ_1, $\emptyset$, $\{0 < \epsilon_1, \epsilon_1 -$ число$, n(\epsilon_1) \leq N\}$). Для получения Q_2 предпринимается упрощение утверждения $\forall_N(N -$натуральное $\& \ p \leq N \rightarrow (0 < \epsilon_1 \ \& \ \epsilon_1 -$число $\& \ n(\epsilon_1) \leq N))$. В результате Q_2 оказывается состоящим из $0 < \epsilon_1$, $\epsilon_1 -$ число, $n(\epsilon_1) \leq p$. Аналогичным образом, для g получаем комментарий (блокпосылок ...), у которого Q_2 состоит из $0 < \epsilon_2$, $\epsilon_2 -$ число, $m(\epsilon_2) \leq p$. Используя дополнительные посылки и кванторные импликации в основных посылках, далее выводим следствия $|a - f(N)| < \epsilon_1$, $|b - g(N)| < \epsilon_2$. Это позволяет получить оценку $\epsilon_1 + \epsilon_2$ выражения $|a + b - f(N) - g(N)|$ из условия задачи. Так как данная оценка не зависит от

N, неравенство $\epsilon_1+\epsilon_2<c$ выдается в качестве ответа на Z'. Далее добавляем к нему все утверждения из накопителей Q_2 и навешиваем квантор существования по ϵ_1,ϵ_2. Получаем утверждение $\exists_{\epsilon_1\epsilon_2}(0<\epsilon_1$ & ϵ_1- число & $n(\epsilon_1)\le p$ & $0<\epsilon_2$ & ϵ_2- число & $m(\epsilon_2)\le p$ & $0<c-\epsilon_1-\epsilon_2)$. Оно и заменяет рассматриваемое импликативное условие. Квантор общности оказался устраненным, и далее задача легко доводится до ответа.

Накопление необходимых условий для получения достаточного условия

Если кванторная импликация является условием задачи на описание, в которой требуется получить полный ответ, то имеется еще одна возможность исключения квантора общности. Она состоит в накоплении такого списка бесквантор ных следствий данного условия, который в итоге оказывается эквивалентным самому условию. Прием, реализующий эти действия, прежде всего проверяет целесообразность попытки исключения квантора. Например, отсекаются ситуации, когда импликация входит в ответ как серия ограничений на допустимые значения неизвестных. После этого выполняется быстрая проверка реализуемости антецедентов в контексте рассматриваемого условия. Для вывода следствий из кванторной импликации используется вспомогательная задача на исследование Z'. Ее посылками служат все утверждения из контекста импликации, пополненные антецедентами и конъюнктивными членами консеквента. Задача имеет цель "длялюбого", а также цель, перечисляющую неизвестные текущей задачи на описание Z. Для использования в задаче Z отбираются лишь такие посылки задачи Z', которые не зависят от переменных кванторной приставки импликативного условия. Они регистрируются в комментарии (полныепосылки ...). В нем же регистрируются бесквантор ные результаты упрощения кванторных импликаций, полученных из рассматриваемого импликативного условия заменой консеквента на выведенные посылки задачи Z', зависящие от кванторной приставки. Это обеспечивается специальным приемом символа "исследовать" (см. выше). По завершении решения задачи Z' рассматривается содержимое R накопителя (полныепосылки ...). Если удается доказать, что импликативное условие является следствием своего контекста Q, пополненного утверждениями списка R, то эти утверждения упрощаются относительно Q, и предпринимается замена рассматриваемого условия задачи Z на их конъюнкцию. Уровень срабатывания приема равен 5.

В качестве примера применения данного приема рассмотрим задачу нахождения множества частичных пределов последовательности $1/2, 1/3, 2/3, 1/4, 2/4, 3/4, \ldots$. После простейших преобразований она сводится к задаче на описание, имеющей условие $\forall_{an}(0<a$ & $a-$ число & $n-$ натуральное $\to \exists_{bm}(n<m$ & $|x-\frac{b}{m+1}|<a$ & $b\in\{1,\ldots,m\}$ & $m-$ натуральное)). Единственной неизвестной является переменная x. Вспомогательная задача на исследование, создаваемая приемом, имеет посылки $x-$ число, $0<a$, $a-$ число, $n-$ натуральное, $\exists_{bm}(n<m$ & $|x-\frac{b}{m+1}|<a$ & $b\in\{1,\ldots,m\}$ & $m-$ натуральное). Квантор существования в последней посылке устраняется специальным приемом, заменяющим переменные b,m на новые переменные c,d. При этом возникают посылки $n<d$, $|x-\frac{c}{d+1}|<a$, $c\in\{1,\ldots,d\}$, $d-$ натуральное и добавляется цель (независит c d). Из неравенства с модулем выводятся следствия $-a+\frac{c}{d+1}<x$, $x<a+\frac{c}{d+1}$. Чтобы получить неравенства для x, не содержащие переменных c,d, применяются приемы, обращающиеся к синтезаторам "верхняяоценка" и "нижняяоценка". Они выводят следствия $-a<x$, $x<a+1$. Сразу по получении каждого из них применяется общий прием символа "исследовать". В первом случае

он упрощает утверждение $\forall_a(0 < a \ \& \ a - \text{число} \rightarrow -a < x)$ и получает $0 \leq x$, во втором - упрощает утверждение $\forall_a(0 < a \ \& \ a - \text{число} \rightarrow x < a+1$ и получает $x \leq 1$. Оба результата регистрируются в комментарии (полныепосылки ...). По окончании решения вспомогательной задачи на исследование предпринимается попытка доказать, что из неравенств $0 \leq x, x \leq 1$ вытекает истинность рассматриваемого импликативного условия. Эта попытка оказывается успешной, и прием заменяет импликативное условие на конъюнкцию неравенств.

Подбор контрпримера к импликативной посылке задачи на исследование, имеющей цель "противоречие"

Задача на исследование, имеющая цель "противоречие", обычно решается для доказательства какого-либо утверждения от противного. При ее решении нужно усмотреть противоречивость посылок, т.е. вывести следствие "ложь". Если кванторная импликация $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_m \rightarrow A_0)$ встречается в посылках такой задачи, то можно попытаться доказать ее отрицание, подобрав контрпример. С этой целью создается вспомогательная задача на описание Z', условиями которой становятся все антецеденты $A_1, \ldots, A_m$ и отрицание консеквента A_0. Посылками служат все остальные посылки текущей задачи на исследование. Цели вспомогательной задачи суть (неизвестные $x_1 \ldots x_n$), (параметры $x_1 \ldots x_n$), "полный", "пример", "отрицание". Последняя цель просто служит пометкой, указывающей на источник появления данной задачи. Максимальный уровень обращения к задаче Z' невелик и устанавливается из эвристических соображений. Если задача Z' решена, то импликативная посылка текущей задачи на исследование заменяется на константу "ложь". Уровень срабатывания приема равен 5.

Попытка вывода следствий из консеквента кванторной импликации для последующей кванторной свертки

Если некоторая задача имеет посылку P вида $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_m \rightarrow A_0)$, то для получения бескванторных следствий может быть использована следующая процедура. Создается вспомогательная задача на исследование Z', посылками которой становятся все отличные от P посылки текущей задачи, антецеденты $A_1, \ldots, A_m$ и консеквент A_0. В задаче Z' предпринимается вывод следствий, причем лимит времени, отпущенного на ее решение, не очень велик. По окончании вывода следствий рассматриваются те посылки Q задачи Z', отличные от ее исходных посылок, которые получены с существенным использованием консеквента A_0. Последнее определяется по комментарию (выводимо ...) к посылке Q. Далее решается задача на упрощение кванторной импликации $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_m \rightarrow Q)$, результатом которой служит некоторое утверждение R. Если R бескванторное, то оно регистрируется в посылках текущей задачи как следствие посылки P. В комментарии (выводимо ...) к R учитываются также другие посылки, использованные при выводе. Уровень срабатывания приема равен 5.

Вывод следствий из кванторной посылки типа "для любого существует" с помощью вспомогательной задачи на описание

Приведем еще один прием вывода следствий из кванторных импликаций, на этот раз имеющих уже специальный вид - $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_m \rightarrow \exists_{y_1 \ldots y_k}(B_1 \ \& \ \ldots \ \& \ B_p))$.

Прием применяет схему рассуждений "от противного". Сначала рассматривается отрицание N кванторной импликации. Оно имеет вид

$$\exists_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_m \ \& \ \forall_{y_1 \ldots y_k}(B_1 \ \& \ \ldots \ \& \ B_{p-1} \to \neg B_p)).$$

Чтобы получить бескванторное утверждение P, следствием которого являлась бы кванторная импликация $\forall_{y_1 \ldots y_k}(B_1 \ \& \ \ldots \ \& \ B_{p-1} \to \neg B_p)$, решается вспомогательная задача на описание, имеющая цель "длялюбого". Как и в рассмотренном ранее приеме, она использует комментарий (блокпосылок ...), накапливающий дополнения к ответу. Далее рассматриваются утверждения $A_1, \ldots, A_m, P$, которые становятся условиями еще одной вспомогательной задачи на описание. В ней требуется подобрать пример для неизвестных $x_1, \ldots, x_n$, причем эти неизвестные объявляются несущественными. Как легко видеть, N является следствием ответа R на указанную задачу. Эта задача имеет цель "попыткаспуска", т.е. по исчерпании средств в качестве ответа выдается конъюнкция ее текущих условий. На оставшиеся неисключенными неизвестные в R навешивается квантор существования. Завершает рассуждения "от противного" присоединение отрицания утверждения R в качестве новой посылки. Уровень срабатывания приема равен 8.

Данный прием возник при рассмотрении задачи на доказательство существования наибольшего либо наименьшего элемента у сходящейся числовой последовательности. Эту задачу можно найти в разделе задачника "Математический анализ" - "Задачи на доказательство" - "Последовательности" и проследить по ней ход применения приема. В некоторый момент решения возникает задача на описание, имеющая посылки $\forall_n(n - $ натуральное $\to f(n) \le d)$, $\forall_c(0 < c \ \& \ c -$ число $\to p(c) -$ натуральное), $\forall_m(m -$ натуральное $\to \exists_k(f(k) < f(m) \ \& \ k -$ натуральное)), $d -$ число, последовательность(f, R), $\forall_{cl}(0 < c \ \& \ c -$ число $\& \ l -$ натуральное $\& \ p(c) \le l \to |d - f(l)| < c)$. Прием применяется к посылке $\forall_m(m -$ натуральное $\to \exists_k(f(k) < f(m) \ \& \ k -$натуральное)). Отрицание ее консеквента имеет вид $\forall_k(k -$натуральное $\to f(m) \le f(k))$. Создается задача на описание с целью "длялюбого", посылки которой суть: $k -$ натуральное, $m -$ натуральное, $\forall_n(n -$ натуральное $\to f(n) \le d)$, $\forall_c(0 < c \ \& \ c -$ число $\to p(c) -$ натуральное), последовательность(f, R), $d -$ число, $\forall_c l(0 < c \ \& \ c -$ число $\& \ l -$ натуральное $\& \ p(c) \le l \to |d - f(l)| < c)$. Условием ее служит неравенство $f(m) \le f(k)$. На эту задачу решатель получает ответ $f(m) \le d - a \ \& \ f(m) \le \inf(\mathrm{set}_x(\exists_i(i \in \{1, \ldots, p(a) - 1\} \ \& \ x = f(i))))$. Здесь a - вспомогательный параметр, условия на который извлекаются из комментария (блокпосылок ...). После учета условий на a ответ преобразуется к виду $\exists_a(a -$ число $\& \ 0 < a \ \& \ f(m) \le d - a \ \& \ f(m) \le \inf(\mathrm{set}_x(\exists_i(i \in \{1, \ldots, p(a) - 1\} \ \& \ x = f(i)))))$. Далее решается задача на подбор значения m, удовлетворяющего предыдущему утверждению и условию "$m -$ натуральное". По исчерпании средств, на нее выдается ответ $\exists_i(f(i) < d \ \& \ i -$ натуральное). Отрицание последнего утверждения, имеющее вид $\forall_i(i -$ натуральное $\to d \le f(i))$, заносится в качестве новой посылки задачи. Так как в посылках имеется встречное кванторное неравенство для f, обнаруживается, что последовательность константная, и далее задача быстро доводится до конца.

Использование индуктивного предположения, имеющего вид кванторной импликации

Приемы доказательства по индукции создают вспомогательные задачи, списки посылок которых пополняются утверждениями - конъюнктивными членами индуктив-

ного предположения. Вывод следствий из этих утверждений является приоритетным, и для усиления его создан ряд дополнительных приемов. Один из них относится к случаю, когда индуктивное предположение имеет вид кванторной импликации $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_m \rightarrow A_0)$. Проверяется, что переменная либо выражение, по которому ведется индукция, встречается в антецедентах. Находится элементарный антецедент A_i, содержащий все переменные кванторной приставки, и для него выбирается посылка P, представляющая собой результат применения к A_i некоторой подстановки s вместо переменных кванторной приставки. Проверяется, что применение s ко всем оставшимся антецедентам дает утверждения, истинность которых относительно контекста быстро устанавливается с помощью процедуры "извлекается". Тогда находится результат применения s к консеквенту A_0, и этот результат регистрируется в посылках текущей задачи. Уровень срабатывания приема равен 3.

Использование кванторной импликации с функциональными переменными для вывода следствий в задаче на исследование, имеющей цель "противоречие"

Если кванторная импликация $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_m \rightarrow A_0)$ имеет для каждой переменной x_i кванторной приставки вхождение в консеквент выражения вида $f(x_i)$, то можно попробовать использовать ее для вывода утверждения, содержащего уже встречавшиеся в задаче выражения вида $g(t)$. Такое утверждение, дающее дополнительную связь между уже имеющимися объектами, может оказаться полезным для усмотрения противоречия. Прием определяет для каждой переменной x_i список T_i таких выражений t, что консеквент A_0 содержит $f(x_i)$, а некоторая элементарная посылка - $f(t)$. Далее перебираются всевозможные наборы $(t_1, \ldots t_n)$ выражений t_i из списков T_i. Предварительно проверяется. что число этих наборов невелико (для рассматривавшихся примеров оказалось достаточно верхней границы числа наборов, равной 3). Как только обнаруживается набор, подстановка которого вместо переменных кванторной приставки в антецеденты дает утверждения, усматриваемые с помощью проверочных операторов, выполняется вывод результата применения той же подстановки к консеквенту A_0. Уровень срабатывания приема равен 3.

10.9 Приемы символа "существует"

Хотя квантор существования с помощью отрицания выражается через квантор общности, лишь простейшие его приемы аналогичны рассмотренным выше. Заметим также, что корневые вхождения квантора существования обычно устраняются с помощью несложных логических переходов - в посылках за счет ввода вспомогательных объектов, в условиях - за счет перехода к задаче на описание либо за счет ввода дополнительных неизвестных такой задачи.

Простейшая стандартизация

Перечислим приемы общей стндартизации кванторов существования. В основном, они аналогичны таким приемам, введенным для квантора общности. Уровень срабатывания их равен 0.

1. Переобозначение связанных переменных. Переменные кванторной приставки переобозначаются так, чтобы они отличались от свободных переменных утверждений того контекста, относительно которого рассматривается квантор существования. Для этого используется процедура "нормализациясвязок".

2. Исключение из кванторной приставки переменных, не имеющих свободных вхождений в подкванторное утверждение.

3. Устранение из кванторной приставки повторных вхождений одной и той же переменной.

4. Слияние кванторных приставок вложенных кванторов существования. Выполняется замена утверждения $\exists_{x_1 \ldots x_n}(\exists_{y_1 \ldots y_m} A)$ на $\exists_{x_1 \ldots x_n y_1 \ldots y_m} A$.

5. Устранение дизъюнкции под квантором существования. Выполняется замена утверждения $\exists_{x_1 \ldots x_n}(A_1 \vee \ldots \vee A_m)$ на $\exists_{x_1 \ldots x_n} A_1 \vee \ldots \vee \exists_{x_1 \ldots x_n} A_m$.

6. Подстановка явно определенного равенством значения связанной переменной. Утверждение $\exists_{x_1 \ldots x_n}(x_i = t \ \& \ A)$, где терм t не зависит от x_i, заменяется на $\exists_{x_1 \ldots x_{i-1} x_{i+1} \ldots x_n} B$. Здесь B - результат подстановки выражения t вместо x_i в утверждение A. На применение этого преобразования имеется ряд ограничений, связанных с теми случаями, когда равенство под квантором существования нужно для задания некоторого параметрического описания.

Разбиение множества конъюнктивных членов подкванторного утверждения на компоненты связности по зависимости от переменных кванторной приставки

Конъюнктивные члены $A_1, \ldots, A_n$ подкванторного утверждения в кванторе существования $\exists_{x_1 \ldots x_m}(A_1 \ \& \ \ldots \ \& \ A_n)$ разбиваются на наибольшее число таких непересекающихся подгрупп $B_1, \ldots, B_k$, что утверждения из двух различных подгрупп не зависят от одной и той же переменной кванторной приставки. Если $k > 1$, то рассматриваются конъюнкции $C_1, \ldots, C_k$ утверждений групп $B_1, \ldots, B_k$, и квантор существования заменяется на конъюнкцию кванторов $\exists_{X_1} C_1$, ... , $\exists_{X_k} C_k$. Здесь X_i - все переменные $x_1, \ldots, x_n$, имеющие свободные вхождения в утверждение C_i. Если какая - либо группа переменных X_i пуста, то C_i берется без навешивания на него квантора существования. В зависимости от контекста, прием может срабатывать на уровнях 0, 1 и 3.

Устранение дизъюнкции, являющейся конъюнктивным членом утверждения под квантором существования

Утверждение $\exists_{x_1 \ldots x_n}(A \ \& \ (B1 \vee \ldots \vee B_m))$ заменяется на $\exists_{x_1 \ldots x_n}(A \ \& \ B_1) \vee \ldots \vee \exists_{x_1 \ldots x_n}(A \ \& \ B_m)$. В зависимости от контекста, уровень срабатывания приема равен 1 либо (в редких случаях)4. При редактировании ответа задачи на описание прием обычно блокируется.

Группировка кванторов наружу

Если конъюнктивный член подкванторного утверждения сам является квантором существования, то этот квантор выносится наружу: утверждение $\exists_{x_1 \ldots x_n}(A \ \& \ \exists_{y_1 \ldots y_m} B)$

заменяется на $\exists_{x_1 \ldots x_n y_1 \ldots y_m}(A \,\&\, B)$. Уровень срабатывания приема равен 1. Заметим, что к моменту применения данного приема уже выполнено переобозначение связанных переменных $y_1, \ldots, y_m$, гарантирующее, что они не имеют свободных вхождений в утверждение A.

Решение задачи на доказательство существования путем подбора примера

Если условием задачи на доказательство служит утверждение $\exists_{x_1 \ldots x_n}(A_1 \,\&\, \ldots \,\&\, A_m)$, $m \geq 1$, то создается вспомогательная задача на описание Z', имеющая тот же список посылок и условия $A_1, \ldots, A_m$. Эта задача имеет цели "полный", "пример", (неизвестные $x_1 \ldots x_n$), (параметры $x_1 \ldots x_n$). Комментарии к ее посылкам переносятся из текущей задачи с помощью справочника "существует". Если на задачу Z' получен ответ, отличный от символов "отказ" и "ложь", то на текущую задачу выдается ответ "истина". В противном случае решение текущей задачи на доказательство продолжается с привлечением других средств. Уровень срабатывания приема равен 2.

Элиминация квантора существования в посылках за счет ввода вспомогательных объектов

Если в посылках задачи на доказательство либо задачи на описание, не имеющей цели "прямойответ", встречается утверждение вида $\exists_{x_1 \ldots x_n} A$, то квантор существования устраняется. Здесь рассматриваются два случая:

1. Текущая задача - на доказательство либо не имеет цели (независит $z_1 \ldots z_m$), переменные которой имеют свободные вхождения в рассматриваемую кванторную посылку. Тогда выбираются не использованные в задаче переменные $y_1, \ldots, y_n$, находится результат B переобозначения в A переменных x_i на y_i, и кванторная посылка заменяется на B.

2. Текущая задача - на описание и имеет цель (независит $z_1 \ldots z_m$), переменные которой имеют свободные вхождения в рассматриваемую посылку. Находится список $Z_1, \ldots, Z_p$ всех таких переменных. Выбираются не использованные в задаче переменные $y_1, \ldots, y_n$ и определяется результат B подстановки в A вместо переменных $x_1, \ldots, x_n$ выражений $y_1(Z_1 \ldots\ Z_p), \ldots, y_n(Z_1 \ldots\ Z_p)$. Затем кванторная посылка заменяется на B.

В обоих случаях корректируется комментарий (новаяпеременная S_1 S_2) к посылкам текущей задачи. Такой комментарий перечисляет в наборе S_1 все вспомогательные переменные посылок, возникшие в процессе вывода следствий. Список S_2 имеет ту же длину, что и список S_1. На позиции его, соответствующей переменной x списка S_1, указывается набор ранее введенных вспомогательных переменных, от которых x может зависеть. В данном приеме происходит добавление к списку S_1 переменных $y_1, \ldots, y_n$ и указание для них в S_2 пересечения S_1 со списком параметров кванторной посылки. Уровень срабатывания приема равен 2.

Элиминация квантора существования в посылках задачи на исследование, имеющей цель "длялюбого"

Этот прием аналогичен предыдущему, но относится к задачам на исследование. Требуется также, чтобы задача имела цель "длялюбого" (см. приведенный ранее прием, анализирующий кванторную импликацию в условиях задачи на описание). Для

рассматриваемой посылки $\exists_{x_1 \ldots x_n} A$ выбираются новые переменные $y_1, \ldots, y_n$. Если задача имела цель (независит ...), то к списку переменных этой цели присоединяются переменные $y_1, \ldots, y_n$. В противном случае создается цель (независит $y_1 \ldots y_n$). Далее посылка заменяется на результат переобозначения в A переменных x_i на y_i. Уровень срабатывания приема равен 2.

Ввод сколемовских функций для элиминации квантора существования в консеквенте кванторной посылки

Квантор существования исключается не только в ситуациях, когда он является заголовком посылки, но и в ситуациях, когда он является заголовком консеквента посылки - кванторной импликации. Именно, если посылка задачи на доказательство либо задачи на описание, не имеющей цели "прямойответ", представима в виде $\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\, \&\, A_m \rightarrow \exists_{y_1 \ldots y_m} A_0)$, то выбираются новые переменные $f_1, \ldots, f_m$, и посылка заменяется на $\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\, \&\, A_m \rightarrow B$. Здесь B - результат подстановки в утверждение A_0 выражений $f_1(x_1 \ldots x_n), \ldots, f_n(x_1 \ldots x_n)$ вместо переменных $y_1, \ldots, y_m$. Прием корректирует комментарий (новаяпеременная S_1 S_2) так же, как это делал предыдущий прием. Переменные $f_1, \ldots, f_m$, возникающие при данном преобразовании, в математической логике обычно называются "сколемовскими функциями". Прием пока реализован лишь для случая отсутствия у задачи на описание цели (независит ...), пересекающейся с параметрами посылки. Уровень его срабатывания равен 4.

Элиминация условия существования в задаче на описание, решаемой с целью "пример"

Если задача на описание имеет условие $\exists_{x_1 \ldots x_n} A$, то в ряде случаев можно заменить данное условие на A, присоединив переменные $x_1, \ldots, x_n$ к списку неизвестных задачи и указав на то, что они являются несущественными. Это делается, если задача имеет цель "пример", либо цель "замещение" (переформулировка условий без использования неизвестных, которые в таких задачах могут встречаться и в посылках), либо цель "функционально" (в задаче нужно лишь установить факт однозначного определения значений неизвестных по значениям известных параметров). Кроме того, срабатывание приема допускается при отсутствии неизвестных либо при наличии у условия комментария "параметры". Прием блокируется на этапе редактирования ответа. Если задача имела цель (независит $y_1 \ldots y_k$), то для каждой переменной x_i вводится дополнительная цель (сокращнеизв x_i y_1 ... y_k), разрешающая неизвестной x_i зависеть от переменных $y_1, \ldots, y_k$. Прием не преобразует указанным выше образом текущую задачу, а вводит и решает вспомогательную задачу. Если на нее получен "отказ", то и на текущую задачу выдается "отказ". В противном случае - в зависимости от целевой установки либо сразу выдается найденный ответ, либо на него навешивается квантор существования по переменным $x_1, \ldots, x_n$, и ответ выдается лишь после упрощения этого квантора. Для упрощения используется еще одна вспомогательная задача на описание, снабженная целью "редакция". Уровень срабатывания приема равен 3.

Удаление посылки "существует(...)" за счет непосредственного подбора значений

Если посылка задачи имеет вид $\exists_{x_1 \ldots x_n}(A_1 \,\&\, \ldots \,\&\, A_m)$, где $A_1, \ldots, A_m$ - элементарные утверждения, то предпринимается попытка усмотреть среди прочих посылок такие элементарные утверждения $B_1, \ldots, B_m$, которые являются результатами подстановки вместо переменных $x_1, \ldots, x_n$ в утверждения $A_1, \ldots, A_m$ некоторых выражений $t_1, \ldots, t_n$. Для этого используется процедура "подборпосылок". Если она выдает искомый результат, то рассматриваемая кванторная посылка удаляется. Уровень срабатывания приема равен 1.

Разрешение подкванторного условия относительно переменных кванторной приставки

Аналогичный прием уже рассматривался для кванторной импликации. Проверяется отсутствие целей и комментариев, блокирующих попытку разрешения подкванторного утверждения условия $\exists_{x_1 \ldots x_n} A$ относительно неизвестных $x_1, \ldots, x_n$. Создается вспомогательная задача Z', посылками которой служат все утверждения из контекста рассматриваемого условия, а условиями - конъюнктивные члены утверждения A. Цели задачи суть "полный", "явное", "прямойответ", (неизвестные $x_1 \ldots x_n$), (параметры $x_1 \ldots x_n$), а также ряд целей, переносимых из текущей задачи. Если при решении задачи Z' получен ответ B, отличный от A, то происходит замена условия на $\exists_{x_1 \ldots x_n} B$. Смысл преобразования состоит в том, что если подкванторное утверждение разрешено относительно переменных кванторной приставки, то обычно удается вообще исключить квантор, используя простейшие эквивалентные преобразования. Уровень срабатывания приема равен 3.

Перенесение неизвестной посылки "существует(...)" из блока анализа во внешнюю задачу на описание

Аналогичный прием переносил из блока анализа во внешнюю задачу дизъюнкцию, содержащую неизвестные. Собственно перенесение квантора существования выполняется процедурой "замещениеусловий", которая пытается дополнить это условие необходимыми сопровождающими утверждениями, извлекаемыми из посылок задачи на исследование, а при возможности - также отбросить условия внешней задачи, оказывающиеся после перенесения избыточными. Уровень срабатывания приема равен 2. Так как появление в посылках блока анализа дизъюнкции либо утверждения существования, содержащих неизвестные, приводит к почти немедленному перенесению их во внешнюю задачу, сопровождаемому для дизъюнкции разбором случаев, а для квантора - попытками его исключения, то нужно вводить новые посылки одного из указанных видов лишь при наличии достаточно веских причин.

Устранение сдвоенных переменных кванторной приставки

Реализована несколько ослабленная версия приема, имевшегося для кванторных импликаций. Рассматривается утверждение вида $\exists_{x_1 \ldots x_n}(A_1 \,\&\, \ldots \,\&\, A_m)$. Если для двух различных переменных кванторной приставки x_i, x_j некоторые A_k, A_p имеют

вид $P(x_i)$, $P(x_j)$, где P - название типа объектов, причем во всех остальных конъюнктивных членах подкванторного утверждения эти переменные встречаются только под знаком одной и той же операции f (в некоммутативном случае - в одном и том же порядке), то предпринимается склейка данных переменных. Предварительно проверяется, что f имеет единицу - это гарантирует, что на значения новой переменной не возникнут дополнительные ограничения. Фактически в качестве новой переменной берется сама переменная x_i: выражения $f(x_i, x_j)$ заменяются на x_i, а $P(x_j)$ отбрасывается.

Параметрические описания, определяющие значения неизвестных задачи

В ответах задач на описание могут появляться утверждения вида $\exists_{x_1 \ldots x_n}(P \ \& \ y_1 = t_1 \ \& \ \ldots y_n = t_n)$, где $y_1, \ldots, y_n$ - неизвестные; P - утверждение без неизвестных, ограничивающее множество допустимых значений переменных $x_1, \ldots, x_n$; $t_1, \ldots, t_n$ - выражения без неизвестных. Такие утверждения называются явными параметрическими описаниями множества допустимых значений неизвестных $y_1, \ldots, y_n$; переменные $x_1, \ldots, x_n$ суть параметры описания. Иногда рассматриваются также неявные параметрические описания, у которых значения неизвестных $y_1, \ldots, y_n$ не задаются явными выражениями через параметры, а связываются с ними каким-либо косвенным образом. Например, могут указываться промежутки для значений неизвестных. Чтобы усматривать и упрощать условия задачи, представляющие собой параметрические описания, а также чтобы непосредственно усматривать готовый ответ, составленный из таких описаний, служат специальные приемы. Перечислим те из них, которые реализованы на ЛОСе:

1. Усмотрение параметрического описания серии значений неизвестных.

   Если условие задачи представляет собой параметрическое описание значений неизвестных либо введено для последующего преобразования к такому описанию, то оно помечается комментарием "серия". Часто комментарии "серия" вводятся приемами, создающими кванторное условие. При отсутствии данного комментария у условия вида $\exists_{x_1 \ldots x_n} A$ предпринимается попытка усмотреть в нем параметрическое описание. Прежде всего, проверяется, что задача на описание не имеет целей "редакция" и "пример". Затем проверяется, что утверждение A имеет вид $y_1 = t_1 \ \& \ \ldots \ \& \ y_k = t_k \ \& \ B$, где $y_1, \ldots, y_k$ - неизвестные; $t_1, \ldots, t_k, B$ не содержат неизвестных. Проверяется, что каждое вхождение в задачу неизвестной y_i, $i \in \{1, \ldots, k\}$, не относящееся к рассматриваемому кванторному условию, относится к простейшему указателю типа данных $P(y_i)$. Проверяется отсутствие у B конъюнктивных членов с заголовками "равно", "число". После этого кванторное условие сопровождается комментарием "серия". Уровень срабатывания приема - 2 либо 3.

2. Переобозначение связанных переменных параметрического описания.

   Переменные кванторной приставки условия $\exists_{x_1 \ldots x_n} A$ переобозначаются так, чтобы они не имели свободных вхождений в утверждения контекста данного условия. Это делает на уровне 0 прием, о котором уже говорилось ранее. Иногда бывает необходимо дополнительное переобозначение. Если задача на описание Z имеет комментарий (контекст Z' B), то при редактировании ее ответа все

утверждения списка B будут присоединяться к данному ответу. Они определяют выражение исключенных ранее неизвестных задачи Z' через оставшиеся ее неизвестные. Исключенные неизвестные в условиях и посылках текущей задачи Z уже не встречаются. К моменту редактирования ответа кванторная приставка рассматриваемого условия может быть отброшена, а переменные ее добавлены к списку неизвестных задачи. Если эти переменные имели свободные вхождения в утверждения списка B, т.е. совпадали с исключенными неизвестными, то произойдет коллизия - отождествление переменных, обозначающих различные объекты. Для ее предотвращения необходимо заблаговременно переобозначить переменные кванторной приставки так, чтобы они не имели свободных вхождений в B. Это переобозначение выполняется приемом, срабатывающим на уровне 2.

3. Выдача ответа, образованного параметрическим описанием.

   Если условие задачи на описание имеет вид $\exists_{x_1 \ldots x_n} A$ и снабжено комментарием "серия", то рассматриваются все переменные $y_1, \ldots, y_m$ этого условия, являющиеся неизвестными либо встречающиеся в цели (серия ...). Выделяются все конъюнктивные члены $A_1, \ldots, A_k$ утверждения A, содержащие переменные y_i. Далее проверяются условия:

   (a) Каждое A_j имеет либо вид $y_i = t$, где t известно, либо вид $P(y_i(x) \ldots)$;

   (b) Рассматриваемое кванторное условие не сопровождается комментарием "фильтрсерии";

   (c) Задача не имеет неизвестных, не встречающихся в данном кванторном условии;

   (d) Каждое вхождение неизвестной y вне данного кванторного условия относится к простейшему указателю типа значения этой неизвестной - утверждению $Q(y)$.

   (e) Отсутствует комментарий (контекст ...) к задаче.

   Если все они выполнены, то принимается решение о том, что список условий задачи уже представляет собой почти готовый ответ. К нему применяется стандартная процедура завершающего редактирования (т.е. процедура "редакторответа"), и далее выдается результат ее применения. Уровень срабатывания приема равен 2 либо 3.

   Требование отсутствия комментария (контекст ...) объясняется тем, что содержащиеся в нем утверждения, выражающие ранее исключенные неизвестные через неизвестные, определяемые параметрическим описанием, требуют явной подстановки значений согласно этому параметрическому описанию. После подстановки значений могут понадобиться действия по упрощению ответа. В такой ситуации немедленная выдача ответа является преждевременной.

4. Выдача ответа, представляющего собой параметрическое описание числового промежутка.

   В предположениях предыдущего приема, вместо перечисляемых в нем условий, проверяется, что утверждения $A_1, \ldots, A_k$ образуют одно либо два неравенства,

определяющих промежуток значений некоторой неизвестной y. Если задача не имеет других неизвестных, а неизвестная y встречается вне рассматриваемого кванторного условия лишь внутри указателя типа ее значения, причем отсутствует комментарий (контекст ...), то предпринимается обращение к процедуре "редакторответа" и выдача ответа. Уровень срабатывания - 2 либо 3.

5. Обращение к вспомогательной задаче на редактирование параметрического описания.

   В предположениях двух предыдущих приемов, оказывается, что приводимые в них условия на утверждения $A_1, \ldots, A_k$ не выполнены. Тогда предпринимается обращение к вспомогательной задаче Z', предназначенной для редактирования рассматриваемого параметрического описания. Предварительно уточняется уровень срабатывания приема: если все утверждения A_i суть неравенства, то он равен 3, иначе - равен 2. Условиями задачи Z' служат все условия текущей задачи Z, отличные от параметрического описания, а также все конъюнктивные члены подкванторного утверждения A. Посылки ее те же, что у текущей задачи. Списки неизвестных и несущественных неизвестных пополняются в задаче Z' переменными $x_1, \ldots, x_n$ кванторной приставки. Если задача Z не имеет цели (серия ...), то вводится цель (серия $x_1 \ldots x_n$), в противном случае старая цель (серия ...) пополняется переменными кванторной приставки. Таким образом, цель (серия ...) перечисляет все переменные задачи, являющиеся параметрами внешних редактируемых параметрических описаний. В задачу Z' переносятся прочие цели задачи Z, а также добавляется цель "учетответа". Последняя служит сигнализатором режима редактирования параметрического описания.

   Если на задачу Z' получен ответ R, отличный от символа "отказ", то на этот ответ навешивается квантор существования по всем входящим в R переменным списка $x_1, \ldots x_n$. Полученное таким образом утверждение упрощается с помощью пакетного нормализатора "нормсуществует", и результат выдается в качестве ответа. В особых случаях предпринимаются дополнительные шаги по исключению из ответа имеющихся в нем кванторов существования.

   Данный прием часто используется при решении тригонометрических уравнений. После получения серии корней $\exists_n(n - \text{целое } \& \ x = f(n))$ обычно бывает нужно учесть дополнительные ограничения на x, например, вытекающие из условий на область допустимых значений. При решении указанной выше вспомогательной задачи Z', в которой внешний квантор существования отброшен, появляется возможность подставить выражение для x в эти дополнительные ограничения. Их анализ приводит к отбрасыванию некоторых подсерий найденной серии, и таким образом получается окончательный ответ.

Усмотрение при редактировании ответа параметрического описания для известной переменной, встречающейся в условиях с неизвестной, и подстановка определяемого им значения этой переменной

Пусть при редактировании ответа задачи на описание в списке условий обнаруживается утверждение вида $\exists_{x_1 \ldots x_n}(y_1 = t_1 \ \& \ \ldots y_m = t_m \ \& \ A)$, не содержащее неизвестных. Пусть также $y_1, \ldots, y_m$ - различные переменные, не встречающиеся в выражениях $t_1, \ldots, t_m$ и отличные от переменных кванторной приставки. Такая ситуация, в которой появляется параметрическое описание для известного параметров

задачи, может возникнуть после решения уравнения относительно этого параметра, а само уравнение - возникнуть при разборе случаев, как указывающее на некоторый вырожденный подслучай. Если известный параметр появляется в условиях, содержащих неизвестные, то делается попытка получить параметрическое описание для неизвестных. Для этого рассматриваются все условия $B_1, \ldots, B_k$, содержащие переменные $y_1, \ldots, y_m$ и отличные от исходного кванторного условия. Находятся результаты $C_1, \ldots, C_k$ подстановки в них выражений $t_1, \ldots, t_m$ вместо переменных $y_1, \ldots, y_m$, к которым применяется процедура быстрого упрощения. Затем вводится вспомогательная задача на описание Z', условиями которой являются утверждение $\exists_{x_1 \ldots x_n}(C_1 \ \& \ \ldots \ \& \ C_k \ \& \ y_1 = t_1 \ \& \ \ldots \ \& \ y_m = t_m \ \& \ A)$ и все остальные (т.е. не содержащие $y_1, \ldots, y_m$) условия текущей задачи. Первое из перечисленных условий снабжается комментарием "серия"; вместо цели "редакция" задаче Z' дается цель "упростить". Это означает, что от редактирования ответа предпринимается откат к повторному рассмаотрению списка условий. Далее задача Z' решается, и найденный на нее ответ (либо отказ) выдается как ответ на текущую задачу.

Переобозначение переменных параметрического описания, определяемое вспомогательным условием

В некоторых случаях при редактировании параметрического описания может оказаться, что целесообразен переход к другим параметрам. Примеры такого рода возникали в связи с редактированием ответов для дифференциальных уравнений. Переход к новым параметрам задается условием вида $\exists_{x_1 \ldots x_n}(y_1 = t_1 \ \& \ \ldots \ \& \ y_n = t_n \ \& \ A)$, которое заблаговременно вводится дргуим приемом. Это условие снабжается комментарием "заменапеременной". Старые параметры суть $y_1, \ldots y_n$; новые - $x_1, \ldots, x_n$. Предполагается, что количества этих параметров одинаковы. Утверждение A и выражения $t_1, \ldots, t_n$ не содержат старых параметров. Прием, реализующий переход к новым параметрам x_i, использует для их обозначения старые переменные y_i. Он активизируется при сканировании задачи на описание, имеющей цель "учетответа" (см. выше). Проверив наличие комментария "заменапеременной", прием определяет выражения $s_1, \ldots, s_n$, получаемые из $t_1, \ldots, t_n$ подстановкой переменных $y_1, \ldots, y_n$ вместо $x_1, \ldots, x_n$. Затем во всех прочих условиях предпринимается подстановка выражений $s_1, \ldots, s_n$ вместо $x_1, \ldots, x_n$. Такая же подстановка предпринимается в термах комментариев (сопровождение ...). В заключение переобозначения определяющее замену переменных кванторное условие отбрасывается, а утверждение A добавляется к списку условий. Прием срабатывает на уровне 0.

Усмотрение рода объекта, определяемого внутри параметрического описания

Если при редактировании ответа задачи на описание встречается параметрическое описание $\exists_{x_1 \ldots x_n}(y = t \ \& \ A)$, где y - неизвестная задачи; t - выражение без неизвестных, причем заголовок выражения t определяет тип P значения этого выражения, то исключается (если оно есть) условие $P(y)$. Прием срабатывает на уровне 2.

Элиминация условного выражения под квантором существования

Если внутри кванторного утверждения $\exists_{x_1 \ldots x_n} A$ встречается условное выражение "вариант$(x = p \ t_1 \ t_2)$", причем x - переменная кванторной приставки, а выражение p

не имеет вхождений переменных, связанных внутри A, то находятся результаты A_1, A_2 замены в утверждении A всех вхождений данного условного выражения на t_1 и t_2, соответственно. Затем предпринимается упрощение утверждений $\exists_{x_1 \ldots x_n}(x = p \,\&\, A_1)$, $\exists_{x_1 \ldots x_n}(\neg(x = p) \,\&\, A_2)$, и исходное кванторное утверждение заменяется на их дизъюнкцию. Уровень срабатывания приема равен 3. Прием гарантирует исключение связанной переменной x по крайней мере в первом дизъюнктивном члене.

Переход от условия существования к условию непустоты области определения отображения

Пусть в некотором терме встречаются одновременно утверждение $\exists_{x_1 \ldots x_n} A$ и выражение "отображение($y_1 \ldots y_n$ B t)", причем усматривается, что первое совпадает (с точностью до переобозначения связанных переменных и допустимых перестановок операндов) с утверждением $\exists_{y_1 \ldots y_n} B$. Тогда предпринимается замена утверждения существования на утверждение "не(равно(класс($y_1 \ldots y_n$ B) $\emptyset$))". Уровень срабатывания приема равен 4.

10.10 Приемы символа "равно"

Равенство относится к общелогическим символам, так как может встречаться в любой предметной области. Основая часть его приемов реализована на ЛОСе.

Равенство с совпадающими частями

Утверждение $a = a$ заменяется на логическую константу "истина". Уровень срабатывания этого приема равен 0.

Выдача ответа задачи на описание, единственное неизвестное условие которой имеет вид $x = t$

Если задача на описание имеет условие $x = t$, где x - неизвестная, t - выражение, не содержащее x, причем все прочие условия не содержат неизвестных, то обычно решение этой задачи прекращается, и для выдачи ответа предпринимается обращение к процедуре "редакторответа". Уровень срабатывания приема равен 0. Предварительно проверяется ряд требований, из числа которых приведем следующие:

1. Задача не имеет цели "редакция", цели "исследовать" либо цели "функционально". В первом случае ответ уже редактируется; во втором - нет необходимости явно выражать неизвестные через известные; в третьем - применяется особый прием.

2. Задача не имеет цели (известно ...). При такой цели нужно не просто выразить неизвестную через известные, но выразить ее через специальное подмножество известных. Схема решения задачи в этой ситуации - особая, использующая блок анализа.

3. Задача не имеет цели "замещение". Такая цель требует, чтобы неизвестные вообще не входили в ответ.

4. Если задача имеет цель "стандравно", т.е. возникла из задачи на исследование, имеющей цель "известно", то проверяется отсутствие условий, не имеющих свободных переменных. Такие условия эквивалентны логическим константам, и требуется сначала определить их истинность либо ложность. Это важно для усмотрения невыполнимых подслучаев (например, при анализе вариантов построения геометрического чертежа).

5. Задача не имеет цели (обозначение ...), некоторая переменная которой входит в условия. В этой цели перечисляются вспомогательные переменные, появление которых в ответе недопустимо.

Если задача имеет цели "полный", "пример", то, во-первых, проверяется допустимость найденного ответа $x = t$ с точки зрения цели (независит ...), и , во-вторых, предпринимается попытка усмотреть истинность всех известных условий в контексте списка посылок. После этого известные условия отбрасываются, и лишь затем реализуется обращение к процедуре "редакторответа".

Выдача ответа задачи на описание, имеющей цель "функционально"

Если задача имеет цель "функционально", то в ней нужно лишь установить факт однозначного определения значений неизвестных по значениям известных параметров. Явного выражения для неизвестных получать не требуется, причем в процессе решения можно вводить новые параметры, выражаемые однозначно через исходные. Такие задачи возникают в комбинаторике, при замене переменных в параметрическом задании класса объектов. Если в процессе решения для каждой неизвестной x, не являющейся несущественной, обнаруживается условие вида $x = t$, где t известно, то в качестве ответа выдается конъюнкция условий задачи. Уровень срабатывания приема равен 0.

Перестановка левой и правой частей равенства

Равенство $t_1 = t_2$ в посылках или условиях задачи определяет два различных обозначения t_1, t_2 одного и того же объекта. Обычно оно используется для стандартизации таких обозначений - t_1 повсюду в "зоне действия" равенства заменяется на t_2. Исключение составляют случаи, оговоренные специальными комментариями к равенству либо к заменяемому вхождению терма t_1. Однако, целесообразно переходить от "более сложных" в том или ином смысле обозначений к "более простым". Поэтому до применения замены, основанной на равенстве, предпринимается такая перестановка его частей, чтобы слева располагалось "более сложное" выражение. Это происходит при текущем уровне, равном 1. Заметим, что прием замены срабатывает тоже на уровне 1, но программа приема, переставляющего части равенства, расположена до программы приема замены.

Для определения целесообразности перестановки частей равенства служит оператор "ориентацияравенства(x1 x2 x3 x4)". Ему сообщается координата (x2,x3,x4) вхождения равенства в задачу x1. Если перестановка частей необходима, то оператор истинен, иначе - ложен. Перестановку частей равенства - посылки либо условия - блокирует комментарий "ориентацияравенства" к данному терму задачи. В ряде случаев этот комментарий игнорируется (наиболее часто - для некорневых вхождений равенства). Пусть процедуре "ориентацияравенства" переданы координаты вхождения

равенства, имеющего части t_1, t_2 (в произвольном порядке). Приоритеты при перестановке этих частей определяются следующим списком ситуаций (каждая очередная рассматривается лишь при условии, что предыдущие не имели места):

1. Если t_1 - неизвестная, а заголовком t_2 служит символ "отображение", причем задача - на исследование и имеет цель "иследовать", то t_1 размещается справа;

2. Если задача - на исследование и имеет цель "исследовать"; t_1 - переменная, t_2 - неоднобуквенный терм, не содержащий символа "отображение" и переменной t_1, то t_1 размещается слева;

3. Если равенство подчинено описателю "класс"; все промежуточные символы между символом этого описателя и равенством - либо конъюнкции, либо кванторы существования, причем t_2 - переменная кванторной приставки описателя, а t_1 - нет, и t_2 не встречается в t_1, то t_2 размещается слева;

4. Пусть задача - на описание и имеет цели (связка ...), "учетответа", т.е. происходит редактирование параметрического описания для решения системы функциональных (например, дифференциальных уравнений). Если t_1 имеет заголовок "значение" и содержит переменные цели "связка", а t_2 - не содержит таких переменных, то t_1 размещается слева;

5. Если равенство расположено в условиях задачи на описание, либо задача - на исследование, причем t_1 содержит неизвестные, а t_2 - не содержит, то t_1 размещается слева;

6. Если равенство расположено в условиях задачи на описание, либо задача - на исследование и не имеет цели "известно"; обе части равенства содержат неизвестные, причем t_1 - неизвестная и не входит в t_2, то t_1 размещается слева. Если оба терма t_1, t_2 - неизвестные, то перестановки не происходит;

7. Если задача - на исследование и имеет цель "известно", t_1, t_2 - неизвестные этой задачи, причем одна из них является неизвестной внешней задачи на описание, а другая - не является, то последняя размещается слева;

8. Если задача - на исследование и имеет цель "известно", t_2 - неизвестная этой задачи, принимающая численные значения, а t_1 не содержит t_2 и содержит неизвестную, принимающую векторные значения, то t_2 размещается справа;

9. Пусть задача - на исследование и имеет цель "известно"; t_1 - неизвестная, принимающая числовые либо векторные значения и не входящая в терм t_2, который не содержит неизвестных, принимающих не числовые и не векторные значения. Тогда t_1 размещается слева. Если оба терма - неизвестные, перестановка не выполняется;

10. Если задача - на исследование и имеет цель "известно"; t_1 содержит неизвестную, принимающую не числовые и не векторные значения, а t_2 не содержит такой неизвестной, то t_1 размещается слева;

11. Если задача - на исследование и имеет цель "известно"; t_1 принимает числовое либо векторное значение и является либо переменной, либо операцией от операндов, хотя бы один из которых не числовой и не векторный; t_2 - операция от числовых либо векторных операндов, то t_1 размещается слева;

12. Если задача - на доказательство, то для нее предусмотрены аналоги последних семи пунктов. При этом неизвестной задачи считается любой ее параметр, не входящий в комментарий (известно ...);

13. Если задача - на доказательство, t_1 - параметр, по которому ведется индукция и не входит в t_2, то t_1 размещается справа;

14. Если равенство - не корневое; t_1 - переменная, не входящая в t_2 и связанная внешним квантором либо описателем; t_2 - либо не переменная, либо не связано внешними кванторами и описателями, то t_2 размещается справа;

15. Если равенство - корневое; t_1 представляет собой переменную, а t_2 имеет заголовок "отображение" и не содержит t_1, то проверяется наличие посылки либо условия, к контексту которого относится равенство, в которое переменная t_1 входит не под символом "значение" либо область. Если такая посылка (условие) нашлась, то t_1 размещается справа;

16. Если равенство представляет собой посылку задачи на описание, имеющей цель (независит ...), причем t_1 не имеет связанных переменных, а t_2 имеет, то t_1 размещается справа;

17. Если текущая задача не имеет типа "исследовать" либо имеет цель "противоречие", причем t_1 - переменная, а t_2 не является переменной и не содержит t_1, то t_1 размещается слева;

18. Если t_1 содержит переменную, а t_2 не содержит переменных, то t_1 размещается слева;

19. Если t_1 имеет заголовок "значение", а t_2 не имеет такого заголовка и не содержит t_1, то t_1 размещается слева;

20. Если равенство представляет собой консеквент корневой кванторной импликации; текущая задача - на исследование, причем t_1 содержит атомарные числовые выражения, отличные от констант, переменных и условных выражений, а t_2 таковых не содержит, то t_1 размещается слева. Этим достигается приведение кванторной импликации к такому виду, чтобы ее можно было использовать для выражения числовых характеристик нечисловых объектов через числовые параметры;

21. Если t_1 имеет меньшую длину, чем t_2, то t_2 размещается слева;

22. Если терм t_1 лексикографически предшествует терму t_2 (в силу предыдущего пункта, длины этих термов равны), то в случае корневого равенства t_1 размещается справа, иначе - слева.

Заметим, что кроме данного общего приема, перестановка частей равенства выполняется множеством специальных приемов, реализованных на ГЕНОЛОГе. Эти приемы, чтобы закрепить выполненную ими перестановку, сопровождают содержащий равенство терм задачи комментарием "ориентацияравенства". Он отменяет перестановку частей равенства "из общих соображений". Иногда по ходу преобразований фиксация ранее введенной ориентации равенства устаревает, и требуются специальные приемы для ее коррекции - либо путем удаления комментария "ориентацияравенства", либо за счет приемов, игнорирующих этот комментарий и выполняющих перестановку частей равенства с учетом текущей ситуации.

Подстановка значения согласно равенству из текущего контекста

Пусть в задаче обнаружено равенство $t_1 = t_2$, причем текущий уровень равен 1 либо 6. Тогда рассматриваются все вхождения выражения t_1, к контексту которых относится данное равенство, и анализируется целесообразность замены этих вхождений на t_2. Программа приема, выполняющего эти действия, распадается на несколько ветвей, относящихся к различным ситуациям:

1. Вхождение равенства - корневое, т.е. равенство представляет собой некоторую посылку либо условие задачи. Проверяется выполнение нескольких общих условий, при нарушении которых замена нецелесообразна. Прежде всего, устанавливается, что t_1 не входит в t_2 и что текущая задача не является задачей на описание, имеющей цель "известно". Если равенство представляет собой посылку задачи на преобразование, причем t_1 - переменная, не входящая в терм t_2, который имеет достаточно большую длину и не является десятичной константой, то находится число вхождений переменной t_1 в условие задачи. Если это число более 3, то замены блокируются. Если задача не имеет типа "преобразовать", то замены выполняются только на уровне 1, иначе - добавляется попытка на уровне 6. Далее, проверяется отсутствие комментария "блок" к рассматриваемому равенству. Такой комментарий используется для управления данным приемом - вводится заблаговременно другими приемами, если применение равенства для стандартизации обозначений нежелательно. После того, как предварительный анализ целесообразности завершен, рассматриваются два подслучая:

   (a) Заголовок выражения t_1 не является ассоциативным и коммутативным символом. В случае задачи на преобразование вводятся дополнительные сильные ограничения на целесообразность замены: либо t_1 - переменная, не входящая в t_2, либо t_1 имеет заголовок "область", либо t_1 имеет свободные переменные, а t_2 - не имеет. Фактически, прием будет применяться для задач на преобразование лишь в исключительных случаях, так как плохо учитывает целевую специфику задачи.

   Рассматриваются все такие посылки и условия A, к контексту которых относится данное равенство. Для текущего A выполняется цикл проверок целесообразности замены. В основном, они относятся к редко встречающимся специальным случаям. Выделим лишь проверку отсутствия комментария "равно" к A. Такой комментарий используется, если нужно защитить какую-либо посылку либо условие от замен подтермов, выполняемых описываемым приемом. Если множество вхождений t_1 в A непусто, то находится результат B замены всех этих вхождений на t_2. Если этот результат представляет собой равенство - посылку задачи на исследование, имеющей цель "известно", то может быть предпринята попытка его упрощения с помощью пакетных операторов общей стандартизации. Кроме того, может быть выполнена перестановка частей такого равенства. После проверки ряда дополнительных условий целесообразности замены A на B, такая замена выполняется. В особых случаях вместо замены может быть предпринят вывод следствия B при сохраненном A.

   По окончании просмотра всех термов A предпринимается коррекция тех комментариев, где нужна синхронная с посылками и условиями замена t_1 на t_2. Особо рассмотрен случай комментария (сопровождение ...).

(b) Заголовок выражения t_1 - коммутативный и ассоциативный символ f. Таким образом, t_1 имеет вид $f(r_1 \dots r_m)$. Этот случай аналогичен предыдущему, однако рассматриваются не явные вхождения терма t_1 в текущий терм A, а вхождения подтермов $f(\dots)$, операнды которых включают все термы $r_1, \dots, r_m$. Замена затрагивает только последние - вместо них вводятся корневые f - операнды терма t_2. Для случая задач на преобразование предусмотрены еще более сильные ограничения - замены предпринимаются только при условии, что t_1 имеет свободные переменные, а t_2 не имеет.

2. Рассматриваемое вхождение равенства $t_1 = t_2$ не является корневым. Тогда уровень срабатывания приема равен 1. Равенство должно являться либо операндом конъюнкции, либо антецедентом кванторной импликации. Предусмотрена проверка нескольких простых и редко нарушаемых условий целесообразности замены. Например, замена блокируется, если равенство относится к стандартному уравнению поверхности. Случай коммутативного и ассоциативного заголовка t_1 особо не выделяется. Просматриваются подчиненные той же конъюнкции или импликации, что и равенство, вхождения подтерма t_1, в контекст которых попадает равенство. Они заменяются на t_2. Как и в предыдущих случаях, корректируется комментарий (сопровождение ...).

Описанный прием стандартизации обозначений, использующий равенство, является чрезвычайно общим и часто применяемым. По мере обучения решателя он сопровождался большим количеством различных эвристических ограничителей. Этот процесс, очевидно, будет продолжаться по мере дальнейшего обучения, причем какие-то из созданных фильтров могут измениться, а какие-то - вообще оказаться исключенными. В некоторых разделах, например, в геометрии, данный прием, возможно, применяется сейчас излишне часто и приводит к образованию неоправданно громоздких выражений. Впрочем, проведенная регулировка решателя пока позволяет справляться с таким недостатком, а сколь-нибудь сильное дополнительное ограничение приема потребует значительной перерегулировки. Попытку такой оптимизации можно будет предпринять впоследствии. На данном этапе работы с решателем происходит лишь предварительное накопление информации, которая позволит создать следующие, более эффективные его версии.

Усмотрение ложности равенства из различия типов его частей

В логической системе предусмотрена некоторая древовидная классификация типов объектов. Если два объекта относятся к различным ветвям этой классификации, то они заведомо различны. Чтобы распознавать одноместные предикаты, задающие тип объекта, служит справочник "родобъекта". Такими предикатами являются: "число(x)", "комплексное(x)", "целое(x)", "точка(x)", "множество(x)", и т.п. Для указания на список всех надтипов данного типа используется справочник "род". Например, на логическом символе "целое" он выдает набор символов "рациональное", "число", "комплексное". Список основных типов объектов пополняется по мере обучения решателя; вся информация о нем заключена в указанных справочниках. Кроме того, введен справочник "тип", указывающий по заголовку операции либо константы набор типов объектов, которые могут являться ее значениями. Иногда здесь приводится полный список, но чаще - только минимальные типы. Например, может быть указан тип "число", но отброшен тип "комплексное". Процедура "родобъекта(...)"

позволяет определить набор типов объектов, которые могут являться значениями заданного подвыражения задачи. Она используется специальным приемом, сравнивающим такие два списка для левой и правой частей усмотренного в задаче равенства. Если данные списки не пересекаются, то равенство заменяется на константу "ложь". Уровень срабатывания приема равен 3.

Усмотрение различия константных выражений, являющихся именами

Некоторые константные выражения используются в качестве имен объектов; понятие "имя" при этом предполагает различие объектов, имеющих различные имена. Например, именами являются десятичные записи чисел. Кроме того, предполагается, что однобуквенные выражения, являющиеся логическими символами, также суть имена. Такое соглашение, разумеется, запрещает вводить различные логические символы для обозначения одного и того же объекта. Вообще, всегда считается, что значением однобуквенного выражения, образованного логическим символом, служит сам этот символ, с которым и отождествляется обозначаемый им объект предметной области. На уровне 0 срабатывает прием, распознающий в левой и правой частях равенства два различных имени. В такой ситуации он заменяет равенство на константу "ложь".

Исключение посылки задачи на доказательство, имеющей вид равенства переменной, не встречающейся в других посылках и условии

Если посылка задачи на доказательство имеет вид $a = t$, где a - переменная, не встречающаяся в выражении t, а также в прочих посылках, условии и в комментариях (известно ...), (неизвестные ...), то эта посылка удаляется. Уровень срабатывания приема равен 1.

Применение тождества из посылок для исключения переменной в условии задачи на преобразование

Пусть посылка задачи на преобразование имеет вид $t_1 = t_2$, причем замена всех вхождений выражения t_1 в условие задачи на выражение t_2 позволяет избавиться в этом условии от некоторой переменной x, не добавляя новых переменных. Если задача не имеет других целей, кроме цели "упростить", то указанная замена реализуется. Уровень срабатывания приема равен 5.

Исключение неизвестной задачи на описание, явно выраженной через остальные неизвестные

Если в условиях задачи на описание возникает равенство $x = t$, где x - неизвестная, t - выражение, не содержащее этой неизвестной, то обычно предпринимается переход к решению новой задачи, не содержащей x. Уровень срабатывания приема, выполняющего такой переход, равен 1.

Если неизвестная x является несущественной, прочие условия задачи не содержат x, и нет комментария (контекст ...), в котором упоминается x, то происходит отбрасывание условия $x = t$, и работа приема на этом завершается. Предварительно проверяется отсутствие цели (независит ...), запрещающей неизвестной x зависеть от какой-либо переменной выражения t.

Если указанная ситуация не имеет места, то проверяется, что задача не имеет целей (известно ...), "редакция" и число ее неизвестных не менее 2. Если цель (независит ...) запрещает неизвестной x зависеть в ответе задачи от какой-либо свободной переменной выражения t, то прием блокируется. В противном случае создается вспомогательная задача на описание Z'. Ее посылки - те же, что у текущей задачи Z, но дополнительно вводится фиктивная посылка - "фиктпосылка(x)". Она нужна, чтобы процедуры, которые при последующем решении будут выбирать новые переменные, знали, что переменная x уже занята. Хотя эта переменная исключается из условий решаемых далее задач, но на этапе завершающего редактирования ответа равенство $x = t$ будет извлечено из комментария (контекст ...), и переменная x вновь появится. Условиями задачи Z' являются результаты подстановки выражения t вместо переменной x во все остальные условия задачи Z. Комментарии к задаче Z' переносятся из Z с помощью справочника "редукция". Цели корректируются соответственно исключению неизвестной x - она удаляется из списков неизвестных и несущественных неизвестных. Для измененных условий задачи Z' корректируются комментарии (сопровождение ...). Если неизвестная x не являлась несущественной, то все несущественные неизвестные выражения t рассматриваются в задаче Z' как существенные. В задачу Z' переносятся из задачи Z комментарии "разборслучаев" к дизъюнктивным условиям.

Для сохранения информации об отброшенном условии $x = t$ служит комментарий (контекст A B). У него B - список всех отброшенных условий, возникавших на различных этапах решения; A - задача, цели которой следует учитывать при завершающем редактировании ответа, после добавления к нему утверждений списка B. Прием, выдающий ответ на тот или иной подслучай задачи Z', обратится к процедуре "редакторответа", осуществляющей учет комментария (контекст ...). Она пополнит ответ утверждениями списка B, подставит в добавленные утверждения найденные значения неизвестных и упростит результат. Для этого будет решаться еще одна вспомогательная задача на описание, снабженная целью "редакция".

После регистрации условия $x = t$ в комментарии (контекст ...) предпринимается обращение к решению задачи Z'. Для полученного ответа R проверяется выполнение ограничений, накладываемых на x целью (независит ...). Затем корректируются комментарии (выводимо ...),(сопровождение ...), и выдается ответ R.

Выдача ответа задачи на описание, если все ее условия суть равенства, определяющие численные значения неизвестных

Если все условия задачи на описание суть либо равенства $x = t$, где x - неизвестная, t - число, либо утверждения "число(x)", то предусмотрена ускоренная выдача в качестве ответа конъюнкции условий. Предварительно проверяется отсутствие комментария (контекст ...), накапливающего ранее найденные фрагменты ответа. Проверяется также, что для каждой неизвестной задачи имеется равенство, определяющее ее значение. Уровень срабатывания приема равен 0.

Равенство в условиях задачи на описание, явно выражающее один из параметров через другие параметры

Если условием задачи на описание является не содержащее неизвестных равенство $a = t$, где a - переменная, не входящая в выражение t, то предпринимается попытка

перейти к решению вспомогательной задачи Z', полученной из текущей задачи подстановкой выражения t вместо a как в посылках, так и в условиях. Предварительно проверяется, что задача не имела цели (известно ...), а указанное равенство возникло при разборе случаев как фрагмент одной из альтернатив. Перед решением задачи Z' предпринимается упрощение максимальных подвыражений ее посылок и условий, изменившихся после подстановки. После получения на задачу Z' отличного от символа "отказ" ответа R формируется конъюнкция $x = t \ \& \ R$, которая и выдается как ответ на текущую задачу. Уровень срабатывания приема равен 1.

Подбор примера объекта, отличного от заданных объектов

Пусть все условия задачи на описание суть $x \neq t_1, \ldots, x \neq t_n$, где x - единственная неизвестная задачи; $t_1, \ldots, t_n$ - выражения без неизвестных. Пусть также задача имеет цель "пример". Если все выражения $t_1, \ldots, t_n$ суть имена, то находится некоторое число a, отличное от всех этих имен, и рассматривается равенство A вида $x = a$. Иначе в качестве A берется равенство "$x =$ внешнийэлемент$(\{t_1, \ldots, t_n\})$". Если задача не имела комментария (контекст ...), содержащего ранее найденные фрагменты ответа, то A выдается как ответ. Иначе все условия текущей задачи заменяются на единственное условие A, и для получения ответа предпринимается обращение к процедуре "редакторответа". Уровень срабатывания приема равен 2.

Вывод в посылках из кванторной импликации, антецедент которой имеет вид равенства, путем унификации частей этого равенства

Пусть в посылках текущей задачи встречается кванторная импликация вида $\forall_{x_1 \ldots x_n}$ $(t_1 = t_2 \rightarrow A)$ либо вида $\forall_{x_1 \ldots x_n}(B \rightarrow \neg(t_1 = t_2))$, где A, B - элементарные утверждения; выражение t_1 не содержит переменных кванторной приставки, а выражение t_2 содержит все эти переменные. Если текущая задача - на описание и имеет цель "пример", либо на доказательство, то предпринимается попытка подобрать такие выражения $s_1, \ldots, s_n$, подстановка которых в t_2 вместо переменных $x_1, \ldots, x_n$ дает (с точностью до простейших тождественных преобразований) выражение t_1. Если это удается, то находится результат C подстановки в A (соответственно, в $\neg B$) выражений $s_1, \ldots, s_n$ вместо $x_1, \ldots, x_n$, который выводится в качестве следствия. Уровень срабатывания приема равен 2.

Решение относительно параметра числового равенства без неизвестных

Если в условиях задачи на описание встречается числовое равенство $t_1 = t_2$, не содержащее неизвестных и не являющееся выражением переменной через другие переменные, то предпринимается попытка разрешить его явным образом относительно какого - либо встречающегося в нем атомарного числового параметра - переменной или числовой характеристики нечисловых объектов. Такая же попытка предпринимается для условия вида $\neg(t_1 = t_2)$, не содержащего неизвестных, если только оно имеет единственную свободную переменную. Предварительно проверяется ряд условий на целевую установку задачи: отсутствие целей (известно ...), "замещение", "стандравно" и др. Проверяется также, что рассматриваемое утверждение не используется для сопровождения по о.д.з. Выбор атомарного числового параметра и обращение к

вспомогательной задаче для разрешения относительно него выполняются процедурой "определениепараметра". Процедура ищет такой параметр, для которого уравнение будет иметь самый простой вид. Если после решения уравнения получается дизъюнкция, содержащая в своих подслучаях явные выражения для какого-либо известного параметра, встречающегося в условиях с неизвестными, то эта дизъюнкция сопровождается комментарием "разборслучаев". Уровень срабатывания приема равен 3.

Решение относительно числового параметра равенства, содержащегося в списке посылок

Если числовое равенство $t_1 = t_2$ встречается в списке посылок, то может быть применена процедура разрешения его относительно некоторого числового атома, аналогичная описанной в предыдущем пункте. Предварительно проверяется ряд условий на целевую установку задачи, из которых выделим следующие:

1. Если задача - на исследование, то она имеет цель "противоречие";
2. Если задача - на описание, то она не имеет целей "редуцирование", "замещение", (известно ...).
3. Если задача - на преобразование и не имеет целей, кроме цели "упростить", то все переменные равенства входят в ее условие;
4. Если задача - на преобразование, то не имеет целей "известны", "частнпроизв", "длина";

Проверяется, далее, что в равенство не входят описатели "класс", "отображение" и что оно не является явным выражением одной переменной через другие. Для выбора числового атома и разрешения равенства применяется та же процедура "определениепараметра", что и в предыдущем приеме. Уровни срабатывания приема - 1, 3, 5. На каждом из них устанавливается свой (равный, соответственно, 2, 5 и 20) максимальный уровень вспомогательной задачи, обеспечивающей разрешение. В приеме особо рассматривается случай комплекснозначного равенства, где применяется аналогичная процедура "Определениепараметра".

Возвращение к условиям задачи на описание после того, как анализ объединенного списка условий и посылок привел к явному выражению для одной из неизвестных

Если при решении задачи на исследование Z', являющейся блоком анализа задачи на описание Z, получается равенство вида $x = t$, где x - неизвестная задачи Z; t - выражение, не содержащее x, то может быть предпринят обрыв решения задачи Z' и возвращение к решению задачи Z, с перенесением в нее найденного равенства. Это делается лишь в тех случаях, когда задача Z' не имеет целей "известно", "контрольвывода", "исследовать". Кроме равенства $x = t$, в задачу Z переносится ряд сопутствующих утверждений, и из условий задачи Z исключаются те утверждения, которые являются следствиями перенесенных в нее новых утверждений. Это делает процедура "замещениеусловий". Уровень срабатывания приема равен 0. Как видно

из приведенных выше ограничений, для задач с целью "известно" прием заблокирован. Это объясняется тем, что здесь нецелесообразно спешить с передачей во внешнюю задачу части ответа - экономнее сначала найти весь ответ в рамках задачи на исследование. Поэтому введен специальный прием, анализирующий равенства $x = t$ для задач на исследование, имеющих цель "известно" (см. ниже). При усмотрении полного ответа он обеспечивает его упрощение и выдачу.

Исключение внешней фиктивной операции для равенства в консеквенте кванторной импликации

Пусть кванторная импликация имеет вид $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_m \to f(x_i t_1) = f(x_i t_2)$, где переменная x_i не встречается в t_1, t_2 и в существенных посылках A_j. Тогда проверяется, что операция f имеет единицу a по переменной x_i; эта единица подставляется во все антецеденты, содержащие x_i, и проверяется, что они являются следствиями прочих антецедентов. После этого выполняется замена кванторной импликации на утверждение $\forall_{x_1 \ldots x_{i-1} x_{i+1} \ldots x_n}(A_{j_1} \ \& \ \ldots \ \& \ A_{j_k} \to t_1 = t_2)$, где $A_{j_1}, \ldots, A_{j_k}$ - все не содержащие x_i антецеденты. Уровень срабатывания приема равен 1.

Попытка решения уравнения из блока анализа, имеющего единственную неизвестную

Если вывод следствий в блоке анализа Z' некоторой задачи на описание Z дает уравнение $t_1 = t_2$, содержащее единственную числовую неизвестную x, то может быть предпринято обращение к вспомогательной задаче на описание для разрешения данного уравнения относительно x. Уровни, на которых предпринимается эта попытка, равны 1, 3 и 5. При наличии в уравнении неизвестных тригонометрических подвыражений допускаются только уровень 5, либо, для не очень длинных уравнений, уровень 3. Далее проверяется выполнение следующих требований:

1. Уравнение не дает явного выражения для x;
2. Из контекста усматривается, что x - числовая переменная;
3. Не существует дизъюнктивно-конъюнктивной посылки, фрагмент которой дает явное выражение для x;
4. Рассматриваемое уравнение не было перенесено в блок анализа из списка условий внешней задачи на описание, имеющей единственную неизвестную и не имеющей цели (известно ...);
5. Не существует более короткого уравнения, удовлетворяющего тем же условиям, для которого еще не выполнялась попытка применения данного приема;

Тогда создается вспомогательная задача на описание Z'', посылками которой служат все не содержащие неизвестных посылки задачи Z', а единственным условием - рассматриваемое уравнение. Цели этой задачи суть "полный", "прямойответ", "или", "явное", (неизвестные x). Если задача Z' имеет цель "известно", то к данному списку добавляются также цели "упростить" и "стандравно". Заметим, что последняя цель используется во всех вспомогательных задачах на описание, образованных подсистемой уравнений задачи на исследование Z', имеющей цель "известно". Она указывает

на возможность применения несколько упрощенной процедуры решения, учитывающей потребности задачи Z'.

Условия и посылки задачи Z'' обычно бывает целесообразно пополнить некоторыми дополнительными утверждениями, извлекаемыми из списка посылок блока анализа. Для такого пополнения в процессе обучения решателя были отобраны правила, перечисляемые ниже. Заметим, что занесение в Z'' слишком сложных сопровождающих утверждений, ограничивающих допустимые значения неизвестной x, оказалось нецелесообразным - во многих случаях оно сильно замедляло решение задачи. Столь же нежелательным и по той же причине оказался и недобор сопровождающих утверждений. Поэтому понадобилась эмпирическая оптимизация, после которой возник приводимый ниже перечень правил. Они формируют объединенный список S дополнительных условий и посылок:

1. В S включаются все утверждения, необходимые для сопровождения уравнения $t_1 = t_2$ по о.д.з. и имеющие свободные переменные;

2. Если имеется посылка "целое(x)", то она включаестя в S;

3. Если есть основания предположить, что при решении задачи будет полезна информация о неотрицательности x (например, если x - значение заведомо неотрицательной числовой характеристики нечислового объекта, либо расположено под достаточно "сложной" операцией), то выполняется быстрая проверка неотрицательности x. В случае успеха список S пополняется утверждением $0 \leq x$.

4. Если блок анализа имеет посылку вида "расстояние(AB)= r"; r - числовое выражение, все параметры которого встречаются в задаче Z'', то в S заносится неравенство $0 \leq r$, в правой части которого отбрасываются заведомо положительные множители;

5. Если уравнение содержит неизвестную x под тригонометрической функцией, то просматриваются посылки блока анализа, имеющие вид "угол(...)= r" либо "уголмежду(...)= r". В S заносится неравенство $0 \leq r$ или, если быстро удается усмотреть строгую положительность, неравенство $0 < r$. Если быстро удается усмотреть неравенство $0 \leq \pi - 2r$, то оно также заносится в S. Иначе - в S заносится $0 \leq \pi - r$.

6. Пусть в блоке анализа имеется равенство $T = r$, где r - либо переменная, либо известное выражение малой длины (например, не более 4). Пусть также T имеет своим заголовком один из символов "угол", "уголмежду", "расстояние", "площадь". Тогда в S заносится неравенство $0 \leq r$. Если T имеет вид "расстояние(AB)", причем удается усмотреть различие точек A, B, то данное неравенство заменяется на строгое. Если r - переменная, то далее рассматриваются следующие подслучаи:

   (a) Если заголовок T - символ "угол" либо "уголмежду", то предпринимается попытка усмотреть, что данный угол - острый, т огда в S заносится неравенство $0 < \pi - 2r$. При неудаче предпринимается попытка доказать, что угол меньше пи, и тогда заносится утверждение $0 < \pi - r$. В прочих случаях заносится утверждение $0 \leq \pi - r$;

   (b) Если заголовок T - символ "площадь", то предпринимается попытка установить ее положительность. При удаче в S заносится утверждение $0 < r$;

7. Если в блоке анализа имеется посылка $r = \sin T$, где r - переменная, встречающаяся в Z''; T - выражение с заголовком "угол" либо "уголмежду", то в S заносится неравенство $0 \leq r$;

8. Если в блоке анализа имеется посылка $r = T$, где r - переменная, встречающаяся в уравнении $t_1 = t_2$ в четной степени, то предпринимается попытка быстрого усмотрения неотрицательности выражения T. При удаче в S заносится неравенство $0 \leq r$.

9. Если неизвестная x встречается в уравнении внутри тригонометрического подвыражения, причем блок анализа имеет своей посылкой строгое либо нестрогое неравенство с $\pi/2 - x$ в одной части и 0 - в другой, то это неравенство заносится в S.

Далее предпринимается упрощение списка S - каждое входящее в него утверждение обрабатывается нормализатором "нормодз" в контексте прочих утверждений списка, а также посылок задачи Z''. Нормализатор "нормодз" уже встречался ранее - он использовался процедурой "одз" для первичной обработки сопровождающих утверждений.

Все содержащие неизвестную x утверждения списка S добавляются к условиям задачи Z'', а остальные - к ее посылкам. После этого предпринимается решение задачи Z'' с максимальным уровнем, равным 6. Если при решении получен "отказ"либо утверждение, отличное от константы "ложь" и не содержащее символа "равно" в каждом из своих дизъюнктивных членов, то уравнение $t_1 = t_2$ помечается комментарием "неизвестная", блокирующим повторные попытки применения приема. В противном случае конъюнктивные члены ответа замещают в задаче Z' рассматриваемое уравнение. Корректируются комментарии "выводимо", "условие", "прообраз".

Выдача полученного в блоке анализа ответа задачи на описание, имеющей цель "известно"

Пусть решается задача на исследование Z' с целью "известно", являющаяся блоком анализа задачи на описание Z, и в ней усмотрено равенство $x = t$, где t - выражение без неизвестных; x - неизвестная задачи Z. Тогда на уровнях 0, 3, 6 предпринимается попытка усмотреть в посылках задачи Z' ответ задачи Z. Прежде всего, проверяется отсутствие в t переменных, выделенных комментарием (вспомпараметр ...). Этот комментарий накапливает переменные, которые при решении задачи Z' вводились как вспомогательные "известные". Они не должны входить в ответ; после того, как значение такой переменной выражается через исходные "известные", она переносится в список неизвестных задачи Z'.

Затем проверяется, что для каждой неизвестной y задачи Z, не выделенной комментарием (вспомнеизвестная ...), среди посылок задачи Z' имеется равенство вида $y = r$, где r не содержит неизвестных и переменных, выделенных комментарием (вспомпараметр ...). Заметим, что комментарий (вспомнеизвестная ...) перечисляет вспомогательные неизвестные, вводившиеся при решении задачи Z'. Они одновременно регистрировались в задаче Z, однако в ответ задачи Z информация о их значениях включаться не будет. Эти неизвестные нужны были лишь для управления ходом решения задачи Z'.

Если задача Z' имеет цель "контроль", то проверяется отличие текущего уровня от 0. Данная цель указывает на то, что рассматривается некоторый подслучай исходной задачи. Вообще говоря, он может оказаться нереализуемым. Например, так часто бывает в планиметрических задачах. Чтобы усмотреть противоречие и отсечь подслучай, предусмотрена пауза перед выдачей ответа. На уровнях от 0 до 2 срабатывают приемы, усматривающие противоречивость посылок задачи Z'. Все они активизируются лишь при наличии цели "контроль".

Наконец, просматриваются всевозможные равенства $y = r$ указанного выше вида, и для каждого из них выполняется упрощение выражения r. Здесь последовательно решаются две задачи на преобразование. Первая из них, имеющая цели "упростить", "известны", предпринимает стандартные действия по упрощению. Уровень ее, как и следующей задачи, равен 4. Вторая задача, с целями "длина" и "учетрезультата", ориентирована на остаточные упрощения и сокращенную переформулировку результата. Цели "известны", "учетрезультата" указывают на данный прием выдачи ответа. В других задачах они не встречаются. После упрощений равенства замещают все ранее имевшиеся условия внешней задачи Z. Чтобы по возвращении к задаче Z сразу же был выдан ответ на нее, данная задача снабжается комментарием "ответ". В заключение прием обрывает решение задачи Z'. Таким образом, никакого дополнительного редактирования ответа в рамках внешней задачи Z уже не происходит - там немедленно выдается конъюнкция равенств, определяющих значения неизвестных.

Усмотрение и решение подсистем числовых уравнений в задаче на исследование, имеющей цель "известно"

Задачи на описание, имеющие цель "известно" (например, планиметрические задачи на вычисление), решаются путем вывода следствий в своем блоке анализа. Если на некоторый момент вывод следствий дает подсистему численных уравнений, в которой количество неизвестных равно количеству уравнений, то может быть предпринята попытка решить ее с помощью вспомогательной задачи. Имеется несколько различных приемов, реализующих данную попытку:

1. Выделение системы из двух уравнений с двумя неизвестными. Прием срабатывает на уровне 6. Его применение начинается с усмотрения уравнения $t_1 = t_2$, содержащего ровно две неизвестных x, y рассматриваемого блока анализа Z' задачи на описание Z, которые обе оказываются числовыми. Составляется список S всех уравнений, не содержащих отличных от x, y неизвестных блока анализа и не являющихся равенствами, выражающими одну неизвестную через другую. Проверяется, что S имеет не менее двух и не более трех элементов. Предпринимается переупорядочение уравнений списка S по возрастанию их длин. Для разрешения относительно неизвестных последовательно выбираются пары уравнений, одно из которых - текущее уравнение $t_1 = t_2$, а другое - отличный от него элемент списка S. Таким образом, в первую очередь рассматриваются более короткие дополнительные уравнения. С помощью комментария (неизвестные ...) блокируются повторные попытки разрешения ранее рассматривавшейся системы. Вводятся накопители P и U, соответственно, посылок и условий вспомогательной задачи на описание Z''. В программе они обозначены переменными x14 и x12. В список P заносятся все не содержащие неизвестных посылки задачи Z', имеющие общий параметр с отобранными уравнениями. В список U заносятся: отобранная для решения пара уравнений; утверждения

"число(x)" и "число(y)"; все неравенства и отрицания равенств, не содержащие неизвестных задачи Z', отличных от x, y; посылки, указывающие на целочисленность значений неизвестных x, y (если таковые имеются). Далее просматриваются всевозможные переменные z, совпадающие с одной из неизвестных x, y либо представляющие собой числовой параметр утверждения списка P. Для них предпринимается пополнение списков P, U перечисляемыми ниже дополнительными утверждениями. Если z - параметр утверждения списка P, то утверждение заносится в список P, иначе - в список U.

(a) Если имеется посылка $z = r$, где заголовок выражения r - один из символов "угол", "расстояние", "площадь", то добавляется утверждение $0 \le z$, где по мере возможности вместо нестрогого неравенства берется строгое. В случае угла предпринимаются попытки усмотреть и добавить неравенство, указывающее, что угол острый. Кроме того, вводится неравенство, указывающее, что угол не превосходит пи; по мере возможности - строгое.

(b) Если r - синус угла, то добавляется неравенство $0 \le z$;

(c) Если удается усмотреть, что z неотрицательно либо положительно, то добавляется соответствующее неравенство.

После этого формируется вспомогательная задача на описание Z''. Она имеет цели "полный", "явное", "прямойответ", "одз", "упростить", "стандравно", (неизвестные xy). Если какая - либо из неизвестных x, y была в задаче Z' выделена комментарием (вспомнеизвестная ...), то она регистрируется как несущественная неизвестная задачи Z''.

Задача Z'' решается с максимальным уровнем 6. Если на нее получен ответ R, отличный от символа "отказ", то проверяется наличие в R равенства, дающего явное выражение для какой-либо из неизвестных x, y. Если R содержит дизъюнкцию, то проверяется, что равенство указанного вида имеется либо в каждом ее дизъюнктивном члене, либо в той части R, которая относится к контексту дизъюнкции. Затем из списка посылок блока анализа Z' исключаются те уравнения, которые разрешались, и вместо них заносится утверждение R.

2. Решение системы из трех уравнений с тремя неизвестными. Уровень срабатывания приема равен 5. Он аналогичен предыдущему приему, однако возникает небольшое отличие при определении списка неизвестных. Если текущее уравнение $t_1 = t_2$ имеет ровно три числовых неизвестных x, y, z, то по ним составляется список S всех уравнений блока анализа, не содержащих других неизвестных, кроме x, y, z. Проверяется, что этот список имеет не менее 3 и не более 5 элементов, и далее выполняются те же действия, что в предыдущем приеме. Если текущее уравнение имеет только две числовых неизвестных x, y, то для определения третьей неизвестной z находится еще одно уравнение, имеющее ровно две числовых неизвестных - например, x и z. Затем по тройке неизвестных составляется список S, и далее - как для двух неизвестных.

3. Решение системы из двух уравнений с тремя неизвестными. Пусть блок анализа имеет всего два уравнения U_1, U_2 для каких-то трех своих числовых неизвестных x, y, z, однако есть уравнение $v = t$, выражающее еще одну числовую

неизвестную v внешней задачи на описание через x, y. Если можно предположить, что v выразимо через отношение x к y (например, после усмотрения в t дроби, числитель и знаменатель которой содержат неизвестные x, y), то вводится вспомогательная задача для решения системы $U_1, U_2, v = t$ относительно неизвестных y, z, v. При этом переменная x считается известной, так что все посылки с x, не содержащие других неизвестных блока анализа, регистрируются в посылках вспомогательной задачи. В остальном действия аналогичны случаю двух неизвестных. Уровень срабатывания приема равен 5.

4. Решение системы из четырех уравнений с четырьмя числовыми неизвестными. Уровень срабатывания приема равен 7. Если текущее уравнение $t_1 = t_2$ имеет ровно 4 числовых неизвестных x, y, z, v, то составляется список S всех уравнений блока анализа, неизвестные которых содержатся среди x, y, z, v. Проверяется, что этот список имеет ровно 4 элемента, и далее выполняются действия, аналогичные случаю двух неизвестных.

Преобразование вспомогательных параметров блока анализа, через которые выражена неизвестная внешней задачи на описание, во вспомогательные неизвестные

При решении задачи на исследование, имеющей цель "известно", могут возникать вспомогательные переменные, характеризующие те или иные объекты задачи (например, высота треугольника) и которые временно считаются известными. Такие переменные x помечаются комментариями (вспомпараметр x) к посылкам задачи. Равенства, определяющие их значения, временно присоединяются к списку посылок внешней задачи на описание Z. После того, как через вспомогательные параметры удалось выразить какую-либо неизвестную внешней задачи, либо выразить один из них через другие, происходит изменение статуса параметра - он объявляется вспомогательной неизвестной. Прием, выполняющий это преобразование, срабатывает на уровне 0. Он инициируется посылкой вида $y = t$, где y - либо неизвестная задачи Z, либо один из вспомогательных параметров; t - выражение, содержащее вспомогательные параметры. Прием составляет список S всех вспомогательных параметров, встречающихся в данной посылке. В задаче Z он исключает все посылки задачи Z, имеющие переменную списка S, и вводит для переменных v из S комментарии (вспомнеизвестная v). В задаче на исследование Z' - блоке анализа задачи Z - прием заменяет комментарии (вспомпараметр v), где $v \in S$, на комментарии (вспомнеизвестная v). В обоих задачах список неизвестных пополняется переменными из S.

Пополнение списка неизвестных задачи на описание параметрами числовых уравнений

Если количество числовых уравнений задачи на описание больше числа неизвестных, имеющихся в этих уравнениях, однако равно числу всех встречающихся в уравнениях числовых переменных - как известных, так и неизвестных, то может помочь прием, решающий данную систему уравнений в предположении, что все ее переменные - неизвестные. Уровень срабатывания приема равен 9. Он вводит вспомогательную задачу Z', полученную из текущей задачи Z объявлением неизвестными всех переменных S, содержащихся в числовых уравнениях, и перенесением в список условий всех посылок, зависящих от S.

10.11 Приемы символа "эквивалентно"

Символ "эквивалентно" встречается в задачах редко, главным образом в консеквентах кванторных импликаций. Для него понадобились лишь простейшие приемы, выполняющие общую стандартизацию. Приводимый ниже список не претендует на полноту - он лишь показывает, какие именно связанные с эквивалентностью преобразования оказались фактически востребованы.

Совпадающие операнды

Утверждение $A \leftrightarrow A$ заменяется на логическую константу "истина". Уровень срабатывания равен 0.

Противоположные операнды

Утверждение $A \leftrightarrow \neg A$ заменяется на логическую константу "ложь". Уровень срабатывания 0.

Заголовок каждого операнда - отрицание

Утверждение $\neg A \leftrightarrow \neg B$ заменяется на $A \leftrightarrow B$. Уровень срабатывания 0.

Заголовок одного операнда - отрицание, а другой операнд есть дизъюнкция либо конъюнкция отрицаний

Утверждение $\neg A \leftrightarrow (\neg B_1 \vee \ldots \vee \neg B_n)$ заменяется на $A \leftrightarrow (B_1 \,\&\, \ldots \,\&\, B_n)$. Двойственное преобразование применяется, если правая часть представляет собой конъюнкцию отрицаний. Уровень срабатывания 0.

Замена кванторной эквивалентности в условиях задачи на доказательство на две кванторных импликации

Если утверждение $\forall_{x_1 \ldots x_n}(A \rightarrow (B \leftrightarrow C))$ входит в условие задачи на доказательство, то оно заменяется на конъюнкцию утверждений $\forall_{x_1 \ldots x_n}(A \,\&\, B \rightarrow C)$, $\forall_{x_1 \ldots x_n}(A \,\&\, C \rightarrow B)$. Уровень срабатывания приема равен 0.

Замена кванторной эквивалентности на импликацию

В специальных случаях можно заменить кванторную эквивалентность на импликацию. Так, утверждение $\forall_{x_1 \ldots x_n}(C \rightarrow (A \vee B \leftrightarrow A))$ заменяется на $\forall_{x_1 \ldots x_n}(C \,\&\, B \rightarrow A))$. Утверждение $\forall_{x_1 \ldots x_n}(C \rightarrow (A \,\&\, B \leftrightarrow A))$ заменяется на $\forall_{x_1 \ldots x_n}(C \,\&\, A \rightarrow B))$. Уровень срабатывания приемов равен 0.

Выражение бесквантороной эквивалентности через дизъюнкцию, конъюнкцию и отрицание

Если утверждение $A \leftrightarrow B$ не расположено под квантором либо описателем и не является условием задачи на доказательство, то оно заменяется на $(A \,\&\, B) \vee (\neg A \,\&\, \neg B)$. Уровень срабатывания равен 1.

Равенство в одной из частей эквивалентности, дающее явное выражение для переменной

Если одна из частей эквивалентности имеет вид $x = t \ \& \ A(x)$, где x - переменная; t - отличное от переменной выражение, не содержащее x, то эта часть заменяется на $x = t \ \& \ A(t)$. Уровень срабатывания равен 2.

Различающиеся фрагменты частей эквивалентности имеют вид отрицаний

Если эквивалентность имеет вид $(\neg A \ \& \ B) \leftrightarrow (\neg C \ \& \ B)$, то она преобразуется к виду $(A \ \& \ B) \leftrightarrow (C \ \& \ B)$. Уровень срабатывания равен 2.

Перенесение в антецедент общего фрагмента двух частей кванторной эквивалентности

Кванторная эквивалентность вида $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_n \rightarrow (B \ \& \ C \leftrightarrow B \ \& \ D))$ преобразуется к виду $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_n \ \& \ B \rightarrow (C \leftrightarrow D))$. В условии задачи на описание, имеющей цель "экв", прием применяется на уровне 0; иначе - на уровне 5.

Кванторная эквивалентность двух равенств с общей переменной

Если кванторная эквивалентность имеет консеквент вида $x = t \leftrightarrow x = s$, где x - переменная кванторной приставки, а выражения s, t не содержат x, то в ее антецедентах предпринимается подстановка t вместо x, консеквент заменяется на $s = t$, а x исключается из кванторной приставки. Уровень срабатывания приема равен 2.

Кванторная эквивалентность допускает разбиение по независимым группам переменных

Если кванторная эквивалентность не имеет антецедентов, а кванторная приставка ее может быть разбита на две подгруппы переменных x и y так, что консеквент имеет вид $(P(x) \ \& \ Q(y)) \leftrightarrow (R(x) \ \& \ S(y))$, то она заменяется на дизъюнкцию утверждений $(\forall_x(\neg P(x)) \vee \forall_y(\neg Q(y))) \ \& \ (\forall_x(\neg R(x)) \vee \forall_y(\neg S(y)))$, $\forall_x(P(x) \leftrightarrow R(x)) \ \& \ \forall_y(Q(y) \leftrightarrow S(y))$. Уровень срабатывания приема равен 4.

Усиление контекста описателя "отображения", встречающегося в кванторной эквивалентности

Пусть кванторная эквивалентность имеет вид $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_n \rightarrow (B \leftrightarrow C \ \& \ D))$; утверждение B содержит описатель "отображение($z \ P \ t$)", причем все свободные переменные конъюнктивного члена C правой части эквивалентности входят в P, а любой другой конъюнктивный член этому условию не удовлетворяет. Тогда эквивалентность преобразуется в конъюнкцию утверждений $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_n \ \& \ C \rightarrow (B \leftrightarrow D))$, $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_n \ \& \ B \rightarrow C)$. Первое из них позволяет использовать при преобразованиях, затрагивающих описатель, дополнительные условия C. Уровень срабатывания приема равен 4.

Доказательство эквивалентности путем рассмотрения двух импликаций

Если условие задачи на доказательство Z имеет вид $A \leftrightarrow B$, то на текущем уровне 1 предпринимается последовательное решение двух вспомогательных задач. Первая из них получается добавлением к посылкам задачи Z утверждения A и заменой условия на B; вторая - добавлением к посылкам B и заменой условия на A. Если обе решены, выдается ответ "истина".

10.12 Приемы символа "вариант"

Большая часть общих приемов символа "вариант" реализована на ГЕНОЛОГе и будет рассмотрена в следующих разделах. На ЛОСе реализованы лишь несколько приемов.

Вынесение из-под условного выражения символа одноместной операции

Если условное выражение имеет вид "вариант(A $f(t_1)$ $f(t_2)$)", где f - символ одноместной операции, то оно преобразуется к виду f(вариант(A t_1 t_2)). Уровень срабатывания этого приема равен 1.

Непосредственное усмотрение истинности либо ложности условия

Если из контекста непосредственно усматривается истинность либо ложность условия A выражения "вариант(A t_1 t_2)", то оно заменяется на соответствующее выражение t_i. Здесь рассматриваются следующие случаи:

1. Дизъюнктивный член утверждения A содержится в контексте условного выражения. Тогда оно заменяется на t_1.

2. Дизъюнктивный член отрицания утверждения A содержится в контексте условного выражения. Тогда оно заменяется на t_2.

Прием срабатывает на нулевом уровне.

Усмотрение части конъюнктивных членов условия

Если часть конъюнктивных членов утверждения A встречается в контексте условного выражения "вариант(A t_1 t_2)", то эти члены в данном выражении отбрасываются. Уровень срабатывания равен 1.

Попытка установления истинности либо ложности условия путем решения задачи на доказательство

Пусть выражение "вариант(A t_1 t_2)" расположено в условии задачи на преобразование, либо в задаче на исследование, целевая установка которой указывает на достаточно высокую вероятность исключения условных выражений. Тогда предпринимается попытка установить истинность либо ложность утверждения A с помощью

обращений к вспомогательным задачам на доказательство. Они решаются с максимальным уровнем 4 и достаточно ограниченной трудоемкостью. При успехе проверки выражение заменяется на t_1 либо t_2. Уровень срабатывания приема равен 4 в случае задачи на преобразование и 0 в случае задачи на исследование. Прием блокируется, если условное выражение содержится под описателем "отображение", а утверждение A зависит от переменных связывающей приставки. Этот случай означает определение функции разбором случаев по значениям аргумента, и устранение "варианта" здесь маловероятно.

10.13 Символ "альтернатива"

Символ "альтернатива" практически никогда в задачах не встречается. Для него имеет единственный прием, реализованный на ЛОСе. Он заменяет утверждение "альтернатива(A $P(t_1$ ... $t_{i-1}p$ t_{i+1} ... $t_n)$ $P(t_1$ ... $t_{i-1}q$ t_{i+1} ... $t_n)$)" на утверждение $P(t_1$ ... t_{i-1}вариант(A p q) t_{i+1} ... $t_n)$. Уровень срабатывания приема равен 1.

10.14 Приемы символов, используемых при задании конечных упорядоченных наборов

Для символов "набор", "префикс", "суффикс", "конкатенация", "поднабор", "наборномеров", позволяющих создавать конечные упорядоченные наборы элементов, на ЛОСе реализован ряд простых приемов.

Равенство двух наборов

Равенство двух наборов "набор($t_1 \ldots$ t_n)" и "набор($p_1 \ldots p_m$)" преобразуется при $m = n$ в конъюнкцию равенств $t_1 = p_1$, ..., $t_n = p_n$. При $m \neq n$ оно заменяется на логическую константу "ложь". Прием срабатывает на уровне 0, однако блокируется, если рассматривается корневое отрицание равенства двух наборов, помеченное комментарием "набор". Такие отрицания равенств представляют собой удобную форму записи условий, которые невыгодно приводить к виду дизъюнкций. Например, условие невырожденности вектора с координатами p, q обычно записывается как $(p, q) \neq (0, 0)$.

Определение длины набора

Выражение "длинанабора(набор($t_1 \ldots t_n$))" преобразуется в десятичную запись числа n. Уровень срабатывания равен 0.

Определение элемента набора по его номеру

Выражение "значение(набор($t_1 \ldots t_n$)i)", где i - десятичная запись натурального числа от 1 до n, заменяется на t_i. Уровень срабатывания равен 0.

Устранение равенства некоторого выражения набору переменных, связанных внешними кванторами

Пусть равенство t = набор($x_1 \ldots x_n$) имеет в своей правой части попарно различные переменные, связанные внешними кванторами и не входящие в выражение t. Чтобы появилась возможность впоследствии применять приемы, исключающие данные связанные переменные, это равенство преобразуется к виду "и(слово(t) длина-набора(t)= n x_1 = значение(t 1) ... x_n = значение(t n))", где эти переменные явно выражены через t. Прием блокируется для некоторых ситуаций, встречающихся в аналитической геометрии (например, если t - выражение вида "коорд(...)", определяющее набор координат точки либо вектора). Уровень срабатывания равен 1.

Приемы символа "префикс"

Несколько простых приемов связаны с исключением символа "префикс". Они срабатывают на уровне 1.

1. Выражение "префикс(a набор($b_1 \ldots b_n$))" преобразуется к виду "набор(a $b_1 \ldots b_n$)".
2. Выражение "префикс(a пустоеслово)" заменяется на "набор(a)".
3. Выражение "префикс(a конкатенация(набор($b_1 \ldots b_n$)$c_1 \ldots c_m$))" заменяется на "конкатенация(набор(a $b_1 \ldots b_n$)$c_1 \ldots c_m$)".
4. Выражение "префикс(a конкатенация($c_1 \ldots c_m$))", где выражение c_1 не имеет своим заголовком символа "набор", преобразуется к виду "конкатенация(набор(a)$c_1 \ldots c_m$)".
5. Выражение "префикс(a префикс(b c))" заменяется на выражение "конкатенация(набор(a b)c)".

Приемы символа "конкатенация"

Следующие приемы символа "конкатенация" срабатывают на нулевом уровне:

1. Если в выражении "конкатенация(...)" идущие подряд операнды имеют заголовок "набор", то они объединяются в один операнд с заголовком "набор".
2. Если выражение "конкатенация(...)" расположено под такой операцией, для которой порядок элементов в наборе несущественен, то предпринимается извлечение всех операндов с заголовком "набор", объединение их в общий операнд с заголовком "набор", и перенесение данного операнда на первое место. Для срабатывания приема необходимо, чтобы либо операндов "набор(...)" было больше одного, либо чтобы единственный такой операнд не был расположен на первом месте.
3. Выражение "конкатенация(набор(a)b)", где операнд b не имеет заголовка "набор", преобразуется к виду "префикс(a b)".
4. Если в выражении "конкатенация(...)" вслед за операндом вида "набор($a_1 \ldots a_n$)" расположен операнд вида "префикс(b c)", то первый операнд заменяется на "набор($a_1 \ldots a_n b$)", а второй - на c.

5. Если в выражении "конкатенация(...)" операнд "префикс(a b)" не устраняется одним из описанных выше приемов, то он заменяется на два идущих подряд операнда "набор(a)", b.

6. В выражении "конкатенация(...)" отбрасываются операнды "пустоеслово".

Если выражение "конкатенация(...)" находится под операцией, для которой несущественен порядок элементов набора, то на уровне 3 предпринимается лексикографическое упорядочение ее операндов. Оно не затрагивает первого операнда "набор(...)", если таковой имеется.

Приемы символа "суффикс"

Имеется всего два таких приема, реализованных на ЛОСе. Уровень срабатывания их равен 1. Первый заменяет выражение "суффикс(набор($a_1 \dots a_n$)b)" на "набор($a_1 \dots a_n$ b)". Второй - заменяет выражение "суффикс(пустоеслово a)" на "набор(a)".

Приемы символов "поднабор", "наборномеров"

Эти символы имеют каждый по одному приему. Первый прием заменяет выражение "поднабор(набор($a_1 \dots a_n$)i j)" на выражение "набор($a_i a_{i+1} \dots a_j$)". Второй - заменяет выражение "наборномеров (i j)" на выражение "набор(i $i+1$ ... j)". Уровни срабатывания этих приемов равны 0.

10.15 Приемы символа "класс"

Общих приемов символа "класс", реализованных на ЛОСе, немного, и все они несложны. Общие приемы этого символа, реализованные на ГЕНОЛОГе, составляют значительно больший список; они собраны в разделе базы приемов, относящемся к алгебре множеств. Чтобы избежать известных парадоксов теории множеств, на использование описателя "класс" накладываются определенные ограничения. Во избежание появления слишком "больших" множеств, встречающихся в парадоксах, разрешается применять данный описатель лишь таким образом, чтобы он выделял подмножество в каком-либо ранее уже построенном множестве. Этот подход хорошо известен в аксиоматической теории множеств.

Нормализация связывающей приставки

На нулевом уровне предпринимается такое переобозначение переменных связывающей приставки описателя "класс(...)", чтобы они не имели свободных вхождений в утверждения, относительно которых рассматривается данный описатель.

Исключение описателя

Выражение "класс(x принадлежит(x t))" заменяется на t. Здесь t - выражение, не содержащее x. Уровень срабатывания приема равен 0.

Условие принадлежности классу

Утверждения "принадлежит(t класс(x $P(x)$))", "принадлежит(набор($t_1 \dots t_n$) класс($x_1 \dots x_n$ $P(x_1 \dots x_n)$))" заменяются, соответственно, на $P(t)$ и на $P(t_1 \dots t_n)$. Если в последнем случае длины набора и связывающей приставки различаются, утверждение заменяется на константу "ложь". Уровень срабатывания приема равен 0.

Отождествление двух описателей "класс", отличающихся переобозначением связанных переменных

Если два выражения "класс(...)" получаются друг из друга лишь переобозначением связанных переменных, причем эти переменные в обоих случаях уже были переобозначены так, чтобы не совпадать со свододными переменными контекстов обоих выражений, то одно из них заменяется на другое. Предпочтение отдается тому выражению, которое идет раньше в лексикографическом порядке. Уровень срабатывания приема равен 0.

Замена на константу "ложь" условия принадлежности области определения отображения, если из контекста усматривается пустота этой области

Если в задаче усматривается утверждение "равно(класс($x_1 \dots x_n$ $P(x_1 \dots x_n)$)пусто)", расположенное в контексте выражения "отображение($x_1 \dots x_n$ $P(x_1 \dots x_n)$ $t(x_1 \dots x_n)$)", то последнее заменяется на "отображение($x_1 \dots x_n$ ложь $t(x_1 \dots x_n)$)". Уровень срабатывания 1.

Расшифровка равенства описателя "класс" пустому множеству

Утверждение "равно(класс($x_1 \dots x_n$ $P(x_1 \dots x_n)$)пусто)" заменяется на "не(существует($x_1 \dots x_n$ $P(x_1 \dots x_n)$))". Попытка применения данного приема выполняется после попытки применения предыдущего. Уровень срабатывания равен 1.

Ввод вспомогательного обозначения для описателя "класс"

Если в условии задачи на описание, не имеющей целей "редакция", "прямойответ", встречается выражение "класс(x $P(x)$)", не содержащее неизвестных, то для него вводится вспомогательное обозначение. Выбирается новая переменная y, и в список посылок заносится равенство "класс(x $P(x)$))= y". Оно снабжается комментарием "ориентацияравенства", так что далее будут применяться приемы, заменяющие все вхождения в задачу данного выражения "класс(...)" на y. Перед регистрацией указанного равенства предпринимается рассмотрение утверждения $\forall_x(x \in y \to P(x))$. Оно последовательно обрабатывается двумя вспомогательными задачами на описание - сначала для развертки "по определениям", затем - для обратной свертки. Те конъюнктивные члены результата, которые представляют собой элементарные утверждения либо простые импликации, регистрируются в посылках. Они могут позволить работать далее с обозначением y напрямую, без вводящего это обозначение равенства. Уровень срабатывания приема равен 5. Заметим, что необходимое для возможности срабатывания приема отсутствие цели "прямойответ" означает, скорее всего, что задача была создана для решения внешней задачи на доказательство. В

этой ситуации дополнительные посылки, сопровождающие y, могут позволить вывести необходимые для доказательства следствия.

10.16 Приемы символа "отображение"

Количество общих приемов символа "отображение" невелико - как реализованных на ЛОСе, так и на ГЕНОЛОГе.

Нормализация связывающей приставки

Прием переобозначает переменные связывающей приставки так, чтобы они отличались от свободных переменных утверждений контекста. Уровень срабатывания равен 0.

Определение значения в заданной точке

Выражения "значение(отображение($x\ P(x)\ t(x)$) A)" и "значение(отображение($x_1 \ldots x_n\ P(x_1 \ldots x_n)\ t(x_1 \ldots x_n)$) набор($A_1 \ldots A_n$))" заменяются, соответственно, на $t(A)$ и $t(A_1 \ldots A_n)$. Уровень срабатывания равен 0.

Область определения отображения

Выражение "область(отображение($x_1 \ldots x_n\ P(x_1 \ldots x_n)\ t(x_1 \ldots x_n)$))" заменяется на результат упрощения выражения "класс($x_1 \ldots x_n\ P(x_1 \ldots x_n)$)". Упрощение предпринимается с помощью вспомогательной задачи на преобразование, решаемой до максимального уровня 4. Уровень срабатывания равен 0.

Переобозначение связанных переменных для отождествления описателей "отображение"

Если в задаче усматриваются два различных выражения "отображение(...)", отличающихся друг от друга переобозначением связанных переменных, то создается эталонная версия C указанных выражений A, B, полученная из них переобозначением переменных связывающей приставки на новые переменные. Затем все вхождения в задачу выражений A, B заменяются на C. Уровень срабатывания равен 1.

Попытка выразить через числовые параметры числовй атом, используемый для определения значения функции

Пусть в посылке задачи на доказательство либо задачи на исследование, имеющей цель "известно", встречается выражение f(отображение($x_1 \ldots x_n\ P(x_1 \ldots x_n)\ t(x_1 \ldots x_n)$)), имеющее внутри $t(x_1 \ldots x_n)$ вхождение числового атома A, отличного от переменной и зависящего от $x_1, \ldots, x_n$. Такой атом представляет собой числовую характеристику нечислового объекта (масса, длина, мощность и т.п.). Пусть, далее, f - числовой функционал (знак конечного суммирования, интегрирования и т.п.). Тогда предпринимается попытка выразить A через числовые параметры задачи, быть может, используя также другие числовые атомы, специальным образом связанные с A.

Для этой цели вводится вспомогательная задача на описание Z', условиями которой служат равенство $A = x$ и утверждение "число(x)"; x - новая переменная, играющая роль неизвестной. К посылкам относятся все утверждения контекста рассматриваемого вхождения A. Задача имеет цель (известно ...), перечисляющую все числовые параметры указанного контекста (в том числе, возможно, некоторые переменные x_i связывающей приставки).

Если внутри числового атома A усматривается вхождение выражения "значение(g s)", где g - переменная, а s содержит все переменные $x_1, \ldots, x_n$, то предпринимается рассмотрение посылок текущей задачи Z, имеющих вхождения выражений вида "отображение($x_1 \ldots x_n$ $P(x_1 \ldots x_n)$ $q(x_1 \ldots x_n)$)". Если внутри $q(x_1 \ldots x_n)$ встречается отличный от A числовой атом B, причем B имеет подвыражение "значение(g s')", где s' содержит все переменные $x_1, \ldots, x_n$, то на время решения задачи Z' числовой атом B объявляется известным. Для этого вводится вспомогательная переменная y; список посылок задачи Z' пополняется равенством $B = y$, и y регистрируется в цели (известно ...).

Задача Z' решается до максимального уровня 5. Если на нее получен ответ, дающий выражение R числового атома A через числовые параметры, то в R восстанавливаются числовые атомы B, временно обозначенные вспомогательными переменными y (если такие переменные вообще имеются в R), и предпринимается замена в текущей посылке атома A на R. Уровень срабатывания приема равен 5.

Попытка определить значение функции в задаче на вычисление

Задачи на описание, имеющие цель "вычисление", предназначены для составления ЛОС-программ, вычисляющих значения неизвестных задачи по значениям ее известных параметров. Такая программа создается в два этапа: сначала условия задачи на описание преобразуются к виду, понятному компилятору ГЕНОЛОГа, а затем предпринимается обращение к компилятору. На первом этапе иногда оказывается достаточно использовать стандартные схемы вычислений для чисто технической переформулировки условий (представление интеграла в виде конечной суммы, переход от дифференциального уравнения к конечно-разностной схеме и т.п.). Иногда же приходится сначала решать обычную задачу на явное выражение неизвестных через известные, и лишь затем переходить к техническим преобразованиям. Такая ситуация складывается, например, при составлении программ, относящихся к физическим процессам, заданным на логическом языке. Типичным явлением здесь является определение траектории какого - либо параметра, т.е. таблицы значений, принимаемых им в последовательные моменты времени. В этом случае исходная задача на описание имеет условие вида "y = отображение(t $P(t)$ $F(t)$)"; y - неизвестная для траектории.

Ее решение сводится к выводу следствий в блоке анализа, где описываемый прием и обнаруживает содержащую выражение "отображение(t $P(t)$ $F(t)$)" посылку. Если выражение $F(t)$ содержит неизвестные блока анализа (например, задано через косвенные характеристики процессов - координаты, скорости, силы, пока еще не выраженные явно через числовые параметры), то создается вспомогательная задача на описание Z'. Ее условиями становятся утверждения $x = F(t)$, "число(x)", посылками - все утверждения из контекста $F(t)$. Неизвестной задачи служит новая переменная x; цель (известно ...) содержит все известные параметры блока анализа и переменную t. Если удается получить ответ на задачу Z', дающий явное выражение R для

ее неизвестной, то предпринимается замена в рассматриваемой посылке выражения $F(t)$ на R. Уровень срабатывания приема равен 2.

10.17 Приемы логических констант

Приемы логических констант обычно выполняют их исключение либо усматривают завершение процесса решения задачи.

Условие "истина" задачи на доказательство

Если условием задачи на доказательство служит логическая константа "истина", то выдается ответ "истина". Уровень срабатывания равен 0.

Условие "истина" задачи на описание

Если логическая константа "истина" служит условием задачи на описание, то рассматриваются два случая. При отсутствии прочих условий выдается ответ "истина" (не логический символ, как в случае задачи на доказательство, а однобуквенный терм). Иначе условие "истина" удаляется из списка условий, и решение продолжается. Если при этом остается единственное условие с неизвестными, вес его понижается до 0. Данная мера нужна, чтобы срабатывали приемы усмотрения ответа, которые в противном случае "не заметят" исчезновения (преобразования в константу "истина") неизвестных условий, ранее не вписывавшихся в требуемый вид ответа. Уровень срабатывания приема равен 0.

Посылка "истина"

Если логическая константа "истина" служит посылкой задачи, причем имеются и другие посылки, то она удаляется. Уровень срабатывания 0.

Консеквент "истина"

Если логическая константа "истина" является консеквентом кванторной импликации, то эта импликация заменяется на константу "истина". Уровень срабатывания 0.

Антецедент "истина"

Если логическая константа "истина" является антецедентом кванторной импликации, то она отбрасывается. При этом могут быть отброшены также частицы "если", "то". Уровень срабатывания 0.

Логическая константа "истина" под дизъюнкцией либо квантором существования

Если константа "истина" является операндом дизъюнкции либо квантора существования, то последние заменяются на эту константу. Уровень срабатывания 0.

Логическая константа "истина" под конъюнкцией либо эквивалентностью

Если константа "истина" является операндом конъюнкции либо эквивалентсности, то этот операнд отбрасывается. Уровень срабатывания 0.

Логическая константа "истина" под отрицанием

Отрицание константы "истина" заменяется на "ложь". Уровень срабатывания 0.

Логическая константа "истина" в условном выражении

Выражение "вариант(истина t_1 t_2)" заменяется на t_1. Уровень срабатывания 0.

Условие "ложь" задачи на доказательство

Если условием задачи на доказательство оказалась константа "ложь", то на уровне 0 выдается отказ. Однако, может так оказаться, что список посылок задачи на доказательство, имеющей условие "ложь", противоречив. В этом случае допустима выдача ответа "истина". Чтобы не спешить с выдачей отказа до того, как предприняты попытки установить противоречивость посылок, предсумотрен комментарий "ложь". При его наличии выдача отказа откладывается до уровня 5.

Условие "ложь" задачи на описание

Если константа "ложь" оказалась условием задачи на описание, то при наличии целей "полный" и "пример" выдается отказ. Иначе выдается ответ "ложь". Уровень срабатывания равен 0.

Посылка "ложь" задачи на доказательство

Если задача на доказательство имеет посылку "ложь", то на нее выдается ответ "истина". Уровень срабатывания 0.

Посылка "ложь" задачи на описание

Если задача на описание имеет посылку "ложь", то на нее выдается ответ "истина". Уровень срабатывания 0.

Посылка "ложь" задачи на преобразование

Если задача на преобразование имеет посылку "ложь", то на нее выдается ответ - однобуквенный терм "противоречие". Символ "противоречие" может возникать в решателе только указанным образом, и в особых случаях нужно учитывать возможность его появления как ответа задачи на преобразование. Уровень срабатывания равен 0.

Посылка "ложь" задачи на исследование

Если задача на исследование является блоком анализа внешней задачи на описание, то при появлении в ее посылках константы "ложь" предпринимаются регистрация этой константы в условиях задачи на описание и возвращение к сканированию последней. Если задача на исследование решалась с целью усмотрения противоречивости списка посылок, то при обнаружении в ней константы "ложь" просто происходит возвращение к внешней задаче. Уровень срабатывания равен 0.

Логическая константа "ложь" под отрицанием либо в антецеденте кванторной импликации

Если утверждение представляет собой отрицание константы "ложь" либо кванторную импликацию с антецедентом "ложь", то оно заменяется на константу "истина". Уровень срабатывания 0.

Логическая константа "ложь" под конъюнкцией либо под квантором существования

Если утверждение представляет собой конъюнкцию с операндом "ложь" либо квантор существования с подкванторным утверждением "ложь", то оно заменяется на константу "ложь". Уровень срабатывания 0.

Логическая константа "ложь" под дизъюнкцией

Если константа "ложь" оказывается операндом дизъюнкции, то она отбрасывается. Уровень срабатывания 0.

Логическая константа "ложь" в консеквенте кванторной импликации

Если логическая константа "ложь" является консеквентом кванторной импликации $\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\&\, A_n \to A_0)$, то она заменяется:

1. При $n = 0$ - на константу "ложь";
2. При $n > 0$ - на $\forall_{x_1 \ldots x_n}(A_1 \,\&\ldots\&\, A_{n-1} \to \neg A_n)$.

Прием срабатывает на уровне 0.

Логическая константа "ложь" под эквивалентностью

Утверждение "эквивалентно(A ложь)" заменяется на $\neg A$. Уровень срабатывания 0.

10.18 Приемы основных символов, связанных с функциями

Для символов "область", "значение", "подфункция", "существперем" на ЛОСе реализована сравнительно большая группа приемов. Напомним, что посредством "значение(f t)" обозначается значение функции f в точке t и что это выражение прорисовывается формульным редактором в виде $f(t)$.

Функция, для которой рассматривается лишь ее область определения

Если связанная переменная x квантора общности либо существования встречается только в выражениях "область(x)", за исключением, быть может, утверждения "функция(x)", расположенного в контексте указанных выражений, то утверждение "функция(x)" заменяется на "множество(x)", а все выражения "область(x)" заменяются на x. Уровень срабатывания приема равен 1.

Условие принадлежности области определения функции, заданной описателем "отображение"

Если в контексте утверждения "принадлежит(t область(f))" встречается равенство "f = отображение(x $P(x)$ $s(x)$)", то данное утверждение заменяется на $P(t)$. Прием применяется и в векторной ситуации - если связывающая приставка x имеет более одной переменной, а выражение t имеет вид "набор(...)". Уровень срабатывания 1.

Замена квантора по функции, определенной на конечном множестве, на квантор по группе переменных - значений функции на элементах этого множества

Пусть связанная переменная f квантора общности либо существования встречается в одном из следующих контекстов:

1. В выражении вида $f(i)$, где i - десятичное число;
2. В утверждении "слово(f)" - антецеденте кванторной импликации либо конъюнктивном члене под квантором существования;
3. В утверждении "равно(длинанабора(f)m)", где m - десятичное число.

Тогда находится список S всевозможных выражений $f(i)$, содержащих f, и выбирается список P новых переменных, которые будут использоваться вместо данных выражений. Затем предпринимается замена под квантором выражений списка S на соответствующие им переменные списка P, замена в кванторной приставке переменной f на переменные P, и отбрасывание утверждений "слово(f)", "равно(длинанабора(f)m)". Уровень срабатывания равен 1.

Определение значения многоместной функции, заданной описателем "отображение"

Выражение "значение(f набор($t_1 \dots t_n$))", в контексте которого встречается равенство "f = отображение($x_1 \dots x_n$ $P(x_1 \dots x_n)$ $g(x_1 \dots x_n)$)", заменяется на результат упрощения выражения $g(t_1 \dots t_n)$. Прием применяется только к условиям задач и в ряде специальных случаев блокируется (например, если указанное выражение является операндом описателя "отображение"). Уровень срабатывания равен 1. Заметим, что в случае одноместных функций аналогичные преобразования выполняются приемами, заданными на ГЕНОЛОГе.

Исключение символа "подфункция"

Выражение "подфункция(f набор($i_1 \dots i_k$) набор($t_1 \dots t_k$))" обозначает результат фиксации аргументов многоместной функции f, номера которых суть $i_1, \dots, i_k$, значениями $t_1, \dots, t_k$. Если f имеет вид "отображение($x_1 \dots x_n$ P g)", либо в контексте содержится равенство f такому выражению, то определяются результаты A, B подстановки выражений $t_1, \dots, t_k$ вместо $x_{i_1}, \dots, x_{i_k}$ в утверждение P и выражение g. Утверждение A упрощается с помощью вспомогательной задачи на преобразование, решаемой до уровня 4; выражение B обрабатывается нормализаторами общей стандартизации. После этого находится результат $y_1, \dots, y_{n-k}$ исключения из связывающей приставки $x_1, \dots, x_n$ переменных с номерами $i_1, \dots, i_k$, и выражение "подфункция(...)" заменяется на "отображение($y_1 \dots y_{n-k}$ A B)". Прием применяется на уровне 0.

Кванторная расшифровка условия существенности переменной

Утверждение "существперем(i f)" означает, что переменная x_i функции $f(x_1 \dots x_n)$ является существенной, т.е. что найдутся такие значения остальных переменных, при подстановке которых функция не обращается в константу. Если данное утверждение является условием задачи на описание, причем f имеет вид "отображение($x_1 \dots x_n$ P F)", то оно заменяется на утверждение существования двух различных значений аргумента x_i (с одинаковыми значениями прочих переменных), при которых значения функции различаются. Уровень срабатывания приема равен 4. Особо рассматривается случай функций алгебры логики, где кванторное утверждение приобретает несколько более простой вид.

Тождественная замена с помощью кванторного тождества, позволяющая исключить символ "значение"

Для переменных f, имеющих своими значениями функции, в посылках задачи часто указываются кванторные тождества $\forall_{x_1 \dots x_n}(A_1 \ \& \ \dots \ \& \ A_n \to B = C$, позволяющие определить значение C выражения B, содержащего подтерм вида "значение(f r)". Имеется прием, предпринимающий попытку воспользоваться таким тождеством для исключения подтермов указанного вида. Он активизируется в задаче на исследование при обнаружении вхождения подтерма "значение(f r)" в консеквент кванторного тождества. Проверяется, что кванторная приставка пересекается с параметрами выражения r и что либо терм C не содержит символа "значение", либо он не содержит неизвестных, в то время как терм B содержит неизвестные. Проверяется, что выражение B не содержит связанных переменных. Затем просматриваются такие посылки текущей задачи на исследование, в которых можно выделить подтерм "значение(f R)", получаемый подстановкой в "значение(f r)" некоторых термов $t_1, \dots, t_n$ вместо переменных $x_1, \dots, x_n$. Проверяется, что эта подстановка переводит антецеденты кванторной импликации в очевидные утверждения, после чего подтерм "значение(f R)" заменяется в посылке на результат применения подстановки к C. Уровень срабатывания приема равен 4. Блокируется преобразование посылок, специально введенных для указания связей значения функции f с прочими объектами и помеченных комментарием "узелвывода".

Применение кванторного тождества, помеченного комментарием "блокзамен"

Чтобы кванторное тождество вида $\forall_{x_1 \ldots x_n}(A_1 \ \& \ \ldots \ \& \ A_n \to f(t) = r$, расположенное в посылках текущей задачи, использовалось для повсеместного исключения выражений вида $f(\ldots)$, оно может быть помечено комментарием (блокзамен f). Исключение выполняется двумя приемами, аналогичными приему предыдущего пункта. Первый из них инициализирует использование тождества. Как только тождество возникает, прием предпринимает просмотр посылок и применяет тождество к их подвыражениям $f(\ldots)$. Второй прием необходим, чтобы тождество могло быть применено к тем выражениям $f(\ldots)$, которые появляются в задаче после применения первого приема. Он активизируется при усмотрении выражения вида "значение(f R)"; предпринимает поиск тождества указанного выше вида, помеченного комментарием (блокзамен f), и применяет его к данному выражению. Уровень срабатывания обоих приемов равен 2.

Попытка получить рекуррентную формулу для числового атома, содержащего подвыражение с заголовком "значение"

Если посылка задачи на исследование имеет числовой атом C, содержащий подвыражение "значение(f t)", где t неконстантное, то предпринимается попытка получить для C рекуррентную формулу. Для этого находится кванторная посылка $\forall_x(A_1 \ \& \ \ldots \ \& \ A_n \to A_0)$, консеквент которой имеет подвыражение "значение(f r)". Проверяется, что антецеденты этой посылки определяют либо конечный отрезок целочисленных значений переменной x, либо множество всех ее натуральных значений, не меньших заданного. В консеквенте A_0 находится еще одно подвыражение "значение(f R)" и проверяется, что r - числовое выражение, большее на единицу выражения R. Если такая кванторная посылка обнаружена, то определяются результаты C_1, C_2 замены в терме C подвыражения "значение(f t)" на выражения "значение(f R)", "значение(f r)" соответственно. Выбираются новые переменные y, z, и создается вспомогательная задача на описание Z'. Ее посылками служат все посылки текущей задачи, пополненные антецедентами $A_1, \ldots A_n$ и утверждениями "число(y)", $y = C_1$. Условия суть утверждения "число(z)", $z = C_2$. Задача имеет цели "полный", "явное", "прямойответ", "рекурсия", "упростить". Ее единственной неизвестной служит z; цель (известно ...) перечисляет все известные параметры текущей задачи и переменную y. Если на задачу Z' получен ответ P, то находится результат P' подстановки в него вместо переменных y, z выражений C_1, C_2. После этого в список посылок заносится утверждение $\forall_x(A_1 \ \& \ \ldots \ \& \ A_n \to P')$, дающее рекуррентную зависимость для числового атома C. Уровень срабатывания приема равен 3. От задачи на исследование дополнительно требуется наличие цели "известно".

Понижение весов кванторных импликаций, используемых для вывода утверждений с символом "значение"

Если задача на преобразование имеет цель (значение x), то при ее решении стимулируется вывод следствий в посылках, содержащих выражения вида "значение(f x)". Имеется прием, активизируемый при обнаружении в условии такой задачи выражения вида "значение(A x)" и понижающий до уровня 2 веса всех кванторных импли-

каций в посылках, содержащих подтерм "значение(A ...)". Этот прием срабатывает на уровне 3.

Вывод следствий, содержащих заданный терм "значение(f t)"

Начиная с данного пункта, будет рассмотрена серия приемов, относящихся к задачам на описание, имеющим цель "длялюбого". Как уже говорилось выше, такая задача Z' вводится для анализа кванторного условия F внешней задачи на описание Z. В посылки ее заносятся все антецеденты импликации F, а конъюнктивные члены консеквента становятся условием. Задача не имеет неизвестных, но имеет цель (независит ...), перечисляющую переменные кванторной приставки F. При ее решении можно пополнять список посылок дополнительными утверждениями существования, не являющимися следствиями исходных. Все такие дополнения регистрируются в комментарии (блокпосылок A_1 A_2). Здесь A_1 - набор троек (X_1 X_2 P), соответствующих новым посылкам "существует(X B)". Связывающая приставка X разбита на две компоненты X_1, X_2, причем переменные первой из них считаются не зависящими от переменных кванторной приставки импликации F, а переменные второй - могут зависеть. Данное разбиение оставляется на усмотрение того приема, который добавляет посылку. P - набор конъюнктивных членов утверждения B. A_2 - набор конъюнктивных членов результата упрощения утверждения, выражающего, что для любых удовлетворяющих антецедентам импликации F значений переменных ее кванторной приставки существуют значения переменных из списков X_2, при которых истинна конъюнкция всех утверждений P. Различные комментарии (блокпосылок ...) относятся к несвязанным между собой группам переменных X_i.

Если условие задачи на описание, имеющей цель "длялюбого", содержит подтерм "значение(f t)", у которого выражение t зависит от переменных цели (независит ...), то необходимо попытаться исключить данный подтерм из условия. Для этого нужно вывести какие-то следствия, содержащие подтерм. В посылках задачи ищется кванторная импликация K, у которой консеквент имеет подвыражение "значение(f r)". Определяется список S_1 всех переменных кванторной приставки импликации K, входящих в r, и проверяется, что t является результатом подстановки в r некоторых термов T_1 вместо переменных S_1. Находится оставшаяся часть S_2 кванторной приставки импликации K. Для ее переобозначения выбирается список S_3 не используемых в задаче переменных. Затем находятся: список D результатов подстановки T_1, S_3 вместо S_1, S_2 в антецеденты импликации K, а также результат E применения данной подстановки к консеквенту K. Проверяется, что E не имеет подтермов "значение(...)", у которых второй операнд содержит переменные списка S_3. Таким образом, оказывается выполнена подготовка к регистрации в посылках нового утверждения E, содержащего подтерм "значение(f t)". Однако, для его вывода требуется истинность всех утверждений списка D, на которые навешен квантор существования по переменным S_3. Эти утверждения регистрируются в указанном выше комментарии (блокпосылок ...), и лишь затем к посылкам задачи присоединяются утверждение E и все утверждения D. Прием срабатывает на уровне 2.

Вывод промежуточных импликаций в задаче с целью "длялюбого"

В предыдущем приеме перед выводом следствия E проверялось, что в нем отсутствуют термы "значение(g s)", у которых s содержит переменные списка S_3. Если такие

термы в E имеются, то прием несколько корректируется - вместо E выводится кванторная импликация, у которой E является консеквентом. Кванторную приставку импликации образуют все переменные списка S_3, входящие в указанные выражения s. Антецедентами служат все утверждения D, зависящие от таких переменных. Перед выводом проверяется, что прочие утверждения списка D являются следствиями списка посылок текущей задачи. Выведенная импликация снабжается комментарием "чисткапосылок". Так как в данном случае выводимое утверждение тоже является следствием посылок, то никакой регистрации его в комментариях (блокпосылок ...) не выполняется. Кванторная импликация, выведенная приемом, называется промежуточной. Бесквантерные следствия из нее выводит прием, описываемый в следующем пункте. Заметим, что применение к промежуточным импликациям приема из предыдущего пункта блокируется. Примеры применения приема можно найти в простейших задачах на доказательство из раздела задачника "Математический анализ". Уровень срабатывания его, как и предыдущего приема, равен 2.

Извлечение следствий из промежуточных кванторных импликаций в задаче с целью "длялюбого"

Промежуточная кванторная импликация K, введенная предыдущим приемом, содержит своем консеквенте подтерм "значение(g s)". Чтобы как-то связать это значение с другой информацией о функции g, используется еще одна кванторная импликация K' из посылок текущей задачи, содержащая g. Предварительно проверяется, что все переменные кванторной приставки K содержатся в s и что утверждение K не имеет других подтермов вида "значение(...)", второй аргумент которых пересекается с кванторной приставкой K. Проверяется также, что ни одна из импликаций K, K' не была использована при выводе другой. Перед выводом следствий предпринимается такое переобозначение связанных переменных обоих импликаций, чтобы они отличались друг от друга и от переменных прочих утверждений задачи. В консеквенте импликации K' находится подтерм "значение(g u)". Составляется список H всех переменных кванторной приставки K', встречающихся в u, и проверяется отсутствие в K' подтермов "значение(...)", содержащих не вошедшие в H переменные кванторной приставки. Далее составляется список H', полученный добавлением к H кванторной приставки импликации K. Находятся такие термы T, подстановка которых вместо H' унифицирует выражения u и s. Эта подстановка определяет значения всех переменных кванторной приставки K и части переменных кванторной прставки K'. Находятся результаты C_1, C_2 ее применения к консеквентам импликаций K, K', а также список L результатов ее применения к антецедентам обеих импликаций. Находится результат U применения данной подстановки к u. Составляется список V переменных связывающих приставок импликаций K, K', встречающихся в утверждениях из L. Он подразбивается на два подсписка V_1, V_2, первый из которых образован всеми переменными, не входящими в U. Чтобы получить следствия C_1, C_2 из K, K', нужно ввести дополнительную посылку - утверждение о существовании значений переменных списка V, при которых истинны антецеденты L. Это утверждение регистрируется в комментарии (блокпосылок ...), причем переменные списка V_1 считаются при регистрации не зависящими от переменных кванторной приставки кванторного условия F внешней задачи, а переменные V_2 - зависящими. Последние переменные присоединяются также к цели (независит ...). Завершает действия приема присоединение к посылкам утверждений C_1, C_2, а также всех утверждений списка L. Уровень срабатывания приема равен 2.

Использование кванторных импликаций для разбора случаев в задаче с целью "длялюбого"

Если в условии задачи на описание, имеющей цель "длялюбого", встречается выражение "значение(f t)", у которого t содержит переменные, выделенные целью (независит ...), то предпринимается поиск в посылках кванторной импликации K, имеющей в своем консеквенте выражение "значение(f r)". Проверяется, что эта кванторная импликация не была использована для получения следствия с выражением "значение(f t)" и не является промежуточной импликацией (см. выше). Составляется список S_1 всех переменных кванторной приставки K, входящих в r, и определяются такие термы T, подстановка которых в r вместо S_1 дает t. Находится список S_2 остальных переменных кванторной приставки K, и для их переобозначения выбираются не используемые в задаче переменные S_3. Далее определяются список D результатов подстановки T, S_3 вместо S_1, S_2 в антецеденты импликации K, а также результат E применения этой подстановки к консеквенту K. Проверяется, что E не содержит подвыражений "значение(...)", второй операнд которых пересекается с S_3. Список D разбивается на два подсписка D_1, D_2, к первому из которых относятся все утверждения, имеющие выделенные целью (независит ...) переменные. Проверяется, что D_1 непуст и что все переменные списка S_3 содержатся в D_2. Далее в комментарии (блокпосылок ...) регистрируется утверждение о существовании таких значений переменных списка S_3, при которых истинны все утверждения из D_2. Однако, этого утверждения еще не достаточно для вывода следствия E, так как остается нереализованным список D_1. Поэтому находится конъюнкция H всех утверждений списка D_1, и в список посылок текущей задачи заносится дизъюнкция $(E \,\&\, H) \vee \neg H$. Она помечается комментарием "разборслучаев". Заметим, что задача имеет цель (независит ...), а альтернативы новой посылки зависят от переменных данной цели. Чтобы эта зависимость не перешла в ответ задачи, он будет формироваться как конъюнкция ответов, найденных для подслучаев (см. выше прием разбора случаев по дизъюнктивной посылке задачи на описание). Такая схема рассуждений является типичной, например, при получении оценок. Сначала неравенства для оцениваемой величины x находятся в каждом подслучае, а затем конъюнкция этих неравенств дает оценку, верную во всех ситуациях. Прием срабатывает на уровне 3.

Попытка использовать константную функцию

Если в условии задачи на описание, имеющей цель "длялюбого", встречается выражение "значение(f t)", у которого t содержит переменные, выделенные цель (известно ...), а f есть неизвестная внешней задачи на описание, то предпринимается поптыка подобрать в качестве f константную функцию. Для этого сначала упрощается выражение "область(f)". Если результат упрощения M не содержит переменной f, то создается выражение F вида "отображение(x $x \in M$ a)". Здесь x, a - новые переменные. Находится список S свободных переменных равенства $f = F$, и в комментарии (блокпосылок ...) регистрируется утверждение о существовании таких значений переменных S, при которых это равенство истинно. Затем равенство $f = F$ заносится в список посылок, и во всех условиях задачи выражения "значение(f X)" заменяются на a. Прием срабатывает на уровне 5. Разумеется, для попытки подбора константной функции нужны дополнительные веские основания; нужно обеспечить возможность отката при неудаче и т.п. Однако, на рассматривавшихся задачах достаточно было и предлагаемой версии. Данный и другие приемы, относящиеся к задачам

с целью "длялюбого", проверены пока на очень небольшом обучающем материале. Они имеют характер грубых предварительных версий и требуют существенной последующей доработки. Тем не менее, они все же иллюстрируют общие принципы развития аппарата, связанного с кванторными условиями задач на описание.

Использование кванторной импликации для получения оценки выражения "значение(f t)"

Если посылка K задачи на описание, имеющей цель "длялюбого", представляет собой кванторную импликацию с неравенством для выражения "значение(f x)" в консеквенте, то предпринимается попытка вывести с ее помощью неравенство для встречающегося в условии P выражения "значение(f t)". Проверяется, что t зависит от переменных, упоминаемых в цели (независит ...) текущей задачи и что x - переменная кванторной приставки K. Далее проверяется, что условие P само имеет вид неравенства, причем направление неравенства для "значение(f t)", получаемого из K, позволяет воспользоваться им для вывода P "по монотонности" из нового условия Q, полученного заменой выражения "значение(f t)" на предлагаемую неравенством оценку этого выражения. Рассматривается кванторная приставка X импликации K. Если она состоит из единственной переменной x, то для подстановки вместо X берется одноэлементный список, состоящий из терма t. В противном случае проверяется, что X двухэлементна, и для подстановки вместо x снова выбирается t, а для подстановки вместо другой переменной - какое-либо выражение r, не имеющее общих переменных с целью (независит ...) и такое, что "значение(f r)" встречается в условиях задачи. Находятся результаты применения указанной подстановки к антецедентам импликации K и проверяется, что они вытекают из посылок задачи. Затем определяется результат применения подстановки к консеквенту импликации, который и регистрируется в посылках задачи. Вес условия с выражением "значение(f t)" понижается до 3. Уровень срабатывания приема равен 4.

Подбор примера с заменой функциональной неизвестной на обычную

Пусть в задаче на описание с целью "пример" имеется такая неизвестная f, которая встречается в условиях "функция(f)", "равно(область(f)A)", а кроме этого - только внутри выражений "значение(f x)", расположенных в консеквентах кванторных импликаций, имеющих антецедент "принадлежит(x A)"; x - переменная кванторной приставки импликации, не встречающаяся в прочих антецедентах. Такие импликации задают некоторое общее условие на произвольное значение функции f, так что всегда можно в качестве f брать константную функцию, определенную на A и имеющую такое значение a, которое удовлетворяет данному условию. Прием, усматривающий описанную ситуацию, составляет список S утверждений, полученных из содержащих f кванторных импликаций удалением антецедента "принадлежит(x A)", исключением из кванторной приставки переменной x и заменой "значение(f x)" на "значение(f y)". Если не только f, но и некоторые другие функциональные неизвестные g удовлетворяют перечисленным выше ограничениям, причем их вхождения "значение(g x)" расположены в тех же кванторных импликациях и имеют те же самые аргументы x, то они вместе с f образуют список X преобразуемых неизвестных. Утверждения списка S преобразуются путем замены выражений "значение(g y)" для g из X на переменные g. После этого создается вспомогательная задача Z' на описание, списком условий которой служит S. Посылки ее составлены из посылок текущей задачи,

всех не содержащих переменных X условий текущей задачи, а также из утверждения $y \in A$. Неизвестные суть переменные списка X. Если после решения задачи Z' получаются значения $t_1, \ldots, t_n$ для составляющих список X неизвестных $x_1, \ldots, x_n$, то в текущей задаче отбрасываются все условия, содержащие эти неизвестные, а вместо них регистрируются утверждения "равно(x_i отображение(y принадлежит(y A) t_i))" и сопровождающие конъюнктивные члены ответа задачи Z'. Уровень срабатывания приема равен 2.

Неизвестная последовательность, входящая в условия с одним и тем же аргументом

Если в задаче на описание с целью "пример" имеется неизвестная f, входящая в условие "последовательность(f A)", а кроме этого встречающаяся только внутри вхождений одного и того же выражения "значение(f t)", не связанного внешними кванторами и описателями, то применяется прием, аналогичный предыдущему - функциональная неизвестная f преобразуется в обычную. Для этого создается новая задача на описание Z', имеющая своими условиями утверждение $f \in A$, а также утверждения, полученные из отличных от "последовательность(f A)" условий текущей задачи заменой вхождений выражения "значение(f t)" на f. Посылки и цели задачи Z' - те же, что у текущей задачи. Если на нее получен отличный от символа "отказ" результат R, определяющий значение s неизвестной f, то выбирается новая переменная y, и значение s заменяется в R на "отображение(y натуральное(y) s)". Если текущая задача не имела комментария (контекст ...), сохраняющего значения внешних неизвестных, выраженных через оставшиеся неизвестные, то сразу выдается ответ R. Иначе условия текущей задачи заменяются на конъюнктивные члены утверждения R, и предпринимается обращение к процедуре "редакторответа", обеспечивающей надлежащий учет комментария (контекст ...). Уровень срабатывания приема равен 2.

Попытка подобрать значение неизвестной из сопоставления аргументов одной и той же функции

Если задача на описание имеет цели "полный", "пример", причем ее единственная неизвестная x встречается в подвыражении условия "значение(f x)", а посылки имеют подвыражение "значение(f t)", то предпринимается попытка проверить, не подойдет ли для x значение t. Проверки истинности результатов подстановки t вместо x в условия задачи предпринимаются с достаточно сильным ограничением на трудоемкость. Если они успешно реализованы, выдается ответ $x = t$. Уровень срабатывания приема равен 4.

10.19 Примеры и упражнения

Большинство из приведенных выше приемов выполняют преобразования, целесообразность которых очевидна. Однако, в ряде случаев она может показаться сомнительной, а то и вовсе непонятной. В этих ситуациях полезно обратиться к задачнику системы, найти те задачи, в которых прием срабатывает, и с помощью отладчика ЛОСа проанализировать его действия подробнее. Иногда это может прояснить смысл

преобразований. В тех редких случаях, когда и рассмотрение примера оставляет сомнения относительно целесообразности приема, следует относиться к нему как к временной "заглушке", созданной по немногим, чаще всего - по единственному примеру. Здесь можно смело браться за доработку приема или даже найти какую-то совсем иную схему решения задачи, позаботившись в первую очередь о достаточно богатом обучающем материале для анализа аналогичных ситуаций. Этот материал должен предоставить информацию о "скрытых" особенностях задачи, которые необходимо учитывать перед принятием решения о применении приема, а также обеспечить тестирование его фактического поведения. Если обновленный или вновь созданный прием имеет сравнительно общий характер и может срабатывать в различных предметных областях, то следует проявлять особую осторожность при занесении его в базу приемов, выполняя прогонку решателя по соответствующим разделам или даже по всему задачнику.

10.19.1 Прием разбора случаев по дизъюнктивному условию задачи на описание

Чтобы проиллюстрировать технику применения отладчика для анализа действий приемов, рассмотрим несколько простых примеров. В качестве первого примера выберем часто встречающийся прием разбора случаев по дизъюнктивному условию задачи на описание. Он находится в подразделе "Приемы решателя" - "Общие приемы" - "Дизъюнкция" - "Разбор случаев по дизъюнктивному условию задачи на описание".

Если задача, в которой срабатывает прием, неизвестна, то для отслеживания моментов его применения удобнее всего вставить в программу приема контрольную точку - оператор "трассировка(стоп 0)". При выходе на этот оператор программа будет останавливаться и включать отладчик ЛОСа, который и позволит рассмотреть контекст срабатывания. Если прием применяется крайне редко, то для поиска задач, где он срабатывает, удобно применять режим "прогонки" решателя по задачнику, предварительно введя указанную контрольную точку редактором ЛОСа.

Если задача, в которой прием срабатывает, известна, или такие задачи идут в рассматриваемом разделе задачника достаточно часто, то можно не пользоваться редактором ЛОСа, а запускать решение задачи через точку оглавления программ, которая соответствует рассматриваемому приему либо некоторому этапу его работы. В нашем случае лучше воспользоваться именно таким подходом. Разбор случаев часто встречается, например, при решении тригонометрических уравнений. Войдем в раздел задачника "Элементарная алгебра" - "Решение уравнений" - "Тригонометрические уравнения" - "Уравнения с неизвестной в знаменателе - 1" и выберем первую же задачу данного раздела. Чтобы перейти в оглавление программ, нажимаем "л", затем входим в указанный выше подраздел оглавления, соответствующий нашему приему. В этом подразделе выделено несколько концевых пунктов, и можно в качестве точки прерывания выбрать любой из них. Наиболее информативным, видимо, является пункт 4 - "Рассмотрение подслучая для дизъюнктивного условия". Он позволяет рассмотреть вспомогательную задачу, соответствующую текущему подслучаю, до входа в ее решение. При необходимости можно будет войти в трассировку подслучая, либо пропустить ее и сразу посмотреть ответ. Поэтому клавишами "курсор вниз - вверх" выделяем данный пункт. Для запуска решения задачи с прерыванием на нем нажимаем "курсор вправо". Начинается решение задачи, и почти сразу возникает

кадр отладчика ЛОСа. В нем выделен малиновым цветом текущий (еще не выполненный) оператор "уровеньобращения(x7)". Оператору предшествует контрольная точка "прием(3 3)", соответствующая выбранному концевому пункту оглавления. Чтобы на экране прорисовать весь текст фрагмента программы, нажимаем "ф". Тогда становится виден оператор "равно(x18 ответзадачи(x17))", обращающийся к рассмотрению текущего подслучая. Для анализа текущего контекста можно выполнить следующие действия:

1. Посмотреть на дизъюнкцию, по которой проводится разбор случаев. Ее текущее вхождение является значением программной переменной x2, так что достаточно нажать "x" (кир.), "2", "Enter". Появляется формула $\sin x + \cos x = 0 \ \vee \ \cos(2x) = -1/2$.

2. Посмотреть всю текущую задачу. Для этого нажимается клавиша "з". На экране появляется текст "Найти x", под которым расположены условие $\neg(\cos x = 0)$ и приведенная выше дизъюнкция. Чтобы вернуться в кадр отладчика ЛОСа, нажимается клавиша "ф". Не следует в данной ситуации нажимать Esc, так как это приведет к обрыву трассировки и возвращению к просмотру исходного текста задачи.

3. Посмотреть текущий рассматриваемый подслучай. Для этого, используя клавиши "Home", "End", перемещаемся вдоль цепочки фрагментов текущей программы, предшествующих текущему фрагменту. Нажатие клавиши "Home" переводит в направлении к началу программы, нажатие "End" - возвращает на один шаг в направлении от начала программы к текущему фрагменту. В одном из фрагментов данной цепочки замечаем оператор "равно(x14 набороперандов(x2))", который создает список x14 дизъюнктивных членов рассматриваемой дизъюнкции. Двигаясь от него по направлению к текущему фрагменту (клавиша "End"), обнаруживаем в следующем фрагменте оператор "позиция(x15 x14)". Очевидно, он и выделяет текущее рассматриваемое вхождение списка x14. За ним расположен оператор "равно(x16 буква(x15))", присваивающий текущий дизъюнктивный член переменной x16. Можно посмотреть значение этой переменной - $\sin x + \cos x = 0$.

4. Посмотреть задачу, созданную для обработки подслучая. Оператор "равно(x18 ответзадачи(x17))" подсказывает, что данная задача присвоена переменной x17. Так как задача представляет собой достаточно сложную структуру данных, удобно использовать сквозной ее просмотр. Поэтому нажимаем клавиши "К" (кир., заглавная); "1"; "7"; "Enter". Возникает список пронумерованных объектов - разрядов набора, представляющего задачу. Сначала идет тип задачи "описать", затем - список посылок, список весов посылок, и т.д. вплоть до 9-го элемента - блока анализа (пока отсутствующего). Выделяем клавишами "курсор вверх - вниз" для просмотра пятый элемент - список условий, после чего нажимаем "курсор вправо". Появляется список из трех утверждений $\neg(\cos x = 0)$, x — число, $\sin x + \cos x = 0$, в котором последнее утверждение - дизъюнктивный член, замещающий дизъюнкцию внешней задачи. Для возвращения в кадр отладчика ЛОСа нажимаем несколько раз "End" либо "Esc". Лучше избегать нажатий последней клавиши, так как лишнее ее нажатие оборвет трассировку.

Используя средства отладчика, можно посмотреть и другие сведения о текущем контексте - например, комментарии к вспомогательной задаче, ее цели и т.п. Продолжим

трассировку работы приема, посмотрев результат решения задачи x17. Для этого нужно переустановить режим трассировки на пооператорные шаги в текущем кадре - нажать клавишу "2". Если этого не сделать, а продолжить трассировку нажатиями клавиши "Enter", то отладчик будет отслеживать минимальные шаги работы интерпретатора и переходить в каждую реализуемую подпрограмму. После указанной смены режима нажимаем "Enter" дважды - сначала для прохождения через оператор "уровеньобращение(x7)", а затем - через оператор "равно(x18 ответзадачи(x17))". Просматриваем значение переменной x18 - найденный для подслучая ответ $\exists_n(x = (4n{+}3)\pi/4 \;\&\; n{-}$целое). Продолжаем трассировку по операторам программы, нажимая на клавишу "Enter". Через несколько шагов инициируется пустым словом накопитель x19. Как видно из следующего оператора, обращающегося к комментарию (выводимо ...), в x19 передается список всех использованных при решении задачи x17 посылок. Далее содержимое x19 присоединяется к накопителю x6, где создается общий список использованных приемом посылок. Следующий оператор - "замена(8 суффикс(x8 наборчленов(и x18)))" - заносит в накопитель x8 набор конъюнктивных членов ответа задачи x17. Таким образом, по окончании рассмотрения всех подслучаев длина набора x8 будет равна числу подслучаев, а его элементы будут определять найденные в подслучаях ответы. Еще одно нажатие клавиши "Enter" вызывает откат к оператору "позиция(x15 x14)", выбирающему следующий дизъюнктивный член. Чтобы сразу перейти от этой точки к моменту входа в решение вспомогательной задачи для очередного подслучая, нажимаем "Ctr-г", переводящее в оглавление программ. Здесь снова выбираем пункт "Рассмотрение подслучая для дизъюнктивного условия" и нажимаем "курсор вправо".

По окончании решения новой версии задачи x17 получаем ответ $\exists_n((x = (3n{+}1)\pi/3 \vee x = (3n{+}2)\pi/3) \;\&\; n{-}$целое). Его вид подсказывает, что при решении задачи имел место еще один разбор случаев по дизъюнктивному условию. Чтобы посмотреть момент этого разбора случаев, можно было сразу же после выхода на контрольную точку "прием(3 3)", предшествующую обращению к задаче x17, выделить оператор "уровеньобращения(x7)" и нажать "Ctr-Enter". Такие действия привели бы к останову отладчика ЛОСа на выделенном операторе безотносительно к тому кадру, в котором была создана установка на прерывание. В нашей задаче это означало бы остановку при попытке нового разбора случаев внутри текущего подслучая. Напомним, что для выделения оператора в отладчике ЛОСа нужно нажать клавишу "курсор вниз", и далее использовать клавиши "курсор вправо - влево". Чтобы отменить выделение, нажимается "курсор вверх". Если после выделения оператора "уровеньобращения(x7)" была бы нажата клавиша "Enter", то остановка произошла бы только при обращении к данному оператору в том же самом кадре, т.е. для очередного подслучая текущей дизъюнкции.

Чтобы посмотреть действия приема по завершении рассмотрения подслучаев, выходим через пункт "Завершающая обработка ответов для подслучаев" оглавления программ на контрольную точку "прием(3 5)". Нажимая "2" для перехода в пооператорную трассировку, каждый очередной шаг выполняем нажатием клавиши "Enter". Доходим до оператора, присваивающего переменной x12 пересечение всех "известных" утверждений, входящих в ответы подслучаев. Далее переменной x13 присваивается результат занесения в конец списка x12 дизъюнкции конъюнкций оставшихся фрагментов ответов, а переменной x14 - конъюнкция утверждений списка x13. Она является результатом разбора случаев. Оператор "изменение(окрестность(x1 4)набор(x14))" заменяет весь список условий текущей задачи на одноэлементный спи-

сок, состоящий из x14. Однако, в качестве ответа выдается не x14, а результат x15 упрощения его пакетным нормализатором "нормили". Этот нормализатор, в основном, реализован на ГЕНОЛОГе, и будет описан в последующих разделах. Продолжая трассировку, выполняем обращение к нормализатору и сравниваем утверждения x14, x15. Видно, что нормализатор объединил два параметрических описания в одно и вынес за скобку общее условие на параметр (n – целое).

10.19.2 Прием редактирования параметрического описания

Следующий пример - прием редактирования параметрического описания. Он применяется к имеющему вид параметрического описания $\exists_{x_1 \ldots x_n} A$ условию задачи на описание, не разрешенному явно относительно неизвестной либо такому, что на определяемые в нем неизвестные существуют дополнительные внешние условия. Прием может быть найден в разделе "Приемы решателя" - "Общие приемы" - "Квантор существования" - "Обращение к задаче на редактирование параметрического описания". Контрольная точка выхода на прием через оглавление программ расположена в начале программы, и не каждое попадание на нее будет означать срабатывание приема. Чтобы найти задачи, в которых прием срабатывает, нужно ввести в его программу контрольную точку - оператор "трассировка(стоп 0)". Для этого выходим на программу приема через оглавление программ и нажимаем "курсор вниз". Переходим к следующему фрагменту, в начале которого расположен оператор "равно(x13 ответзадачи(x12))". Он осуществляет обращение к вспомогательной задаче на редактирование параметрического описания, так что контрольную точку удобно разместить непосредственно перед ним.

После того, как контрольная точка, вызывающая выход в отладчик ЛОСа при попытке применить прием, создана, можно выбрать какой-либо раздел задачника и запустить цикл решения задач. Учитывая то, что параметрические описания часто возникают в тригонометрических уравнениях и неравенствах, выберем подраздел задачника "Элементарная алгебра" - "Решение уравнений" - "Тригонометрические уравнения" - "Уравнения с неизвестной в знаменателе - 1" и запустим цикл решения нажатием клавиши "Ctr-з". Уже на третьей задаче будет выявлено срабатывание приема. Однако, оно относится к сравнительно простому случаю, и для дальнейшего перейдем сразу к 22-й задаче подраздела, где ситуация несколько более интересная. Чтобы перейти к данной задаче после появления кадра отладчика ЛОСа в цикле решения, нажимаем клавишу "Ctr-з". Лишь после этого нажатие "Esc" оборвет цикл и выведет в главное меню, откуда еще раз нужно будет зайти в задачник и выбрать указанную задачу. Она имеет следующее условие:

$$(\sec x)^2 - (\cos x + \sin x \operatorname{tg} \frac{x}{2}) = \frac{\sin(x - \frac{\pi}{6}) + \cos(\frac{\pi}{3} - x)}{\cos x}.$$

Запускаем решение задачи клавишей "о" и попадаем в кадр отладчика ЛОСа. Просматриваем текущее параметрическое описание, на котором сработал прием - его вхождение является значением переменной x2: $\exists_n (x = n\pi \ \& \ n$ – целое). Просматриваем задачу x12 на редактирование данного параметрического описания. Так как ее решение еще не начато, используем сквозной просмотр набора x12, нажимая "К12". Пятый элемент этого набора - список условий; он состоит из утверждений $x = \pi n$, n – целое, $\neg(\cos \frac{x}{2} = 0)$, $\neg(\cos x = 0)$, x – число. Таким образом, смысл редактирования заключается в подстановке значения неизвестной x в сопровождающие условия

и упрощении результатов. Заметим, что задача x12 имеет две неизвестные - x и n, причем неизвестная n объявлена несущественной. Для входа в трассировку по шагам решения задачи x12 нажимаем "Enter". Появляется первый оператор программы символа "описать". Чтобы инициировать указанную трассировку задачи, нажимаем клавишу пробела. Теперь каждое новое нажатие "Enter" будет приводить к очередному шагу решения. Начинаем трассировку. На первом шаге предпринимается исключение неизвестной x, выраженной через n. Возникает вспомогательная задача, имеющая условия "n – целое, $\neg(\cos\frac{\pi n}{2} = 0)$, $\neg(\cos(\pi n) = 0)$, πn – число" и неизвестную n, уже существенную. Следующее нажатие "Enter" - условие $\neg(\cos(\pi n) = 0)$ заменяется на константу "истина", которая отбрасывается приемом, не выводимым на уровень данной трассировки. Далее условие $\neg(\cos\frac{\pi n}{2} = 0)$ заменяется на

$$\neg(((-1)^{\frac{n}{2}} \text{ при } n - \text{even}, \text{ иначе } 0) = 0).$$

Последнее, после нескольких преобразований, дает условие $n -$ even. Условие четности разворачивается в еще одно параметрическое описание - $\exists_m(n = 2m \;\&\; m-$целое$)$.

Через несколько шагов общей стандартизации условий вновь срабатывает анализируемый нами прием, и мы опять оказываемся в кадре отладчика ЛОСа. На этот раз значением переменной x2 служит только что созданное параметрическое описание для n. Можно убедиться, что имеется внешний кадр программы того же приема. Для этого достаточно два раза нажать "Page Up". Вернувшись в текущий кадр (два нажатия "Page Down"), просматриваем задачу x12. Она имеет три условия - $n = 2m$, $m -$ целое, $n -$ целое. Входим в просмотр списка комментариев к задаче x12 (седьмой элемент набора x12 - список списков комментариев, и последний элемент данного списка - требуемый список). В нем обнаруживается комментарий (контекст A_1 A_2), где A_2 - отложенное до редактирования ответа равенство $x = \pi n$, с помощью которого выше была исключена переменная x. A_1 - та задача, в которой произошло данное исключение.

Продолжаем трассировку внутри новой задачи x12. Как и ранее, нажимаем "Enter", "пробел", и далее каждый новый шаг вызываем нажатием "Enter". Снова исключается неизвестная, явно выраженная через другую, на этот раз с помощью равенства $n = 2m$. Возникает задача с двумя условиями - "$m -$ целое, $2m -$ целое". На первом шаге ее решения условие "$2m -$ целое" заменяется на константу "истина" и отбрасывается. В единственном оставшемся условии решатель усматривает ответ (относительно неизвестной m). Он обращается к процедуре "редакторответа", которая извлекает из комментария (контекст ...)отложенные условия $x = \pi n, n = 2m$, присоединяет их к текущему условию "$m -$ целое", и обрабатывает весь этот список во вспомогательной задаче, имеющей цель "редакция". Неизвестными вспомогательной задачи служат переменные x, n, m, причем две последние - несущественные. Поэтому после подстановки значения $n = 2m$ в первое условие равенство для n отбрасывается. По завершении редактирования ответ приобретает вид $x = 2\pi m \;\&\; m -$ целое. В момент его выдачи следует войти в кадр отладчика ЛОСа, нажав клавишу "ф". Это делается для того, чтобы продолжить трассировку применения анализируемого приема после решения задачи x12. Имеется в виду внутреннее срабатывание приема, относящееся к описанию $\exists_m(n = 2m \;\&\; m-$целое$)$. Возвращаемся к кадру данного срабатывания приема, нажимая "Page Up". Войдя в него, нажимаем "2" для установки пооператорной трассировки в кадре. Далее нажимаем "Enter" и проходим оператор "равно(x13 ответзадачи(x12))". Убеждаемся в том, что значением переменной x13 служит найденный выше ответ.

Продолжаем трассировку в кадре. Через несколько шагов переменной x15 присваивается утверждение "$\exists_m(x = 2\pi m \ \& \ m -$ целое)". Список условий текущей задачи (т.е. задачи на редактирование исходного параметрического описания) заменяется на одноэлементный набор, состоящий из указанного утверждения. Для получения окончательного ответа x16 к утверждению x15 применяется нормализатор общей стандартизации параметрических описаний "нормсуществует", который оставляет x15 неизменным. Наконец, реализуется оператор "ответ(x16)". Чтобы вернуться после выполнения данного оператора во внешний кадр, нужно переустановить режим трассировки на пошаговый (клавиша "1"). Если этого не сделать и продолжать нажимать "Enter", то следующее прерывание произойдет значительно позднее - снова по контрольной точке "трассировка(стоп 0)", но уже для другого подслучая. Переустанавливаем режим, нажимаем "Enter" и попадаем во внешний кадр на оператор "равно(x12 ответзадачи(x11))". Этот кадр - промежуточный на пути к тому кадру, где было начато редактирование исходного параметрического описания. Здесь реализуется прием, исключивший неизвестную x с помощью уравнения $x = n\pi$. Нажимаем "2" и продолжаем трассировку в кадре. Через несколько шагов приходим к оператору "ответ(x12)". Опять нажимаем "1", "Enter", и возвращаемся наконец к той точке, с которой было начато редактирование. Нажимаем "2", и далее, по уже известной траектории, доходим до оператора "ответ(x16)", завершающего срабатывание приема. Заметим, что утверждение "$\exists_m(x = 2\pi m \ \& \ m -$ целое)"на последних шагах осталось неизменным - вся фактическая работа по редактированию ответа в рассматриваемом подслучае уже была выполнена оператором "редакторответа" до возвращения по цепочке кадров. В данной задаче прием срабатывает еще один раз, что легко проверить, отключив трассировку нажатием клавиши "0" и нажав "Enter". Снова произойдет прерывание по достижении контрольной точки "трассировка(стоп 0)". В качестве упражнения, рекомендуется самостоятельно проанализировать поведение решателя на этом срабатывании приема. По окончании нужно не забыть устранить из программы приема контрольную точку "трассировка(стоп 0)".

10.19.3 Прием исключения неизвестной задачи на описание, явно выраженной через остальные неизвестные

Часто применяется при решении задач с несколькими неизвестными прием, исключающий неизвестную x с помощью условия $x = t$, где выражение t не содержит x. Применение приема происходит безотносительно к тому, расположено ли x в левой или в правой части равенства. Чтобы он мог сработать, необходима предшествующая работа других приемов, создающих равенство указанного вида. С другой стороны, если такое равенство уже имеется, но применение приема нецелесообразно, могут понадобиться приемы, срабатывающие на уровне 0 и "маскирующие" явное выражение для x (например, перенесением всех ненулевых членов числового уравнения в одну часть).

Прием находится в разделе "Приемы решателя" - "Общие приемы" - "Равенство" - "Исключение неизвестной задачи на описание, явно выраженной через остальные неизвестные" оглавления программ. Для анализа его работы рассмотрим какую - либо систему алгебраических уравнений, например, задачу 8 из раздела задачника "Элементарная алгебра" - "Решение уравнений" - "Системы алгебраических уравне-

ний - 2". Эта система имеет вид:

$$\begin{cases} x^3+y^3 &= 7z^3 \\ x-y &= 3z \\ y-z &= x-2 \end{cases}$$

Для анализа поведения данного приема нет необходимости вставлять в его программу контрольную точку "трассировка(стоп 0)". Достаточно войти в просмотр задачи, нажать "л", выбрать концевой раздел оглавления программ, содержащий прием, и нажать клавишу "курсор вправо". Почти сразу возникает прерывание с выходом в отладчик ЛОСа. Вызываем на экран значение переменной x2 - текущее уравнение "$x = y + 3z$". Операторы текущего фрагмента программы подсказывают нам, что значением программной переменной x7 служит исключаемая неизвестная x, а значением переменной x6 - вхождение этой неизвестной в равенство. Чтобы определить общий контекст, выходим на уровень просмотра задачи (клавиша "з"). Остальные два условия имеют вид $-7z^3 + x^3 + y^3 = 0$, $y - z = x - 2$. Возвращаемся в кадр отладчика (клавиша "ф"), и для продолжения трассировки в этом кадре нажимаем "2". После нескольких шагов, связанных с обработкой фильтров и учетом специальных случаев, попадаем на операторы "операнд(x2 x8) не(равно(x6 x8)) равно(x9 подтерм(x8))". Они выделяют противоположную часть x8 рассматриваемого равенства; переменной x9 присваивается выражение $y + 3z$, расположенное в этой части. Продолжаем трассировку до оператора, присваивающего переменной x10 набор результатов подстановки выражения $y + 3z$ вместо x в остальные условия. Нажимаем "К10" для просмотра этого набора. Далее создается вспомогательная задача x11, имеющая x10 своим списком условий. Просматриваем ее, нажимая "К11". В списке целей (8 - й элемент набора задачи) находим измененную цель (неизвестные y z), в которой отсутствует исключенная неизвестная x.

Через несколько шагов появляется оператор "замечание(x11 обращение)", сопровождающий новую задачу комментарием "обращение". Этот комментарий необходим для того, чтобы в процессе трассировки решения "по срабатываниям приемов" система автоматически входила в просмотр решения вспомогательной задачи x11. Оно существенно важно для понимания общего процесса решения, и пропуск его был бы нежелателен.

Еще одна важная подробность - при продолжении трассировки задача x11 сопровождается комментарием (контекст x1 A), где A - одноэлементный набор, состоящий из равенства $x = y + 3z$.

Наконец, коррекция новой задачи x11 завершается, и возникает оператор "равно(x12 ответзадачи(x11))". При рассмотрении предыдущего примера мы уже оказывались в его окрестности. В качестве самостоятельного упражнения, можно продолжить трассировку процесса решения задачи x11 до получения ответа и момента использования комментария (контекст ...). Пока мы пропускаем эту трассировку, нажимая "Enter" в очередной раз. Выполнение оператора "равно(x12 ответзадачи(x11))" требует определенного времени, так как фактически в нем решение всей задачи доводится до конца. Получаем результат x12 и выводим его на экран: $z = 1/2$ & $x = 1$ & $y = -1/2$. После нескольких дальнейших шагов сопровождающего характера (например, коррекции комментария (выводимо ...) для учета использованных посылок) прием выдает ответ x12.

10.19.4 Попытка решения уравнения из блока анализа, имеющего единственную неизвестную

В заключение рассмотрим прием, усматривающий в блоке анализа уравнение с единственной числовой неизвестной x и предпринимающий попытку решить его относительно x. Заметим, что к уравнениям относительно единственного атомарного числового подвыражения t с неизвестными, не являющегося переменной, данный прием неприменим. Чтобы разрешить такие уравнения относительно выражения t, нужно вводить для этого выражения вспомогательное обозначение - новую неизвестную y, и лишь тогда прием сработает.

Найти программу приема можно в разделе "Приемы решателя" - "Общие приемы" - "Равенство" - "Попытка решения уравнения из блока анализа, имеющего единственную неизвестную". Он часто срабатывает в тех предметных областях, где интенсивно используется блок анализа, например, в геометрии. Рассмотрим в качестве примера простую планиметрическюу задачу на вычисление, расположенную в разделе задачника "Элементарная геометрия" - "Задачи на вычисление" - "Треугольник" - "Общий случай" - "Биссектрисы" и имеющую номер 1. Дан треугольник ABC, у которого известны длины b, c сторон AC, AB, причем длина биссектрисы AD равна длине отрезка DB. Требуется найти длину стороны BC.

Чтобы проанализировать срабатывание приема, войдем в просмотр условия задачи, нажатием клавиши "л" перейдем в оглавление программ, найдем указанный выше подраздел приема, выберем в нем концевой пункт "Формирование вспомогательной задачи на описание x9", и нажмем "Enter". Почти сразу появится кадр отладчика - контрольная точка "прием(1 3)", после которой расположен выделенный малиновым цветом оператор "равно(x9 ...)". Как следует из названия пункта, данный оператор присваивает переменной x9 заготовку создаваемой вспомогательной задачи на решение уравнения. Просматриваем само уравнение - его корневое вхождение является значением переменной x2. Уравнение имеет вид "$(b + x)(x - b) = bc$".

В качестве самостоятельного упражнения, можно порекомендовать просмотреть предысторию появления данного уравнения путем трассировки на уровне применений приемов. При этом нужно постараться точно определить момент смены режима трассировки, чтобы снова попасть в данный кадр отладчика ЛОСа.

Нажатием клавиши "2" входим в режим пооператорной трассировки, нажимаем "Enter" и просматриваем с помощью процедуры сквозного просмотра заготовку задачи x9 (нажимается "К9"). Видно, что в список посылок отобраны утверждения $0 < c$, $0 < b$, $b -$ число, $c -$ число. Единственным пока условием является уравнение; неизвестной служит переменная x. Вернувшись из сквозного просмотра в кадр отладчика, продолжаем нажимать "Enter". Попадаем в цикл просмотра условий задачи x9 и обращений к справочнику "одз", пополняющему список условий сопровождающими утверждениями. На некотором шаге этого цикла переменной x16 присваивается утверждение "x - число", заносимое в список x10. Он представляет собой накопитель условий и посылок, которые впоследствии будут присоединены к задаче x9. После цикла учета о.д.з. продолжаются попытки пополнения списка условий задачи x9 путем усмотрения различных простых дополнительных ограничений на x, вытекающих из всего списка посылок блока анализа. На некотором шаге трассировки обнаруживаем присвоение переменной x18 утверждения $0 \leq x$, вытекающего из равенства $l(BC) = x$. Далее предпринимается обращение к оператору "разныеточки",

устанавливающему различие точек B, C, и замена x18 на более сильное утверждение $0 < x$. Затем x18 присоединяется к списку x10. Дальнейшая трассировка не изменяет содержащих x утверждений списка x10, хотя в этот список попадают еще два утверждения - $0 < b$ и $0 < c$. После достаточно длинной цепочки шагов (в том числе цикла попыток упрощения утверждений из x10 относительно списка посылок задачи x9), приходим к оператору "длялюбого(x12 если входит(x12 x10)то альтернатива(...))". Этот оператор добавляет к списку условий задачи x9 все утверждения списка x10, содержащие x, а к списку посылок - все остальные утверждения. После его выполнения идут операторы "уровеньобращения(6) равно(x12 ответзадачи(x9))", обращающиеся к решению задачи x9. Здесь можно снова нажать "K9" и посмотреть на окончательную версию данной задачи. Если нужно посмотреть ход решения задачи x9, то, вернувшись в кадр отладчика, устанавливаем пошаговый режим трассировки (клавиша "1"), нажимаем "Enter" и оказываемся в программе символа "описать". Нажимаем "пробел" для перехода к трассировке по срабатываниям приемов, и далее прослеживаем ход решения. Получив уравнение $x^2 = bc + b^2$, решатель сразу извлекает корень, отбрасывая отрицательное значение. Здесь помогает извлеченное из блока анализа дополнительное условие $0 < x$. После упрощения, выдается ответ $x = \sqrt{b(b+c)}$. Для возвращения в отладчик ЛОСа нажимаем "ф". Нажимая "PageUp", возвращаемся в тот кадр, где был расположен оператор "равно(x12 ответзадачи(x9))". Здесь переходим в режим трассировки по операторам данного кадра (клавиша "2"). Нажимая далее "Enter", выходим на оператор, следующий за "равно(x12 ответзадачи(x9))". Убеждаемся, что значением переменной x12 стал найденный выше ответ. После выполнения ряда операторов, связанных с коррекцией комментариев для предстоящего преобразования, приходим к оператору "удалениепосылки(x1 x3)". Он удаляет посылку - рассматриваемое уравнение блока анализа. Вместо этого в список посылок вводятся конъюнктивные члены утверждения x12, сопровождаемые необходимыми комментариями. Выполнение приема завершает оператор "пересмотр".

10.19.5 Упражнения

1. Найти в разделе задачника "Теория множеств" задачу, использующую прием доказательства от противного. Посмотреть ход рассуждений при получении противоречия.

2. Найти в том же разделе все задачи, использующие прием подбора подстановки вместо неизвестных, преобразующей все условия в посылки.

3. Найти прием "фиксация значения связанной переменной через равенство в антецеденте". Найти в разделе "Элементарная алгебра" задачу, в которой он применяется.

4. В подразделе оглавления программ "Импликативное условие задачи на описание, не требующей полного ответа" найти пункт "Попытка обратного вывода с помощью кванторной посылки". Найти в разделе задачника "Математический анализ" - "Задачи на доказательство" задачу, в которой этот прием срабатывает. Проанализировать это срабатывание.

5. Найти прием "Решение относительно параметра числового равенства без неизвестных". Найти в разделе задачника "Логарифмические уравнения" такую задачу, где этот прием применяется к равенству, не являющемуся линейным относительно своего параметра.

6. Найти прием "Подбор примера с заменой функциональной неизвестной на обычную". Найти в разделе задачника "Пределы" - "Теоретические задачи" задачу, в которой этот прием срабатывает. Проанализировать срабатывание.

Глава 11

Язык для записи приемов ГЕНОЛО́Г

В первых версиях решателя задач все приемы были реализованы на языке ЛОС. Анализ этих приемов позволил прийти к выводу, что в подавляющем большинстве случаев программа приема основана на применении какой-либо одной теоремы предметной области. Чтобы ускорить программирование приемов и сделать программы их достаточно наглядными (последнее особенно важно для возможностей быстрой коррекции логики принятия решений в процессе обучения), в этой ситуации естественно было разработать язык, в котором прием представляется как теорема предметной области, снабженная некоторой алгоритмизирующей разметкой. Таким образом и возник язык, получивший название ГЕНОЛО́Га (ГЕНетический язык ЛОГического программирования). Алгоритмизирующая разметка теоремы здесь играет роль "генотипа" приема, из которого компилятор создает собственно программу на ЛОСе, реализующую прием.

Развитие ГЕНОЛОГа означает, фактически, описание и систематизацию различных алгоритмических "трюков", применение которых позволяет преобразовать теорему в эффективную программу. Это предполагает рассмотрение все новых и новых предметных областей, которые, как показывает опыт развития решателей, постоянно вносят новые элементы в копилку "алгоритмизаторов" и логических режимов, используемых при решении задач. По этой причине, освоение новой предметной области, как правило, требует (в основном небольших) добавлений к реализованной на ЛОСе программе компилятора для ГЕНОЛОГа. Впрочем, эта работа по своей трудоемкости оказывается несопоставима с последующей экономией, достигаемой при использовании ГЕНОЛОГа - собственно программирование приемов и анализ их устройства при коррекциях ускоряется по сравнению с ЛОСом на порядок. Следует также добавить, что поток новых элементов языка, после проработки значительного обучающего материала, явно идет на убыль.

Помимо соображений экономии трудоемкости программирования и придания описаниям приемов возможно большей наглядности, ГЕНОЛОГ открывает возможность автоматизации синтеза новых приемов, так как играет роль "связующего звена" между логическим языком, на котором происходит накопление новых знаний в предметной области, и алгоритмическим языком, заставляющим работать эти знания при решении задач.

В настоящее время подавляющее большинство приемов решателя реализованы на ГЕНОЛОГе. Он обеспечивает быстрое преобразование логических описаний предметной области (теорем) в описания для логических структур данных, каковыми

являются ЛОС - программы. Однако, остается небольшая часть приемов, основанных на утверждениях, которые уже относятся к структурам данных (в основном это приемы "общелогического" типа), и для них надобность в специальном переходнике от предметного к структурному уровню отсутствует. Эти приемы реализованы непосредственно на ЛОСе. Впрочем, часть реализованных пока на ЛОСе приемов имеют в себе невырожденную компоненту знаний в конкретных предметных областях и, при некотором развитии ГЕНОЛОГа, могли бы также быть переведены на него.

11.1 Основные компоненты описания приема на языке ГЕНОЛО́Г

Описание приема на ГЕНОЛОГе состоит из следующих компонент:

1. Теорема предметной области, на которой основан прием. Эта теорема вводится при помощи формульного либо текстового редактора, в стандартной математической либо скобочной записи. Для большей наглядности предусмотрено сопровождение теорем с геометрическими объектами чертежом; компилятор этот чертеж никак не использует.

   Заметим, что хотя ГЕНОЛОГ и приближен к логическому языку предметной области, все же он представляет собой язык программирования. Настоящие теоремы предметной области содержатся в базе теорем. В теореме приема же допустимы (как правило, небольшие) отклонения от логических стандартов предметного уровня. Эта теорема обычно преобразуется к виду, более удобному для практического использования - например, в посылках могут вводиться вспомогательные обозначения для объектов, упоминаемых в утверждении теоремы; часть посылок, которые в обычных "разумных" контекстах автоматически бывают выполнены, отбрасывается, и т.п. В особых случаях теорема приема вообще может представлять собой техническую заглушку - "псевдотеорему".

2. Заголовок приема. Он указывает на общую схему использования теоремы в приеме.

3. Список фильтров приема. Элементы этого списка суть сформулированные с помощью несколько модифицированного логического языка условия на целесообразность применения приема в текущем контексте. Это - второй логический уровень описания приема; первым уровнем является сама теорема, также задаваемая на логическом языке. Однако, в случае теоремы логический язык используется для описания ситуаций на предметном уровне, а в случае фильтров - для описания ситуаций на структурном уровне. В этом он близок ЛОСу и может рассматриваться (с рядом существенных оговорок) как "ЛОС в теоремных переменных". В действительности многие операторы ЛОСа имеют своих одноименных двойников в языке фильтров приемов, хотя эти двойники используются уже по-другому - у них часто отбрасываются однозначно восстанавливаемые из контекста описания приема операнды. Кроме того, в языке фильтров приема возникают конструкции, совершенно не имеющие аналогов в ЛОСе.

4. Список указателей компилятору (далее называем их просто указателями приема). Элементы этого списка уточняют способ формирования ЛОС-программы

приема компилятором ГЕНОЛОГа. По существу, это основная часть "генотипа" приема. Здесь уточняются: средства, используемые для усмотрения в задаче "замаскированных" возможностей применения теоремы; способы обработки отдельных посылок теоремы; технические комментарии, вводимые в задачу либо удаляемые при применении приема; логика переключения внимания после применения приема, и др.

5. Список нормализаторов. При формировании результирующих (включаемых в структуру данных задачи) либо вспомогательных (рассматриваемых в процессе работы приема) термов могут применяться процедуры для их упрощения либо каких-либо иных специальных преобразований. Такие процедуры задаются последовательным применением нормализаторов - либо специальных пакетов приемов для тождественных и эквивалентных преобразований, либо вспомогательных задач. Использование небольшого пакета приемов вместо общей базы приемов, активизируемой при сканировании задачи, во многих случаях на порядок ускоряет вычисления. Эти "локальные" приемы отличаются от приемов сканирования задачи - они применяются не в контексте задачи, а в некотором другом специальном контексте, состоящем из преобразуемого терма, списка посылок, в предположении истинности которых ведутся преобразования, и списка технических комментариев. Задаются они, как и приемы сканирования задач, на ГЕНОЛОГе.

   Специальный интерфейс позволяет достаточно быстро вводить обращения к нормализаторам путем указания в теореме (либо в сопровождающих теорему термах описания приема) подлежащих нормализации точек. В результате использования нормализаторов и вспомогательных процедур обработки посылок теоремы, срабатывание приема оказывается иногда чрезвычайно сложным вычислительным процессом, основанным не на единственной теореме предметной области, а на десятках и даже сотнях (с учетом прменяемой рекурсии) различных теорем.

6. Шаблоны сопровождения. Здесь хранятся текст-формульные шаблоны, используемые при создании текстов, объясняющих срабатывания приемов. Способ их применения регламентируется специальными указателями приема.

Для примера описания приема по данной схеме, рассмотрим простейший прием решения уравнения $ax = b$.

Теорема приема имеет в этом случае вид:

$\forall abx(\text{число}(a)\ \&\ \text{число}(b)\ \&\ \text{число}(x) \Rightarrow (ax = b \Leftrightarrow (\neg(a = 0)\ \&\ x = b/a) \vee (a = 0\ \&\ b = 0)))$

Заголовком приема служит логический символ "второйтерм", указывающий, что теорема будет применяться для эквивалентной замены слева направо.

Фильтры приема указывают следующий контекст, в котором выполняется замена:

а) Тип решаемой задачи - на описание;

б) Преобразуемое равенство входит в условие задачи, причем это вхождение - корневое;

в) Текущий уровень сканирования, при котором происходит срабатывание приема, равен 1;

г) Выражение b не содержит неизвестных задачи, а x - содержит.

Указания компилятору уточняют следующие подробности:

а) Переменная x идентифицируется как совокупность всех множителей рассматриваемого произведения, содержащих неизвестные задачи (это автоматически предопределяет, что переменная a будет идентифицирована с выражением без неизвестных);

б) При идентификации выражения ax возможен учет стоящего перед ним знака "минус", который относится программой приема к коэффициенту a;

в) Посылки теоремы проверяются при помощи специального пакета приемов, обеспечивающего быстрое усмотрение условий вида "число(...)".

В приеме используются следующие обращения к нормализаторам:

а) Предпринимаются обращения к вспомогательному пакету приемов для разложения на множители выражений a, b. Если такие разложения не находятся сразу же, до преобразования рассматриваемого уравнения, то впоследствии может возникнуть разбор случаев, приводящий к дублирующим попыткам разложения на множители в различных подслучаях;

б) Выполняется обращение к пакету стандартизации дробных выражений для выражения b/a. Этот пакет состоит из нескольких десятков простых продукций, выполняющих, в частности, сокращение дробей, деление и умножение дробных выражений, учет констант 0 и 1, и т.п.;

в) Для уравнений $a = 0, b = 0$ выполняются обращения к пакетам приемов, стандартизирующим уравнения данного вида (сокращение левой части на множители, не обращающиеся в 0; усмотрение тождественной ложности; преобразование равенства нулю произведения в дизъюнкцию равенств нулю сомножителей, и т.п.);

г) К заменяющей дизъюнкции применяется пакет общелогических приемов (устранение логических констант "истина" и "ложь"; стандартизация конструкций с логическими связками и кванторами).

В результате выполнения указанных преобразований заменяющий терм в данном приеме приобретает достаточно устойчивый вид, не вызывающий немедленного срабатывания цепочки простейших нормализующих приемов. Заметим, что трудоемкость одного цикла сканирования задачи обычно возрастает пропорционально суммарной длине находящихся в активной области ее термов, а всего времени решения задачи - пропорционально квадрату средней такой длины. Поэтому вынесение во вспомогательные задачи преобразований подтермов сложного выражения приводит, если эти преобразования достаточно интенсивны, к значительному ускорению решения задачи. Соответственно, стандартом программирования на ГЕНОЛОГе является как можно большее использование нормализаторов.

11.2 Типы заголовков приемов

Перечислим в этом разделе типы заголовков приемов, используемые в ГЕНОЛОГе.

11.2.1 Приемы, осуществляющие тождественную либо эквивалентную замену

1. Замена подтерма с помощью тождества либо эквивалентности. Если теорема приема имеет вид $\forall x(A \to B = C)$ либо вид $\forall x(A \to (B \leftrightarrow C))$, то заголовок приема, использующего эту теорему для замены слева направо (B на C) есть "второйтерм"; заголовок для замены справа налево - "первыйтерм". Название заголовка приема здесь указывает на номер заменяющего операнда в равенстве либо эквивалентности. Заметим, что приемы с указанными двумя заголовками (а также приемы вывода в посылках) - наиболее часто встречающиеся в базе приемов. Пример приема эквивалентной замены был приведен в предыдущем подразделе.

2. Замена группы условий задачи на описание при помощи эквивалентности. Если теорема приема имеет вид $\forall x(A \to ((B_1 \ \& \ \ldots \ \& \ B_n) \leftrightarrow C))$, где $n \geq 1$, то на ней может быть основан прием, идентифицирующий группу условий задачи на описание с утверждениями $B_1, \ldots, B_n$, и заменяющий их на утверждение C (если оно само является конъюнкцией, то прием не разбивает ее на атомарные подутверждения - это выполняет после его применения другой прием). Заголовок приема имеет вид "заменаусловия(второйтерм)", если замена выполняется слева направо; если же части приведенной выше эквивалентности поменять местами и замена будет выполняться справа налево, то заголовок приема будет иметь вид "заменаусловия(первыйтерм)". Заметим, что рассматриваемая эквивалентность может одновременно порождать и другой прием ГЕНОЛОГа - для эквивалентной замены конъюнкции $B_1 \ \& \ \ldots \ \& \ B_n$, встречающейся внутри некоторого одного терма задачи. Здесь уже будет использован заголовок "второйтерм" (соответственно, "первыйтерм"). Пример приема замены группы условий - использование теоремы $\forall xAB(x \in A \ \& \ x \in B \leftrightarrow x \in (A \cap B))$ для группировки условий на неизвестную x при известных множествах A, B.

3. Замена группы посылок задачи при помощи эквивалентности. На теореме указанного выше вида $\forall x(A \to ((B_1 \ \& \ \ldots \ \& \ B_n) \leftrightarrow C))$, где $n \geq 1$, может быть основан также прием, идентифицирующий группу посылок задачи с утверждениями $B_1, \ldots, B_n$, и заменяющий их на утверждение C. Заголовок этого приема - "заменатермов(второйтерм)" в случае замены слева направо либо "заменатермов(первыйтерм)" в случае замены справа налево (в последнем случае части указанной выше эквивалентности переставлены). Пример приема замены группы посылок - использование теоремы $\forall x(\text{число}(x) \to 0 \leq x \ \& \ \neg(x = 0) \leftrightarrow 0 < x)$ для объединения нестрогого неравенства и отрицания равенства нулю в строгое неравенство.

4. Исключение несущественных неизвестных задачи на описание. Если теорема приема имеет вид $\forall x(A \to (\exists y(B) \leftrightarrow C))$, то на ней может быть основан прием, идентифицирующий конъюнктивные члены утверждения B со всеми условиями $U_1, \ldots, U_m$ задачи на описание, содержащими неизвестные списка y. Если все эти неизвестные - несущественные (т.е. достаточно установить существование их значений, не включая информацию об этих значениях в ответ задачи; такие неизвестные $z_1, \ldots, z_k$ определяются целью "параметры $z_1 \ldots z_k$"), то теорема позволяет заменить $U_1, \ldots, U_m$ на C и устранить неизвестные y из списка неизвестных. Прием, выполняющий такое преобразование, имеет заголовок

"связка". Пример приема исключения несущественных неизвестных - использование теоремы $\forall abc(\exists c(a < c \ \& \ c < b \ \& \ \text{число}(c)) \leftrightarrow a < b)$ для исключения несущественной неизвестной c, ограничения на которую сводятся к указанию числового типа значений и интервала значений.

5. Лексикографическое упорядочение операндов коммутативных операций и предикатов. Если некоторая операция (предикат) A не изменяет своего значения при любой перестановке операндов, то с ней связан прием, обеспечивающий лексикографическое упорядочение операндов. При выдаче терма на экран обычно происходит некоторое стандартное переупорядочение операндов для прорисовки формул в их привычном виде, однако во внутреннем представлении для коммутативных логических символов постоянно применяется именно лексикографическое упорядочение. Заголовок приема, выполняющего такое упорядочение, - логический символ "лексупорядочение". Теорема приема в этом случае имеет вид "коммутативно(A)" (например, "коммутативно(умножение)").

6. Циклическая перестановка операндов для размещения на первом месте первого в лексикографическом порядке операнда (для теоремы "циклупорядочение(A)"). Если операция A не изменяет своего значения при произвольных циклических перестановках операндов, то используется прием, переносящий на первое место первый в лексикографическом порядке операнд. Заголовок приема - "циклупорядочение"; теорема его - "циклупорядочение(A)".

7. Устранение вложенных одинаковых ассоциативно - коммутативных операций. Для устранения вложенных одинаковых ассоциативно - коммутативных операций A: $A(\ldots A(\ldots)\ldots) = A(\ldots\ldots\ldots)$ - служит прием, имеющий заголовок "спускоперандов". Теорема этого приема имеет вид "коммутативно(A)".

8. Лексикографическое упорядочение элементов набора, расположенного под одноместной операцией. Иногда многоместная операция с переменным числом операндов представляется в виде одноместной операции над набором, представленным термом вида "набор($A_1 \ldots A_n$)". В тех случаях, когда значение этой операции не зависит от порядка элементов набора, используется прием, выполняющий лексикографическое упорядочение операндов $A_1, \ldots, A_n$. Заголовок этого приема - "стандупорядочение(A)"; теорема его - "список(A)".

11.2.2 Приемы для вывода следствий

1. Вывод следствий в посылках задачи. Теорема вида $\forall x_1 \ldots x_n(A_1 \ \& \ \ldots \ \& \ A_m \rightarrow A_0)$ может быть использована для вывода следствий в посылках задачи. Часть утверждений $A_1, \ldots, A_m$ при этом идентифицируется с посылками; остальные утверждения этого списка - проверяются с помощью вспомогательных процедур либо служат для ввода вспомогательных обозначений. В список посылок заносится утверждение A_0. Заметим, что иногда список непосредственно идентифицируемых с посылками утверждений A_i может быть и пустым: попытка применения приема может начинаться с усмотрения встречающегося где-либо в задаче (в том числе и в условиях) терма специального вида, и далее выполняются лишь проверки истинности необходимых A_i с помощью вспомогательных процедур. Такие режимы применения приема уточняются специальными его указателями. Заголовок приема, обеспечивающего вывод в посылках

задачи - "вывод". Пример приема вывода в посылках - использование теоремы $\forall ABC(B \in$ прямая(AC) & $l(AB) = l(BC)$ & разныеточки$(A, C) \rightarrow B \in$ отрезок $(AC))$ для усмотрения принадлежности отрезку точки, равноудаленной от его концов и лежащей на той же прямой, что и эти концы.

2. Вывод следствий в условиях задачи на описание. Теорема указанного выше вида $\forall x_1 \ldots x_n(A_1$ & $\ldots$ & $A_m \rightarrow A_0)$ может быть использована не только для вывода следствий в посылках, но для вывода следствий в условиях задачи на описание. Вывод такого рода используется достаточно редко, так как загромождение списка условий задачи следствиями исходных условий нежелательно - усложняется последующая проверка найденных значений. Как правило, совместные следствия условий и посылок извлекаются в блоке анализа задачи на описание - специально предназначенной для этого вспомогательной задаче на исследование. Часть утверждений $A_1, \ldots, A_m$ (возможно, пустая) идентифицируется с посылками либо условиями; остальные утверждения этого списка - проверяются с помощью вспомогательных процедур либо служат для ввода вспомогательных обозначений. В список условий заносится утверждение A_0. Заголовок приема, обеспечивающего вывод в условиях задачи на описание - "выводусловия". Большей частью, вывод в условиях задачи на описание используется либо для регистрации дизъюнкции, по которой будет выполняться разбор случаев, либо для регистрации утверждений, подготавливающих такой разбор случаев (например, утверждений о принадлежности значения целочисленной неизвестной некоторому конечному промежутку). Пример приема вывода в условиях задачи на описание - использование теоремы $\forall xn(x < n$ & натуральное$(x) \rightarrow \exists m(m \in \{1, \ldots, n-1\}$ & $x = m))$ для занесения в условия задачи утверждения $(m = 1 \vee \ldots \vee m = n-1)$ & $x = m$, позволяющего найти неизвестное значение x путем разбора случаев.

11.2.3 Обратный вывод для условий задачи

1. Использование кванторной импликации для обратного вывода условия задачи на описание. Теорема вида $\forall x(A_1$ & $\ldots$ & $A_n \rightarrow A_0)$ - утверждения такого вида называем кванторными импликациями - может быть использована для попытки замены условия задачи на описание, идентифицированного с A_0, на группу антецедентов A_i, выделенных в описании приема указателями "подборзначений(i)". Перед попыткой проверяется истинность остальных антецедентов. Прием, реализующий эти действия, имеет заголовок "подборзначений". Применение его допустимо лишь в задачах, не требующих получения полного описания множества допустимых значений неизвестных. Чаще всего прием применяется для выбора конкретного значения неизвестной, входящей в A_0, если прочие ограничения на эту неизвестную сводятся только к указанию типа значения. Пример приема - использование теоремы $\forall ax(x = \emptyset \rightarrow x \subseteq a)$ для выбора пустого множества в качестве неизвестного подмножества x некоторого множества a.

2. Попытка использования параметрического описания для эквивалентного преобразования условия задачи на описание. Теорема вида $\forall x(A \rightarrow (B(x) \leftrightarrow \exists y(F(x, y))))$ может быть использована для замены условия задачи на описание, идентифицированного с утверждением $B(x)$, на параметрическое описание $\exists y(F(x, y))$

(в большинстве случаев $F(x,y)$ дает явное выражение x через y, но не обязательно). Такая замена реализована как переход к вспомогательной задаче, условия которой получены из условий текущей задачи заменой $B(x)$ на группу конъюнктивных членов утверждения $F(x,y)$ и присоединением новых переменных y к списку неизвестных. Если вспомогательная задача решена, то на ее ответ навешивается квантор существования по y и предпринимается редактирование полученного таким образом параметрического описания; если же ее решить не удалось, то возобновляется процесс решения текущей задачи без применения данной замены. Заголовок приема, выполняющего указанные действия - "параметризация". Заметим, что хотя теорема приема и имеет вид эквивалентности, допустимо применение приема и в случаях, когда эквивалентность не имеет места - если в задаче не требуется получить полный ответ (например, есть цель "пример"), то достаточно лишь, чтобы имела место импликация в направлении справа налево. Формально, для простоты компиляции, и в этом случае теорема приема записывается в виде эквивалентности. Пример приема - использование теоремы $\forall ab(a \subseteq b \leftrightarrow \exists c(\text{set}(c)\ \&\ b = a \cup c))$ для представления неизвестного надмножества b множества a виде объединения a с некоторым вспомогательным множеством c, которое далее будет рассматриваться как новая неизвестная; прием применяется, если нужно лишь подобрать какой-либо пример для b.

3. Попытка решить задачу на доказательство путем идентификации ее условия с консеквентом кванторной импликации, а посылок - с антецедентами этой импликации. Если теорема имеет вид $\forall x(A \rightarrow \forall y(B_1\ \&\ \ldots\ \&\ B_n \rightarrow B_0))$, то ее можно использовать для решения задачи на доказательство, условие которой идентифицируется с B_0, а утверждения $B_1, \ldots, B_n$ - со всеми посылками, содержащими переменные списка y. При установлении истинности конъюнктивных членов утверждения A (без использования посылок, содержащих переменные списка y) с помощью средств, уточняемых в указателях приема (решение вспомогательных задач на доказательство, проверочных пакетных операторов и т.п.) выдается ответ "истина". Прием имеет заголовок "свертка". В решателе приемы такого типа используются для доказательств по индукции. Пример - использование теоремы $\forall mg(\text{натуральное}(m)\ \&\ g(m)\ \&\ \forall n(\text{натуральное}(n)\ \&\ g(n)\ \&\ 0 \le n - m \rightarrow g(n+1)) \rightarrow \forall n(\text{натуральное}(n)\ \&\ 0 \le n - m \rightarrow g(n)))$ для доказательства утверждения $g(n)$ при всех натуральных n, не меньших m, индукцией по n.

11.2.4 Приемы для усмотрения ответа задачи

1. Усмотрение явного описания для значения неизвестной. Для задания вида явного описания значений неизвестной x используется псевдотеорема "явное(x набор($f_1 \ldots f_n$) набор($g_1 \ldots g_m$) набор($h_1 \ldots h_k$))". Здесь $f_1, \ldots, f_n$ - все не более чем однократно встречающиеся в явном описании утверждения; $g_1, \ldots, g_m$ - все утверждения, которые могут встречаться в явном описании несколько раз. Каждая отличная от x переменная может встречаться в наборе $f_1, \ldots, f_m, g_1, \ldots, g_m$ лишь однократно; при неоднократном использовании некоторого g_i такие переменные переобозначаются в g_i на новые переменные. $h_1, \ldots, h_k$ - условия на отличные от x переменные явного описания, при которых оно непротиворечиво (набор этих условий, по возможности более полный, тем не менее не

обязан гарантировать непустоту множества удовлетворяющих явному описанию значений x). Возможны случаи $n = 0; m = 0; k = 0$ - тогда вместо терма "набор(...)" используется логический символ "пустоеслово".

Псевдотеорема набирается при помощи текстового редактора. На ней основана группа приемов, соответствующих идентификации, начинающейся с усмотрения различных утверждений списка $f_1, \ldots, f_n, g_1, \ldots, g_m$. Заголовки этих приемов имеют, соответственно, вид "ответ(f_1)", ..., "ответ(f_n)", "ответ(g_1)", ..., "ответ(g_m)". Для удобства ввода таких приемов рекомендуется, набрав текст псевдотеоремы и введя логический символ, за которым закрепляется прием, далее нажать Esc и "a" (кир.). Тогда включается цикл автоматической генерации указанной серии приемов - для регистрации очередного приема следует нажимать F3, и далее - снова "a", пока новые приемы не иссякнут.

Указанное выше дублирование точек активизации приема позволяет выявлять ответ задачи сразу же, как только он появляется - в противном случае точка привязки могла бы находиться в одном из условий задачи, имеющем большой вес и таким образом находящемся вне поля зрения решателя, в то время как только что измененное условие (веса 0), после преобразования которого возникла требуемая форма ответа - не анализировалось бы с точки зрения усмотрения ответа.

Прием срабатывает при текущем уровне сканирования, равном 1; сопровождение его фильтрами не предусмотрено (компилятор игнорирует указание фильтров).

Пример приема - использование псевдотеоремы "явное(x1 набор(число(x1)меньшеилиравно(x1 x2)меньшеилиравно(x3 x1))набор(не(равно(x1 x4)))пустоеслово)" для усмотрения явного описания значений числовой неизвестной x1 в виде одного, двух (либо нуля) нестрогих неравенств, ограничивающих ее промежуток изменения, быть может, сопровождаемых указанием конечного числа отбрасываемых точек.

2. Выдача ответа задачи на описание с учетом контекста. Хотя приемы выдачи ответа задачи на описание при усмотрении явного описания значений ее неизвестной являются наиболее употребительными, они обладают одним недостатком - применение их не может быть ограничено с учетом целевой установки задачи. Поэтому введена другая форма усмотрения и выдачи ответа задачи на описание. В этом случае теорема приема имеет вид A_1 & ... & A_n - эта конъюнкция определяет вид списка условий задачи, при котором следует выдавать ответ. Специальные указатели приема ("смответ", "серия") уточняют, в котором из утверждений A_i выбирается точка привязки - с рассмотрения ее будет начата попытка применения приема, а также выделяют те A_i, которые могут идентифицироваться с несколькими условиями задачи. Прием имеет заголовок "ответзадачи". В отличе от приема "ответ", он допускает обычное использование фильтров для уточнения контекста срабатывания. Пример приема - использование описания число(x) & $a \le x$ & $x \le b$ & $f(x) \le y$ & $y \le g(x)$ & число(y) вида ответа при решении системы неравенств с двумя неизвестными x, y. Фильтры приема уточняют, что выражения a, b не содержат неизвестных, а выражения $f(x), g(x)$ - не содержат неизвестной y.

11.2.5 Перенесение во внешнюю задачу на описание утверждений из блока анализа

В процессе вывода совместных следствий из условий и посылок задачи на описание (такой вывод реализуется во вспомогательной задаче на исследование, называемой блоком анализа данной задачи на описание) может сложиться необходимость передачи некоторых ценных следствий во внешнюю задачу на описание. Это выполняется приемом с заголовком "замещениеусловий"; псевдотеорема приема имеет вид "замещениеусловий(A)", где A - утверждение, идентифицируемое с посылкой блока анализа, переносимой во внешнюю задачу на описание. При перенесении указанного утверждения предпринимается попытка исключить возможно большее число условий задачи на описание, если они становятся следствиями A и некоторых (дополнительно переносимых) простых утверждений, имеющих по отношению к A сопровождающий характер. Специальный указатель приема ("обрывзадачи") может сразу же по перенесении A прервать сканирование блока анализа и возобновить сканирование внешней задачи на описание.

Пример приема - использование псевдотеоремы "замещениеусловий(эллипс(x1))" для передачи в список условий внешней задачи на описание, решаемой с целью характеризации типа кривой второго порядка x1, найденного типа "эллипс(x1)" этой кривой. Прием не имеет указателя "обрывзадачи", и после передачи указанного утверждения анализ кривой продолжается.

11.2.6 Ввод и удаление комментариев

Предусмотрены приемы "управляющего" характера, которые не вносят каких-либо изменений в логические структуры данных задачи, а выполняют ввод, удаление либо изменение ее комментариев. Псевдотеорема такого приема имеет вид "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то контекст(A_0))", если идентификация начинается с усмотрения вхождения терма A_0, либо "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то $\emptyset$)", если идентификация начинается с усмотрения некоторой посылки либо условия (в зависимости от указания в фильтрах приема) A_i. Собственно действия с комментариями обеспечиваются указателями приема так же, как это происходит при применении невырожденных приемов, изменяющих логические структуры данных. Приемы данного типа имеют заголовок "замечание". Примеры использования указателей для определения преобразований комментариев задачи будут приведены в разделе, посвященном таким указателям.

11.2.7 Пакетные операторы и их приемы

Процедура сканирования задачи оказалась чрезвычайно эффективным "диспетчером", организующим совместные действия по решению задачи всего многообразия приемов, накопленных при обучении. По существу, это - механизм "внутреннего логического зрения" интеллектуальной системы, позволяющий ей планировать свои действия на основе прямого доступа к огромному массиву знаний и умений.

Однако, этот механизм имеет существенный недостаток - полный просмотр и осмысление всего "поля задачи" оказывается весьма трудоемкой процедурой. Как показывает трассировка, основную часть времени решатель, сканирующий задачу, затрачивает на обработку начальных отрезков фрагментов программ встречаемых им в

задаче логических символов - то есть в каком-то смысле "впустую". На этом фоне какая-либо оптимизация либо замедление вычислений, выполняемых фактически реализуемыми приемами, оказывается практически незаметной - она не влияет на итоговое время решения задачи.

Наоборот, весьма ощутимо сказывается на времени работы решателя уменьшение числа циклов сканирования за счет укрупнения процедур, выполняемых отдельными приемами. Оптимизация решателя оказалась связанной с максимальным увеличением объема вычислений и преобразований, выполняемых приемом вне цикла сканирования. Поведение его как-бы состоит из циклов "планирования" (поиск наилучшего приема при сканировании) и реализации найденного плана. Чем больше необходимых действий вовлекается в этот план, тем меньше будет число сканирований, и тем больше - быстродействие решателя.

Одним из наиболее эффективных средств для "укрупнения" приемов сканирования задачи оказалось использование в них пакетов "локальных" приемов, применяемых уже не в контексте сканирования всей задачи, а для обработки каких-либо выделенных подтермов. Такие объединения локальных приемов получили название пакетных операторов. Эффективность их применения объясняется малым числом используемых приемов и обычно малым по сравнению со всей задачей размером обрабатываемых термов. Приемы пакетного оператора упорядочены по уровням срабатывания, и на каждом уровне организованы в древовидную процедуру, позволяющую отсекать заведомо не реализуемые ситуации. Посимвольного просмотра обрабатываемых термов здесь не происходит - поиск приема выполняется однократным прохождением программы пакетного оператора.

Приемы пакетных операторов (как и приемы сканирования задачи) задаются на ГЕНОЛОГе и затем компилируются в программы ЛОСа. В них допускаются обращения к другим пакетным операторам и рекурсивные обращения. Возникло 4 основных типа пакетных операторов, аналогичных 4 основным типам задач (на доказательство, описание, преобразование и исследование):

1. Аналогом задачи на доказательство является проверочный оператор, получающий в качестве входных данных проверяемое утверждение (точнее, набор выражений, подставляемых вместо переменных в "шаблон", задающий вид такого утверждения); список посылок, в предположении истинности которых проводится проверка, а также список комментариев управляющего характера. Помимо проверки истинности, проверочный оператор также выдает список посылок, фактически использованных при проверке. Пример проверочного оператора - оператор "усмменьше", получающий в качестве входных данных: выражения a, b, для которых проверяется условие $a < b$; список посылок, в предположении истинности которых происходит проверка; набор комментариев (в частности, в нем могут содержаться ограничения на применяемые при проверке средства).

2. Аналогом задачи на описание является пакетный оператор, названный синтезатором. Он предназначен для решения задач на описание с единственным условием, имеющим строго ограниченный вид, определяемый некоторым "шаблоном". Часть переменных шаблона выделена как "известные", а часть - как "неизвестные". Синтезатор определяет значения неизвестных, получая в качестве входных данных: набор выражений, подставляемых вместо "известных" переменных в шаблон; список посылок, в предположении истинности которых

предпринимается подбор значений неизвестных, и набор комментариев, имеющих управляющий характер (целевая установка и т.п.). Результатом работы синтезатора служат набор искомых значений неизвестных и список фактически использованных посылок. В зависимости от своего типа, синтезатор либо находит единственный пример допустимых значений неизвестных, либо перечисляет такие примеры. Пример синтезатора - оператор "верхняяоценка", реализующий утверждения вида $a \leq x$ для известной переменной a и неизвестной x, который перечисляет представляющие интерес (с учетом комментариев) константные верхние оценки x выражения a.

3. Аналогом задачи на преобразование является пакетный оператор, названный нормализатором. Он получает в качестве входных данных преобразуемый терм (выражение либо утверждение), список посылок, в предположении истинности которых выполняются преобразования, и набор комментариев, имеющих управляющий характер. Прием нормализатора выполняет некоторое преобразование его "условия". Это преобразование (в зависимости от предназначения нормализатора) не обязательно должно быть тождественным (например, при получении асимптотической оценки какого-либо выражения). В отличие от проверочного оператора и синтезатора, представляющих собой операторы ЛОСа, нормализатор является операторным выражением - его значением служит возникший после цепочки преобразований терм. Чтобы определить, какие из посылок фактически были использованы при его получении, в комментариях нормализатора используется информационный элемент "выводимо A", накапливающий такие посылки в наборе A. Пример нормализатора - операторное выражение "нормдробь", осуществляющее общую стандартизацию дробных выражений (отбрасывание знаменателя 1, деление дроби, деление на дробь, вынесение минуса из числителя либо знаменателя, и т.п.).

4. Аналогом задачи на исследование является пакетный оператор, названный анализатором. Он обрабатывает копию списка посылок текущей задачи Z, оформленную в виде вспомогательной задачи на исследование Z'. Анализатор не обращается для сканирования Z' к основной базе приемов, а использует свои собственные приемы. Так как их существенно меньше, то удается за сравнительно небольшое время получить значительное число следствий. Иэ этих следствий отбираются лишь ценные для внешней задачи Z, которые по окончании работы анализатора переносятся в ее список посылок. Отбор выполняется приемами анализатора, помечающими нужные посылки комментарием "внешвывод". Обращение к анализатору обеспечивается специальными приемами, указывающими лимит времени его работы. Передача отобранных утверждений происходит по исчерпании этого лимита. При получении особо важных следствий возможен немедленный обрыв работы. Входными данными обращения к анализатору служат: задача Z, максимальный уровень используемых приемов и целевая установка, передаваемая задаче Z'. Вспомогательная задача Z' создается самим анализатором. Он же реализует передачу отобранных утверждений в задачу Z.

Кроме указанных выше основных четырех типов пакетных операторов, используется еще один, вспомогательный. В фильтрах нескольких различных приемов иногда возникает необходимость применения одного и того же условия, представляющего

собой конъюнкцию либо дизъюнкцию большого (и возрастающего по мере обучения) числа описаний частных ситуаций. Здесь становится целесообразным создание реализованного на ГЕНОЛОГе пакетного оператора, выполняющего проверку этого условия либо определение значений его "выходных" переменных. Этот оператор, в отличие от перечисленных выше, относится к "структурному" уровню - он проверяет условие, относящееся не к объектам предметной области, а к логическим структурам данных, имеющимся в задаче. Хотя, в принципе, для задания операторов "структурного" уровня мог бы с равным успехом использоваться ЛОС (ведь он представляет собой логический язык для формулировки условий на текущую структуру данных), в решающих правилах приемов часто возникает необходимость задавать вид термов "предметного" уровня - и здесь существенное упрощение записи создает ГЕНОЛОГ. Пакетный оператор данного, пятого, типа, называется оператором фильтра.

Прием проверочного оператора

Прием проверочного оператора обычно имеет теорему вида $\forall x(A_1 \ \& \ \ldots \ \& \ A_n \rightarrow A_0)$. Здесь утверждение A_0 идентифицируется с проверяемым условием (оно само должно иметь вид, согласованный с шаблоном проверочного оператора). Применение приема заключается в проверке (с учетом указателей приема) утверждений $A_1, \ldots, A_n$ и выходе из проверочного оператора (как из программы на ЛОСе) по значению "истина"; выходной переменной оператора при этом присваивается список фактически использованных при проверке $A_1, \ldots, A_n$ посылок. Заголовок приема имеет вид "спуск(B)", где B - название проверочного оператора. Пример приема проверочного оператора - использование теоремы $\forall x(\text{число}(x) \ \& \ \neg(x = 0) \rightarrow \neg(-x = 0))$ для проверки отличия от нуля значения выражения, начинающегося со знака "минус". Заголовок данного проверочного оператора - "усмне0"; шаблон проверяемых им утверждений - $\neg(a = 0)$.

Прием синтезатора

Прием синтезатора имеет теорему вида: $\forall x(A_1 \ \& \ \ldots \ \& \ A_n \rightarrow A_0)$. Здесь утверждение A_0 идентифицируется с реализуемым условием; применение приема заключается в проверке либо реализации (с учетом указателей приема) утверждений $A_1, \ldots, A_n$ и выдаче найденных значений переменных; дополнительной выходной переменной оператора при этом присваивается список фактически использованных при реализации посылок. Заголовок приема имеет вид "значение(B)", где B - название синтезатора. Пример приема синтезатора - использование теоремы $\forall abc(0 < c \ \& \ a \leq b \ \& \ 0 \leq a \rightarrow a^c \leq b^c)$ для получения верхней оценки степенного выражения a^c, у которого показателем степени служит положительная константа, а основание степени неотрицательно. Название синтезатора - "верхняяоценка"; значение b определяется из антецедента $a \leq b$, представляющего собой рекурсивное обращение к тому же самому синтезатору. Шаблон реализуемых синтезатором утверждений - $x \leq y$, где переменная x - входное (известное) выражение; переменная y - выходная (не известная).

Прием нормализатора

Приемы нормализатора бывают следующих типов:

1. Если теорема представляет собой тождество $\forall x(A \to (B = C))$ либо эквивалентность $\forall x(A \to (B \leftrightarrow C))$, то она может быть использована для создания приема нормализатора, выполняющего замену слева направо либо справа налево. В первом случае заголовок приема имеет вид "замена(второйтерм N)"; во втором случае - "замена(первыйтерм N)". Здесь N - название нормализатора. Пример приема такого типа - использование теоремы $\forall_{abc}(b/(c/a) = ab/c)$ в нормализаторе "нормдробь" для исключения делений на дробное выражение.

2. Возможно создание нормализаторов, в которых предусмотрен разбор случаев при выполнении преобразований. Обычно потребность в таком разборе выявляется не сразу, а после некоторой цепочки преобразований. Разбор случаев инициируется специальным приемом, организующим откат к циклу преобразований исходного терма в различных подслучаях. В рамках рассмотрения отдельного подслучая возможна инициализация вложенного разбора случаев - без ограничений на глубину вложенности. По завершении рассмотрения всего дерева подслучаев формируется объединяющий их терм - с помощью логической конструкции "вариант($A\ t_1\ t_2$)"; этот терм и выдается как результат применения нормализатора. Примером нормализатора, допускающего указанный разбор случаев, является нормализатор вычисления пределов "нормпредел".

   Заметим, что при откатах, вызванных разбором случаев, теряются ранее проведенные вычисления. Однако, для нормализаторов, выполняющих сколь-нибудь трудоемкие вычисления, предусмотрено автоматическое сохранение результатов обращений к ним в специальном буфере. При каждом последующем обращении к нормализатору прежде всего проверяется наличие в буфере готового результата, и если есть, то он сразу и выдается. Это позволяет экономить при откатах, фактически извлекая из буфера готовые результаты там, где они не зависели от определяющего подслучай утверждения. Буферы, сохраняющие результаты нескольких последних обращений, используются и для некоторых других типов пакетных операторов; их применение позволило при программировании на ГЕНОЛОГе практически не заботиться о предотвращении дублирований, возникающих при обращении из различных приемов к одним и тем же вспомогательным вычислениям.

   Возвращаясь к разбору случаев в нормализаторах, укажем вид псевдотеорем, применяемых для инициализации этого разбора: $\forall_x(A \to (B =$ разборслучаев$(C)))$. Здесь дизъюнкция C определяет необходимые подслучаи; терм B идентифицируется с текущим преобразуемым термом либо (в зависимости от типа нормализатора) с его подтермом. Заголовок приема здесь по-прежнему имеет вид "замена(второйтерм N)", где N - название нормализатора, хотя в действительности прием не выполняет никакой замены. Пример приема данного типа - использование псевдотеоремы

   $\forall_{xfabc}(lim_{x\to b\backslash a} f(x) = 0 \to lim_{x\to b\backslash a} f(x)^c =$ разборслучаев $(0 < c\ \vee\ c < 0\ \vee\ c = 0))$

   при инициализации разбора случаев для знака показателя степени, если предел основания степени равен нулю.

3. Для устранения вложенных одинаковых ассоциативно - коммутативных операций f: $f(\ldots f(\ldots)\ldots) = f(\ldots\ldots\ldots)$ - служит прием нормализатора, имеющий заголовок "замена(спускоперандов B)", где B - название нормализатора. Теорема этого приема имеет вид "коммутативно(f)".

4. Для лексикографического упорядочения операндов коммутативной операции f в нормализаторе используется прием теоремы "коммутативно(f)", имеющий заголовок "замена(лексупорядочение B)", где B - заголовок нормализатора.

5. В тех случаях, когда требуется ввести, удалить либо изменить комментарий нормализатора, не предпринимая изменений его преобразуемого терма, используется прием с заголовком "замена(замечание N)", где N - заголовок нормализатора. Теорема такого приема имеет вид "длялюбого($x_1 \dots x_n$ если $A_1 \dots A_m$ то контекст(A_0))", причем идентификация начинается с усмотрения вхождения в преобразуемый терм терма A_0. Необходимые действия с комментариями определяются предназначенными для этого указателями приема.

Прием анализатора

Как и приемы вывода, приемы анализаторов имеют теорему вида $\forall x(A_1 \,\&\, \dots \,\&\, A_n \to A_0)$, при помощи которой осуществляется вывод следствия A_0; способ обработки утверждений $A_i (i = 1, \dots, n)$ при этом уточняется указателями приема (возможны непосредственная идентификация такого утверждения с одной из посылок; проверка его истинности при помощи проверочного оператора либо задачи на доказательство; использование для ввода вспомогательных обозначений, и т.п.). Заголовок приема - "внутрвывод(B)", где B - название анализатора.

Прием оператора фильтра

Прием оператора фильтра имеет псевдотеорему "блок" (на экран при просмотре описания приема вместо этой псевдотеоремы выводится текст концевого пункта оглавления, в котором расположены задающие формат оператора фильтра приемы справочников "блок" и "блокпроверок"; этот текст обычно напоминает смысл входных и выходных переменных оператора). Заголовок приема - "контекст(A)", где A - название оператора фильтра. При отсутствии выходных переменных оператор получает значение "истина", если выполнены условия на текущий контекст, определяемые фильтрами хотя бы одного из его приемов; при наличии выходных переменных - выдает их значения при срабатывании какого-либо из его приемов. В зависимости от типа оператора, при его работе, кроме входных переменных, могут быть доступны другие структуры данных (текущая задача и координата вхождения в нее - если обращение к оператору происходит из приема сканирования задачи, либо список комментариев внешнего пакетного оператора - если обращение имеет место из такого оператора, и т.п.).

11.2.8 Вычислительные пакеты

Рассмотренные выше пакетные операторы предназначены для работы с логическими структурами данных. Однако, ГЕНОЛОГ можно использовать и для создания программ, работающих с "обычными" техническими структурами данных - числами, массивами, таблицами и т.п. Эти программы, разумеется, тоже имеют своими источниками теоремы или группы теорем, так что естественным образом возникает задача компиляции их непосредственно с теоремного уровня.

Для задания обычных, нелогических вычислений на ГЕНОЛОГе используются так называемые вычислительные пакеты. Они компилируются в программу оператора

ЛОСа и состоят обычно из небольшого числа вычислительных "приемов" или "продукций".

В приемах логического характера четко разделены структурный и предметный уровни. Обрабатываемая информация представлена на структурном уровне, в виде формул, косвенным образом ссылающихся на объекты и отношения предметного уровня. Поэтому возникает необходимость в двухуровневой конструкции приемов: на предметном уровне записывается теорема, обеспечивающая возможность выполняемых приемом преобразований, а на структурном уровне записываются условия, при которых это преобразование представляется целесообразным. Но для вычислительного приема предметный и структурный уровни сливаются. Здесь обрабатываемая информация состоит из технических структур данных, которые однозначно определяют объекты и отношения между ними - являются, своего рода, именами этих объектов и отношений. Поэтому вся управляющая часть приема может быть полностью погружена в предметный уровень - в виде дополнительных посылок теоремы, определяющих целесообразность ее применения. Соответственно, приемы вычислительных пакетов имеют (за редкими исключениями, связанными с использованием комментариев) пустой список фильтров.

Выделим здесь два типа вычислительных пакетов - пакеты продукций и технические анализаторы.

Пакет продукций можно рассматривать как вычислительный аналог синтезатора. Прием его имеет заголовок "продукция(A)", где A - заголовок пакета. Если список термов и логических символов, задающих формат пакета продукций, имеет элемент "эквивалентно", то приемы этого пакета работают в режиме эквивалентных преобразований группы входных данных, приводя ее к такому виду, для которого значения выходных переменных вычисляется непосредственно. Такой режим аналогичен режиму работы пакетных нормализаторов. При отсутствии элемента "эквивалентно" каждый прием пакета продукций определяет значения выходных переменных за один шаг, как и у пакетного синтезатора. Разумеется, этот шаг может включать в себя рекурсивные обращения к вспомогательным вычислениям.

Технический анализатор выполняет разметку некоторой сложной структуры данных, снабжая ее элементы наборами значений параметров, характеризующих эти элементы либо связи между ними. Разметка осуществляется в режиме сканирования технической структуры данных. Список параметров определен форматом анализатора; хранятся они во вспомогательной структуре данных, организованной подобно анализируемой. Режим работы технического анализатора аналогичен режиму работы пакетного анализатора, так как представляет собой, по существу, процесс вывода следствий вплоть до полного их исчерпания. Прием технического анализатора имеет заголовок "знач(A)", где A - название анализатора.

Чтобы перейти от рекурсивного характера вычислений, свойственного логическим пакетным операторам, к вычислениям итерационным, в описаниях формата вычислительных пакетов используются указатели, регулирующие объединение программ различных приемов в общую программу пакета.

11.2.9 Простейшие приемы непосредственного усмотрения истинности либо ложности

Для усмотрения истинности утверждений вида $f(A)$, где f определяет один из базисных типов объектов (число; множество; функция; точка и т.п.), вводятся приемы, усматривающие этот тип для значений выражения A из анализа заголовка A. Псевдотеорема такого приема имеет вид "родобъекта(f)"; заголовок - "родобъекта".

Для усмотрения противоречивых указаний на тип объекта вводится также прием с заголовком "различны" (его псевдотеорема - та же, т.е. "родобъекта(f)").

11.2.10 Приемы справочников

Количество различных справочников, используемых в системе, приближается к сотне. Задание их приемов на ГЕНОЛОГе обычно весьма простое - фильтры и указатели приема здесь, как правило, отсутствуют (в таких ситуациях, для обеспечения работы компилятора ГЕНОЛОГа, приходится вводить какой-либо незначащий логический символ в качестве указателя - как правило, используется логический символ "пусто" либо "справка"). Для каждого нового справочника в компилятор ГЕНОЛОГа вводится свой (обычно коротенькая) программа трансляции - копируется с аналогичной программы другого справочника, с минимальными изменениями. Функционирование справочников достаточно подробно описано в информации о логических символах - названиях этих справочников (доступной, например, по F2 из редактора программ ЛОСа); названия справочников перечислены, например, в информации по заголовкам приемов ГЕНОЛОГа, доступной по Ctr-з (кир.) из просмотра описания какого-либо приема ГЕНОЛОГа. Поэтому мы ограничимся здесь лишь рассмотрением справочников, используемых для задания формата пакетных операторов. Приемы этих справочников создаются перед созданием самого пакетного оператора; после того, как они введены, можно переходить ко вводу и компиляции его приемов. В интерфейсе редактора приемов предусмотрены средства автоматической инициализации пакетного оператора.

Задание формата проверочных операторов

Формат проверочного оператора определяется справочником "легковидеть". Теорема приема этого справочника имеет вид "легковидеть($f(x_1 \dots x_n\ x_{n+1}\ x_{n+2}\ x_{n+3})$ уровень(N)T)". Здесь $f(x_1 \dots x_n\ x_{n+1}\ x_{n+2}\ x_{n+3})$ - вид обращения к оператору; T - шаблон проверяемого утверждения. $x_1, \dots, x_n$ - все свободные переменные утверждения T; Входной переменной x_{n+1} передается список утверждений, истинность которых может быть использована при проверке (посылок); переменной x_{n+2} - список комментариев. Выходной переменной x_{n+3} присваивается набор посылок, фактически использованных при усмотрении. N - число уровней срабатывания оператора f (при $N = 1$ указатель "уровень(N)" может отсутствовать).

Кроме обычных проверочных операторов, можно создавать операторы усиленной проверки. Их формат определяется справочником "проверка", аналогичным справочнику "легковидеть". Теорема приема этого справочника имеет вид "проверка($f(x_1 \dots x_n\ x_{n+1}\ x_{n+2}\ x_{n+3})$ уровень(N)T)", и далее - так же, как у обычного проверочного оператора.

Задание формата синтезаторов

Формат синтезатора определяется справочниками "синтезатор" и "значения". Оба этих справочника основаны на одной и той же псевдотеореме вида "синтезатор(f T вход($x_1 \dots x_n$) выход($y_1 \dots y_k$) $A_1 \dots A_m$)". Здесь f - заголовок синтезатора (логический символ); T - конъюнкция утверждений, связывающих значения входных и выходных переменных (T не имеет двух различных вхождений одной и той же переменной); $x_1, \dots, x_n$ - список входных переменных; $y_1, \dots, y_k$ - список выходных переменных. $A_1, \dots, A_m$ - информационные термы, уточняющие тип синтезатора (см. ниже). Обращение к синтезатору имеет вид $f(x_1 \dots x_n\ x_{n+1}\ x_{n+2}\ y_1 \dots\ y_k\ y_{k+1})$, где x_{n+1} - список посылок; x_{n+2} - список комментариев; y_{k+1} - список утверждений, использованных в качестве обоснования результата.

Используются следующие типы информационных термов A_i:

1. уровень(N). N есть число уровней срабатывания приемов синтезатора;

2. перечисление. Синтезатор определяет перечисление значений выходных переменных;

3. коррекцияпосылок. Перед обращением к синтезатору из приема сканирования задачи в комментариях синтезатора создается накопитель (коррекцияпосылок C). Набор C (вначале пустой) заполняется синтезатором тройками ($B_1\ B_2\ B_3$), где B_1 - набор дополнительных посылок, введенных для того, чтобы обеспечить непосредственное усмотрение принадлежности области допустимых значений входящих в результат обращения к синтезатору выражений; B_3 - список исходных посылок, из которых были выведены посылки списка B_1; B_2 - либо 0, либо подмножество списка B_3, эквивалентное списку B_1 в предположении истинности остальных утверждений списка B_3.

Задание формата нормализаторов

Формат нормализатора определяется справочником "быстрпреобр". Псевдотеорема приема этого справочника имеет вид "быстрпреобр(f $U_1 \dots\ U_n$)", где f - заголовок нормализатора; $U_1, \dots, U_n$ - указатели типа нормализатора (см. ниже). Кроме справочника "быстрпреобр", для нормализаторов общей стандартизации термов с заголовком B создается также справочник "нормализатор". Псевдотеорема приема такого справочника имеет вид "нормализатор(B f)". Данный справочник нужен для определения названия f нормализатора по заголовку B преобразуемого терма. Заметим, что такое название всегда имеет вид "нормB"("нормплюс", "нормминус", нормстепень" и т.д.). Кроме приема справочника "нормализатор", обычно вводится также прием справочника "ключоператора" с псевдотеоремой "ключоператора(f B)" - он решает обратную задачу определения B по f. Заголовки приемов всех трех справочников совпадают с названием этих справочников.

Используются следующие указатели типа нормализатора:

1. "уровень(A)" - A есть логический символ, равный числу различных уровней срабатывания приемов нормализатора. Отсутствие такого терма означает, что все приемы срабатывают на одном и том же уровне (рекомендуется вводить хотя бы два различных уровня).

2. "списокпосылок" - обращение к нормализатору имеет вид $f(A_1\ A_2\ A_3)$, где A_1 - преобразуемый терм; A_2 - набор посылок, в предположении истинности которых выполняются преобразования; A_3 - набор комментариев, включающий, в частности, набор (выводимо B); B - накопитель утверждений списка A_2, фактически использованных в процессе преобразований. Если элемент "списокпосылок" отсутствует, то обращение к нормализатору имеет вид $f(A)$, где A - преобразуемый терм, либо (см. ниже случай указателя "неизвестные") вид $f(A\ B)$.

3. "корень" - приемы применяются только к корневому вхождению. Это - наиболее распространенный тип нормализаторов (в частности, к нему относятся все нормализаторы общей стандартизации). Во-первых, здесь происходит ускорение поиска приема за счет рассмотрения единственной "точки привязки" - корня преобразуемого терма. Во-вторых, упрощается ввод сопровождающих утверждений для указания области допустимых значений новых термов, возникающих в процессе преобразований (фактически такое сопровождение пока предусмотрено только для нормализаторов "корневого" типа, и при работе с числовыми выражениями, за редкими исключениями, используются только они). Недостаток таких нормализаторов - затруднения при усмотрении и реализации преобразований некорневых подтермов. Обычно эти затруднения преодолеваются за счет обработки соответствующими корневыми нормализаторами всех операций, упомянутых в заменяющем терме теоремы приема, а также перехода от тождества $A = B$ к тождеству вида $g(\ldots A\ldots) = g(\ldots B\ldots)$, если корневой контекст $g(\ldots\ldots)$ некорневого преобразования $A = B$ прослеживается однозначно либо с малым числом вариантов.

4. "1" - результат выдается после однократного применения любого приема.

5. "неизвестные" - обращение к нормализатору имеет вид $f(A_1\ A_3)$ (при отсутствии указателя "списокпосылок") либо вид $f(A_1\ A_2\ A_3)$; наличие в A_3 элемента (неизвестные $x_1 \ldots x_n$) означает, что переменные $x_1, \ldots, x_n$ суть неизвестные той внешней задачи, из которой произошло обращение к нормализатору, причем если эта задача - на описание, то преобразуемый терм относится к ее условиям. Компилятор ГЕНОЛОГа автоматически обеспечивает создание такого элемента в наборе A_3 при обращениях к нормализаторам данного типа.

6. "стандупорядочение" - перед выдачей результата осуществляется обращение к операторному выражению "стандупорядочение", обеспечивающему стандартное упорядочение операндов операций - там, где допустимы простейшие типы перестановки операндов.

7. "разборслучаев" - предусмотрена нормализация с откатами для разбора случаев.

8. "глуб" - в комментариях нормализатора формируется элемент (глуб N), у которого N есть число внешних обращений к данному нормализатору в цепочке обращений к нормализаторам.

9. "группировка" - нормализатор, работающий в режиме отбора нескольких наиболее упрощающих выражение преобразований непересекающихся его подтермов и последующей их одновременной реализации.

10. "контрольнормализации" - используется буфер "контрольнормализации", сохраняющий результаты нескольких последних обращений к нормализатору. При трассировке "по шагам решения задачи" обращения к таким нормализаторам выводятся на экран и происходит показ их пошаговых действий. Использование буфера позволяет при программировании не слишком заботиться о экономии на повторных обращениях к обработке одного и того же терма, происходящих из различных приемов - такие обращения обычно идут друг за другом достаточно компактной группой, и размеров буфера (50 позиций) оказывается вполне достаточно для их отслеживания.

11. "коррекцияпосылок" - перед обращением к нормализатору из приема сканирования задачи в комментариях нормализатора создается накопитель (коррекцияпосылок A). Набор A (вначале пустой) заполняется тройками (B_1 B_2 B_3), где B_1 - набор дополнительных посылок, введенных для сопровождения результата обращения к нормализатору. Эти посылки позволяют непосредственно усматривать принадлежность области допустимых значений (о.д.з.) подвыражений результата. B_3 - список исходных посылок, из которых были выведены посылки списка B_1; B_2 - либо 0, либо подмножество списка B_3, эквивалентное списку B_1 в предположении истинности остальных утверждений списка B_3. Использование указанного накопителя совершенно необходимо в тех случаях, когда предметная область требует аккуратного сопровождения термов по о.д.з. (например, в элементарной алгебре), причем выполняемые нормализатором преобразования способны привести к появлению термов с неочевидным в исходном контексте выполнением условий на о.д.з.

Задание формата анализаторов

Формат анализатора определяется справочником "анализатор". Псевдотеорема приема этого справочника имеет вид "анализатор(f S)", где f - логический символ, являющийся названием анализатора, S - либо отсутствует, либо представляет собой один из логических символов "корень", "полный". Если S отсутствует, то анализатор сканирует все вхождения логических символов в посылки своей вспомогательной задачи, обращаясь на каждом уровне сканирования только к посылкам, вес которых не превосходит этого уровня. Если S есть символ "корень", то сканирование ограничивается лишь корневыми точками посылок. Наконец, если S есть символ "полный", то сканирование ведется по корневым точкам, а веса посылок при сканировании игнорируются.

Задание формата операторов фильтра

Формат оператора фильтра определяется справочниками "блок" и "блокпроверок". Псевдотеорема приема этих справочников - общая; она имеет вид "блок(A вход(B_1 $\ldots$ B_n) выход(B_{n+1} $\ldots$ B_m) C_1 $\ldots$ C_k)". Здесь A - заголовок оператора; обращение к оператору имеет вид $A(y_1 \ldots y_p\ x_1 \ldots x_m)$. Каждое B_i - указатель типа данных соответствующей входной либо выходной переменной x_i (лог. символы "терм", "вхождение", "логсимвол", "переменная"); $y_1, \ldots, y_p$ - дополнительные входные переменные, значения которых автоматически определяются (как это указано ниже) из текущего контекста. Если выходных переменных нет, то терм "выход($B_{n+1} \ldots B_m$)" отсутствует. $C_1, \ldots, C_k$ - указатели типа оператора структурного уровня.

Концевой пункт оглавления базы приемов, в котором хранятся указанные два приема справочников "блок" и "блокпроверок", располагается первым в подразделе, охватывающем приемы заданного оператора фильтра A. Текст этого пункта для удобства программирования автоматически выводится на экран вместо псевдотеоремы "блок" всех приемов оператора A, и его следует составлять таким образом, чтобы из него извлекалась информация о назначении вход-выходных переменных x_i.

Указатели типа оператора фильтра таковы:

1. Указатель текущего контекста, в котором происходит обращение к оператору - лог.символ "пусто" (число p равно 0), либо "решить" (текущий контекст - сканирование задачи, и тогда $p = 5$ - для программных переменных x1 - x5, определенных при сканировании), либо "замена" (текущий контекст - пакетный нормализатор, и тогда $p = 3$ - для программных переменных x1 - x3 этого нормализатора).

2. "перечисление" - указатель на наличие режима перечисления без повторений выходных наборов.

11.3 Типы фильтров приемов

Логический язык, на котором формулируются фильтры приема, относится к уровню "структур данных" - в этом он весьма близок к ЛОСу. Отличие от ЛОСа состоит в том, что при задании фильтра приходится ссылаться на термы задачи через посредство "теоремных" переменных, идентифицированных с этими термами (иногда - с переменными либо логическими символами); при использовании ЛОСа такая ссылка осуществляется непосредственно, через программные переменные, которым заранее присваивается необходимое значение. С некоторым приближением, можно считать, что язык для задания фильтров - это "ЛОС в теоремных переменных". Многие операторы и операторные выражения, используемые для записи фильтров, попросту копируют соответствующие операторы и операторные выражения ЛОСа; иногда при этом удается опускать ряд их входных параметров, определяемых в контексте "по умолчанию".

При задании фильтров можно использовать, наряду с логическими связками, также и квантор существования (квантор общности сводится к нему с помощью отрицаний). Однако, этот квантор представляется здесь с помощью специальной конструкции "контекст(...)" (см. подробнее подраздел "Логические связки и квантор существования"), в которой опущена кванторная приставка - по умолчанию к ней относятся все переменные, встречающиеся внутри данной конструкции, не определенные к моменту ее рассмотрения внешним образом (в частности, не входящие в теорему). Такое соглашение укорачивает записи и практически не усложняет понимания их смысла. Другое важное отличие от обычного квантора существования состоит в том, что внутри конструкции "контекст(...)" могут использоваться условия, идентифицирующие термы задачи с некоторыми заданными термами, и в этом случае наряду с "чисто логическими" записями здесь будут встречаться технические указатели, уточняющие способ идентификации.

Как правило, порядок размещения фильтров в описании приема несущественен - компилятор размещает их программы в программе приема "по-своему", стараясь

обеспечить определяемое ими отсечение как можно раньше. Однако, существует возможность влиять на размещение программы фильтра в программе приема (иногда это все же позволяет уменьшить "холостой ход" приема) - путем использования специальных сопровождающих фильтр пометок либо путем обычного их упорядочения (если в программе необходимые для выполнения проверок переменные определяются для нескольких фильтров одновременно, то их программные фрагменты размещаются в программе приема в том же порядке, что и сами фильтры в описании приема).

В большинстве случаев компилятор автоматически определяет тип значения переменной, используемой в фильтре (терм, либо вхождение, либо символ, и т.п.) - в особых случаях предусмотрено явное указание такого такого типа.

Часть встречающихся в фильтрах подвыражений называем, по аналогии с ЛОСом, операторными выражениями. Иногда они действительно оказываются тождественны операторным выражениям ЛОСа. Кроме операторных выражений, в фильтрах могут использоваться также термы предметного уровня. В отличие от операторных выражений, составленных из операций над объектами структурного уровня (термами, списками, задачами и т.п.), термы предметного уровня образованы операциями и отношениями над объектами рассматриваемой предметной области (например, числами, точками, функциями, и т.п.). Термы предметного уровня, встречающиеся в фильтрах, будем называть также термами "в теоремных переменных", так как они аналогичны термам теоремы приема. При описании фильтров уточняется, в каких случаях допустимо использование операторного выражения, а в каких - терма "в теоремных переменных".

11.3.1 Логические связки и квантор существования

В фильтрах можно использовать обычные логические связки "не(A)", "и($A_1 \dots A_n$)", "или($A_1 \dots A_n$)", а также конструкцию "альтернатива(A_1 A_2 A_3)" - в случае истинного A_1 ее истинность совпадает с истинностью A_2, а иначе - с истинностью A_3.

Роль квантора существования при задании фильтров играет конструкция "контекст($A_1 \dots A_n$)". Термы A_i в этой конструкции бывают трех типов:

а) Идентифицирующие термы - они аналогичны операторам ЛОСа, имеющим выходные переменные (в том числе перечисляющим) и определяют значения некоторых новых, до этого не определенных, переменных, используемых в других термах A_j того же списка. Вне данной конструкции "контекст(...)" эти переменные будут считаться не определенными. Изначально определенными при рассмотрении фильтров считаются все переменные, входящие в теорему, а также некоторые переменные, дополнительно определяемые в специальных указателях приема;

б) Проверочные термы - они формулируют условия на значения ранее определенных переменных и представляют собой произвольные правильно построенные фильтры;

в) Указатели идентификации - они уточняют способо применения идентифицирующих термов и идентичны аналогичным указателям для описания всего приема.

Фильтр "контекст($A_1 \dots A_n$)" выражает условие существования значений переменных, определяемых его идентифицирующими термами A_i, при которых истинны все проверочные термы. Порядок размещения термов A_i в списке, в общем, произвольный, за исключением одного ограничения: после первого же идентифицирующего

терма с заголовком "вид" либо "подтерм" либо "усм" могут идти только такие идентифицирующие термы, которые имеют один из данных трех заголовков. Допускаются вложенные (с произвольной глубиной вложения) конструкции "контекст(...)".

В качестве простого примера использования фильтра "контекст(...)" рассмотрим следующую ситуацию. Пусть в теореме встречается переменная x, идентифицированная с неизвестной задачи на описание, и требуется ввести фильтр, разрешающий применение приема лишь тогда, когда эта неизвестная встречается в каком-либо неравенстве из условий задачи. Этот фильтр может быть записан как "контекст(условие(A)заголовок(A меньше меньшеилиравно) входит(x A))". Здесь A - какая-либо переменная, не встречающаяся в теореме приема. Идентифицирующий терм "условие(A)" будет перечислять все условия задачи; проверочный терм "заголовок(A меньше меньшеилиравно)" - отбирать из них лишь те, заголовком которых служит отношение "меньше" либо "меньшеилиравно", а проверочный терм "входит(x A)" - проверять вхождение в условие переменной, идентифицированной с x.

Квантор общности в фильтрах не предусмотрен - он выражается через квантор существования и отрицание.

11.3.2 Равенство объектов

Условие "равно(A B)" используется для указания на равенство значений операторных выражений A и B. Если нужно сравнить два терма P, Q "предметного уровня", т.е. заданных через переменные теоремы, то в качестве A, B следует брать операторные выражения "терм(P)", "терм(Q)".

Для сокращенной записи используется также обозначение "равно(A $B_1 \dots B_n$)", означающее, что значение операторного выражения A равно хотя бы одному из значений операторных выражений $B_1, \dots, B_n$.

11.3.3 Сравнение числовых характеристик термов

Если требуется указать, что некоторый терм предметного уровня A имеет значение числовой харктеристики $f(A)$ меньшим, чем любой из термов $B_1, \dots, B_n$, то используется запись "упрощение(x_i и(A $B_1 \dots B_n$) $f(x_i)$)". Здесь x_i - новая переменная, используемая в определяющем характеристику f операторном выражении $f(x_i)$.

Если терм A должен иметь значение числовой характеристики $f(A)$ меньшим, чем хотя бы один из термов $B_1, \dots, B_n$, то используется запись "упрощение(x_i или(A $B_1 \dots$ B_n) $f(x_i)$)".

Например, если требуется применять прием лишь в том случае, когда число неизвестных задачи, встречающихся в терме, идентифицированном с переменной a, меньше числа неизвестных в любом из термов, идентифицированных с переменными b, c, d, то используется фильтр "упрощение(x и(a b c d) число(неизвестная(y) входит(y x))".

11.3.4 Ограничения на задачу

Текущий уровень задачи

Для указания допустимых значений $A_1, \dots, A_n$ текущего уровня сканирования текущей задачи используется фильтр "уровень($A_1 \dots A_n$)". При вводе описания приема,

реализуемого в режиме сканирования задачи, обязательно должен быть указан такой фильтр (иначе компилятор сразу будет выдавать отказ).

Цели задачи

Фильтр "цель(A)" означает, что текущая задача имеет цель A. Здесь A - либо логический символ, либо операторное выражение, определяющее цель. Если посредством идентифицирующих термов либо операторных выражений выделена некоторая другая задача Z, то для указания на то, что она имеет цель A, применяется запись "цель(Z)".

Фильтр "цели($A_1 \dots A_n$)" означает, что текущая задача не имеет цели, заголовок которой не входил бы в список $A_1, \dots, A_n$.

Тип задачи

Фильтр "тип(A)" означает, что текущая задача имеет тип A. - логический символ либо операторное выражение. Фильтр "тип(Z A)" означает, что задача Z (введенная при помощи идентифицирующих термов либо операторных выражений) имеет тип A.

Текущая задача - корневая

Для указания на то, что текущая задача - корневая, т.е. выбрана непосредственно из задачника, используется фильтр "исходнаязадача". Он помогает в тех случаях, когда для корневой задачи нужно разрешить применение различного рода усиленных средств.

Неизвестные задачи

Чтобы указать, что текущая задача имеет не более одной неизвестной, используется фильтр "неизвестные(1)". В случае $N > 1$ фильтр "неизвестные(N)" применяется несколько иначе - он указывает, что текущая задача имеет не менее N неизвестных.

Раздел, к которому относится условие задачи на доказательство либо на преобразование

Фильтр "раздел(A)", где A - название раздела, используется, если нужно проверить, что условие текущей задачи на преобразование либо на доказательство содержит хотя бы один логический символ, относящийся к разделу A либо к подразделу раздела A.

Проверка наличия посылок специального вида

Во многих предметных областях бывает полезно вводить в процессе решения задач фиктивные посылки вида "актив(A)", где A - некоторое выражение, которое полезно рассмотреть в задаче, но которое пока не встречается в других ее термах. Такие посылки также упрощают обнаружение в задаче выражений специального вида и часто используются для инициализации применения приемов. Например, в

геометрических задачах посылки "актив(расстояние(AB))", "актив(прямая(AB))", "актив(окружность(AB))" и т.п. выделяют для рассмотрения при сканировании различные элементы чертежа. Фильтр "См($A_1 \dots A_n$)" используется для указания на то, что задача имеет посылки "актив(A_1)", ..., "актив(A_n)".

11.3.5 Ограничения на термы, переменные и логические символы

Простейшие свойства термов и переменных

1. Фильтр "входит(X A)" означает, что переменная либо логический символ, заданные операторным выражением X, входят в терм, заданный операторным выражением A.

2. Фильтр "переменная(X)" означает, что переменная X идентифицирована с переменной.

3. Фильтр "константа(A)" означает, что операторное выражение A задает терм без свободных переменных.

4. Фильтр "вхождениетерма(A B)" означает, что терм B входит в терм A. Здесь A, B - не операторные выражения, а термы предметного уровня, записанные непосредственно в "теоремных" переменных.

5. Фильтр "короче(A B)" означает, что терм, заданный операторным выражением A, короче терма, заданного операторным выражением B. Под длиной терма понимается число вхождений в него логических символов и символов переменных.

6. Фильтр "родобъекта(A B)" означает, что терм, определенный операторным выражением A, имеет своим заголовком операцию, тип значения которой определяется логическим символом B.

7. Фильтр "род(A B)" представляет собой усиленную версию фильтра "родобъекта", использующую при проверке не только заголовок терма A, но и посылки текущего контекста.

8. Фильтр "подобныетермы(A B)" означает, что термы, определенные операторными выражениями A и B, получены друг из друга изменением порядка операндов в коммутативных операциях.

9. Фильтр "связка(A X)" означает, что переменная, определенная операторным выражением X, входит в связывающую приставку терма, определенного операторным выражением A.

Заголовки термов

Фильтр "заголовок(A $B_1 \dots B_n$)" означает, что терм, определенный операторным выражением A, имеет своим заголовком один из логических символов, определенных операторными выражениями $B_1, \dots, B_n$.

Фильтры "первыйсимвол(A B)", "второйсимвол(A B)", "предпоследсимвол(A B)", "последнийсимвол(A B)" аналогичны одноименным операторам ЛОСа; здесь операторное выражение A задает некоторый терм, а операторное выражение B - логический символ.

Характеризация терма либо переменной по отношению к задаче

1. Фильтр "неизвестная(x A)" означает, что переменная x идентифицирована с неизвестной задачи A. Обычно A не указывается - тогда по умолчанию рассматривается текущая задача.

2. Фильтр "параметр(x A)" означает, что переменная x идентифицирована с несущественной неизвестной задачи на описание A (при отсутствии A - текущей задачи). Несущественные неизвестные выделяются целью "параметры $x_1 \dots x_n$"; они представляют собой подмножество неизвестных задачи. При решении задачи требуется лишь установить существование удовлетворяющих условиям значений несущественных неизвестных, не находя полного их описания и не включая найденные значения в ответ.

3. Фильтр "известно(T A)" означает, что операторное выражение T идентифицируется с термом, не содержащим неизвестных задачи A (при отсутствии A - текущей задачи).

4. Фильтр "симметрично(T_1 T_2 A)" означает, что термы T_1, T_2 входят симметричным образом в список посылок (при A = "списокпосылок") либо в условия (при A = "списокусловий") текущей задачи, либо в текущий терм задачи или нормализатора (при A = "корень").

5. Фильтр "независит(A)" означает, что терм, заданный операторным выражением A, не содержит переменных, указанных в цели "независит $x_1 \dots x_n$" текущей задачи на описание. Такая цель указывает переменные, которые не должны иметь свободных вхождений в ответ задачи.

6. Фильтр "обозначения(A)" означает, что терм, заданный операторным выражением A, содержит дополнительные переменные, возникшие при выводе в посылках и зарегистрированные в комментарии к посылкам "новаяпеременная ...". Одним из источников появления таких переменных является прием, заменяющий в посылках задачи утверждение о существовании объектов, удовлетворяющих некоторому условию P, на само условие P, в котором данные объекты переобозначены на новые переменные.

Алгебраические выражения

В ГЕНОЛОГе используются не только фильтры общего характера, но и множество специальных фильтров, связанных с различными понятиями конкретных предметных областей. Разумеется, по мере освоения новых разделов приходится пополнять запас типов таких фильтров и, соответственно, расширять компилятор ГЕНОЛОГа. Однако, эта процедура достаточно проста - она сводится к реализации на ЛОСе оператора для нового фильтра и вводу коротенькой программы приема справочника "блокпроверок", обеспечивающего создание обращений к этому оператору для проверки фильтра.

1. Фильтр "линейно($A\ X_1 \dots X_n$)" означает, что операторное выражение A определяет алгебраическое выражение, которое после раскрывания скобок становится линейным относительно переменных, определяемых операторными выражениями $X_1, \dots, X_n$.

2. Фильтр "линейноеуравнение($A\ X_1 \dots X_n$)" означает, что операторное выражение A определяет алгебраическое уравнение, которое после раскрывания скобок становится линейным относительно переменных, определяемых операторными выражениями $X_1, \dots, X_n$.

11.3.6 Ограничения на вхождения

Символы, расположенные по заданному вхождению

Чтобы указать, что на вхождении, определенном операторным выражением t, расположен один из логических символов списка $A_1, \dots, A_n$, используется фильтр "символ($t\ A_1 \dots A_n$)". Этот же фильтр можно использовать, если t определяет вхождение в набор, не являющийся термом, а $A_1, \dots, A_n$ - операторные выражения, определяющие объекты, один из которых должен находиться на заданном вхождении.

Взаимное расположение двух вхождений

1. Для указания на то, что вхождение, определенное операторным выражением t_1, расположено внутри вхождения, определенного операторным выражением t_2, используется фильтр "подчинено($t_1\ t_2$". Здесь допускается совпадение вхождений.

2. Чтобы указать, что вхождение t_1 расположено не левее вхождения t_2, применяется фильтр "постпозиция($t_1\ t_2$)".

3. Фильтр "алгебрвхождение($t_1\ t_2$)" означает, что вхождение t_2 расположено внутри вхождения t_1 (или совпадает с ним) и достижимо из него только через операции "плюс", "минус", "умножение", "дробь", "модуль", равно", меньше", "меньшеилиравно", а также через основания степеней.

4. Фильтр "обобщмножитель($t_1\ t_2$)" означает, что вхождение t_2 расположено внутри вхождения t_1 (или совпадает с ним) и достижимо из него только через операции "минус", "умножение", "дробь", "модуль", а также через основания степеней.

Свободные переменные

Фильтр "свобоперанд(t)" означает, что операторное выражение t определяет вхождение подтерма, каждая свободная переменная которого свободна во всем терме.

11.3.7 Ограничения на наборы

1. Фильтр "входит(t набор($A_1 \dots A_n$))" означает, что объект, определяемый операторным выражением t, входит в набор, элементы которого задаются операторными выражениями $A_1, \dots, A_n$.

2. Фильтр "длинатекста(A B)" означает, что длина набора, определяемого операторным выражением A, равна символьному числу - значению операторного выражения B.

3. Фильтр "длинаменее(A B)" означает, что длина набора, определяемого операторным выражением A, меньше символьного числа - значения операторного выражения B.

11.3.8 Ограничения на новые термы, вводимые приемом

1. Фильтр "длина(A)" означает, что применение приема тождественной либо эквивалентной замены привело к получению более простого в смысле оператора A терма. Оператор A, реализованный на ЛОСе, должен иметь 5 входных переменных: x1 - задача; x2,x3,x4 - вхождение в эту задачу заменяемого терма; x5 - заменяющий терм. Истинное значение оператора означает наличие упрощения. Допускается отсутствие явного указания A, и тогда по умолчанию берется оператор "стандартизация". Возможно использование фильтра "длина(A)" и в случае нормализаторов. Тогда оператор A имеет уже 4 входных переменных: x1, x2 - заменяемый и заменяющий термы; x3 - список посылок нормализатора; x4 - список его комментариев.

2. Фильтр "контрольодз" означает, что все подвыражения заменяющего терма (в приеме замены либо в приеме нормализатора) удовлетворяют условиям на о.д.з.

3. Фильтр "контрольвывода" означает, что новое утверждение, заносимое в список посылок либо условий, не содержит выражений, до этого отсутствовавших в задаче.

11.3.9 Ограничение на способ идентификации

Если прием замены основан на тождестве либо эквивалентности с заменяемой частью $F(t_1 \ldots t_n)$, где F - ассоциативно-коммутативный символ, то фильтр "полный" означает, что идентификации с $t_1, \ldots, t_n$ подлежат все операнды рассматриваемого в задаче вхождения операции F.

11.3.10 Числовые предикаты

1. Для указания того, что номер логического символа либо переменной A меньше номера логического символа либо переменной B, используется фильтр "менее(A B)".

2. Если операторные выражения A, B задают десятичные числа, то для сравнения этих чисел используется фильтр "меньше(A B)".

3. Фильтры "десчисло(A)", "целое(A)", "натуральное(A)" выражают условия на то, что значением операторного выражения A служит терм - десятичная запись числа (соответственно, целого числа и натурального числа).

11.3.11 Учет комментариев задачи либо пакетного оператора

Для задания комментария в приводимых ниже фильтрах используется набор (A B_1 $\ldots$ B_n), где A - логический символ, являющийся заголовком комментария; $B_1, \ldots, B_n$ - операторные выражения, определяющие элементы комментария. Для B_i имеются следующие возможности:

а) B_i - логический символ либо терм C в теоремных переменных - тогда данный разряд комментария представлен, соответственно, как логический символ либо как терм. Если логический символ C нужно преобразовать к формату терма, то в качестве B_i берется запись "терм(C)".

б) B_i - служебное слово "теквхожд" (подтерм, идентифицируемый с заменяемым термом) либо "корень" (текущий терм задачи - условие либо посылка) либо "результат" (заменяющий терм приема).

в) B_i - указатель вхождения в теорему "фикс($i_1 \ldots i_k$)". Здесь i_1 - номер антецедента (начиная с 1) либо указатель на консеквент (0), в котором находится вхождение; i_2 - номер операнда выделенного антецедента либо консеквента, к которому следует переходить на втором шаге, и т.д. - вплоть до перехода к требуемому вхождению. Соответствующий B_i разряд комментария есть подтерм по найденному вхождению.

г) Если B_i имеет вид "вхождение(C)", то операторное выражение C интерпретируется как задающее вхождение; если B_i имеет вид "логсимвол(C)" либо "переменная(C)", то C интерпретируется как определяющее соответствующий символ (а не терм).

д) Если B_i имеет вид "набор($C_1 \ldots C_m$)", то оно определяет набор значений операторных выражений $C_1, \ldots, C_m$.

При работе с комментариями используются фильтры следующих типов:

1. Фильтр "комментусловия(A $B_1 \ldots B_n$)"означает, что текущий терм задачи (для приема замены - тот, который содержит заменяемый подтерм) не имеет комментария (A $B_1 \ldots B_n$). В зависимости от того, где расположен в задаче текущий терм, здесь имеется в виду комментарий посылки либо условия.

2. Фильтр "коммент(A $B_1 \ldots B_n$)" означает, что текущая задача либо (в случае приема пакетного оператора) пакетный оператор не имеет комментария (A B_1 $\ldots$ B_n). В случае задачи на исследование здесь рассматриваются общие комментарии к посылкам.

3. Фильтр "комментпосылки(i A $B_1 \ldots B_n$)" означает, что терм задачи, идентифицированный с i-м антецедентом теоремы, не имеет комментария (A $B_1 \ldots B_n$). Вместо номера антецедента (номер здесь символьный) i может быть операторным выражением, определяющим данный терм задачи, либо логическим символом "теквхожд", указывающим терм задачи, содержащий точку привязки.

4. Фильтр "комментпосылок(A $B_1 \ldots B_n$)" означает, что текущая задача не имеет комментария к посылкам (A $B_1 \ldots B_n$).

11.3.12 Ограничения на точку привязки

Под точкой привязки приема понимается то текущее вхождение в терм задачи либо пакетного оператора, рассмотрение которого инициализирует применение приема.

1. Фильтры "условие" и "посылка" означают, соответственно, что точка привязки приема располагается в условии либо в посылке текущей задачи.

2. Фильтр "внешзнак($A_1 \dots A_n$)" означает, что заменяемый подтерм расположен только внутри таких термов, заголовки которых принадлежат списку логических символов $A_1, \dots, A_n$.

3. Фильтр "корень" означает, что заменяемый подтерм совпадает со всем рассматриваемым термом (является "корневым" его подтермом).

4. Фильтр "отрицание" означает, что заменяемый подтерм A - корневой либо находится под корневым отрицанием, т.е. весь текущий терм имеет вид A либо "не(A)".

5. Фильтр "отр" означает, что заменяемый подтерм находится под корневым отрицанием, т.е. текущий терм задачи имеет вид "не(A)".

6. Фильтр "неизвтермы" означает, что точка привязки расположена внутри терма задачи на описание либо исследование, контролируемого при выделении повторных вхождений неизвестных. При наличии уравнений с неизвестными этот фильтр отсекает рассмотрение условий, не являющихся уравнениями.

7. Фильтр "внешоператор(i B)" означает, что i - й внешний стэковый кадр (по цепочке кадров глобального стэка - источников его текущего кадра) является кадром оператора либо операторного выражения с заголовком B, либо задачи типа B.

8. Фильтр "блокнеравенства" используется в тех случаях, когда прием выполняет преобразования, направленные на решение неравенства, и нужно проверить целесообразность таких действий в контексте задачи (например, заблокировать решение неравенства при наличии уравнений с теми же неизвестными).

11.3.13 Обращение к проверочному оператору из фильтра

Иногда бывает необходимо перед применением приема убедиться в том, что некоторое утверждение предметного уровня не является (реже - что является) очевидным в контексте задачи. Например, это может понадобиться для предотвращения вывода малоинформативных следствий. В этих случаях используется фильтр "легковидеть (A B)", где A - проверяемое утверждение, сформулированное "в теоремных переменных". B - терм "посылки(C)", определяющий дополнительные посылки C; этот терм может в фильтре отсутствовать. Фильтр используется только для таких A, которые получаются подстановкой в "шаблон" какого-либо проверочного оператора.

11.3.14 Идентифицирующие термы

Идентифицирующие термы используются в фильтрах "контекст(...)" и в некоторых специальных операторных выражениях. Они представляют собой аналог операторов, имеющих выходные переменные, и позволяют доопределять ("идентифицировать") значения новых переменных, используемых в том же фильтре.

Определение вида термов путем их идентификации с заданными термами предметного уровня

Если требуется выполнить идентификацию терма T, определенного операторным выражением A, с заданным термом B предметного уровня, то используется идентифицирующий терм "вид(A B)". Терм B может содержать новые (не встречающиеся в теореме либо во внешних фильтрах "контекст(...)") переменные, и после идентификации этим переменным оказываются сопоставлены некоторые термы предметного уровня. Такие переменные могут встречаться в фильтрах, относящихся к той же конструкции "контекст(...)", что и "вид(A B)", и там их значения уже будут определены.

Фильтры, использующие результаты идентификации с помощью терма "вид(A B)", могут быть расположены как после данного терма, так и до него - транслятор автоматически переупорядочивает проверки. Требуется лишь, чтобы группа идентифицирующих термов "вид(...)", "подтерм(...)" и "усм(...)" (последние два типа таких термов описываются ниже) завершала список идентифицирующих термов внутри фильтра "контекст(...)". Фильтры после данной группы располагаться могут. Эта группа обрабатывается компилятором отдельно; возникающая для нее программа на ЛОСе выполняет идентификацию всей группы одновременно, учитывая имеющиеся связи между ее элементами.

В некоторых случаях для определения целесообразности применения приема замены бывает необходимо уточнить вид надтерма заменяемого терма. Чтобы идентифицировать этот надтерм с заданным термом A, в котором собственно заменяемый терм обозначен посредством служебного логического символа "теквхожд", применяется идентифицирующий терм "подтерм(A)". Как и в случае идентифицирующего терма "вид(A_1 A_2)", внутри A могут находиться новые переменные, значение которых будет определяться при идентификации.

Если требуется уточнить вид надтерма не заменяемого терма, а терма, расположенного по какому-либо другому вхождению v, то в идентифицирующем терме "подтерм(A)" вместо логического символа "теквхожд" используется терм "теквхожд (B)", где B определяет вхождение v одним из следующих способов:

а) B - переменная x, имевшая в ранее идентифицированной части единственное вхождение;

б) B - указатель вхождения в теорему "фикс(...)" (см. ниже подраздел "Операторные выражения").

Использование идентифицирующих операторов

Для косвенной идентификации различных часто встречаемых отношений между объектами предметной области создаются специальные операторы ЛОСа, называемые

идентифицирующими операторами. Например, в геометрии введены: оператор, перечисляющий все точки, для которых в контексте усматривается их принадлежность заданной прямой; оператор, перечисляющий все прямые, для которых усматривается принадлежность им заданной точки; оператор, перечисляющий все прямые, для которых усматривается их перпендикулярность заданной прямой, и т.д. Эти операторы могут использовать простейшие логические переходы, извлекая новые объекты перечисления из имеющихся в контексте утверждений. Несколько подробнее о них будет сказано в разделе, посвященном указателям обработки антецедентов теоремы. Однако, такие операторы могут возникать и при обработке компилятором фильтров приема. Именно, если группа утверждений $A_1, \ldots, A_n$, определяющих отношения, допускающие обработку идентифицирующими операторами, объединяется внутри фильтра "контекст(...)" в идентифицирующий терм вида "усм
($A_1 \ldots A_n$)", то компилятор обеспечит идентификацию новых переменных, встречающихся в $A_1, \ldots, A_n$, с помощью идентифицирующих операторов.

Идентификация с использованием операции подстановки

Для работы с операцией подстановки служит идентифицирующий терм "результ-подст(A B C D)". Здесь возможны два случая:

а) D - новая переменная, и тогда ей присваивается результат подстановки в терм, определяемый операторным выражением C, набора термов, определяемого операторным выражением B, вместо набора переменных, определяемого операторным выражением A.

б) B - новая переменная, и тогда ей присваивается такой набор термов, подстановка которого вместо набора переменных, определяемого операторным выражением A, в терм, определяемый операторным выражением C, дает терм, определяемый операторным выражением D.

Операнды и подтермы

1. Идентифицирующий терм "операнд(A B)", как и одноименный оператор ЛОСа, может использоваться в двух вариантах: либо A есть операторное выражение, имеющее своим значением вхождение в терм, и тогда новая переменная B идентифицируется как операнд этого вхождения, либо B - операторное выражение, имеющее своим значением вхождение, и тогда новая переменная A идентифицируется как вхождение, непосредственным операндом которого является вхождение B.

2. Идентифицирующий терм "позиция(A B)" позволяет идентифицировать новую переменную A с вхождением в терм либо набор, определяемый операторным выражением B.

3. Идентифицирующий терм "подчинено(A B)" аналогичен одноименному оператору ЛОСа. Если A - операторное выражение, значением которого служит вхождение v в терм, то новая переменная B идентифицируется с вхождением строгого надтерма v; если B - операторное выражение, значением которого служит вхождение v в терм, то новая переменная A идентифицируется с вхождением нестрогого подтерма v. В случае необходимости идентифицировать B с

вхождением нестрогого надтерма v вместо терма "подчинено(A B)" используется "Подчинено(A B)".

4. Идентифицирующий терм "вхождениетерма(A B C)" используется для идентификации переменной C со вхождением терма, заданного операторным выражением B, в терм, заданный операторным выражением A.

5. При работе с кванторной импликацией применяются идентифицирующие термы "антецедент(A B)" и "консеквент(A B)". В них операторное выражение A задает некоторую кванторную импликацию, а переменная B идентифицируется, соответственно, с антецедентом либо консеквентом этой импликации.

Заголовки операндов

Идентифицирующие термы "символ(A B)", "первыйсимвол(A B)", "второйсимвол(A B)", "предпоследсимвол(A B)", "последнийсимвол(A B)" используются для идентификации новой переменной B с символом, расположенным по вхождению, заданному операторным выражением A либо по непосредственному операнду этого вхождения (соответственно, первому, второму, предпоследнему и последнему).

Условия на вхождения и термы, связанные с предметной областью

Такие идентифицирующие термы время от времени придется добавлять к данному списку, по мере того, как этого будут требовать новые предметные области. При этом понадобится расширять и компилятор ГЕНОЛОГа. Однако, это расширение весьма несложно - оно сводится к написанию небольших программ справочника "значение", обеспечивающего компиляцию идентифицирующих термов.

1. Идентифицирующий терм "алгебрвхождение(A B)" позволяет идентифицировать переменную B с вхождением, расположенным внутри вхождения, заданного операторным выражением A, и достижимым из него только через операции "умножение", "плюс", "минус", "модуль", "дробь", "равно", "меньше", "меньшеилиравно", а также через основания степеней. Возможна обратная идентификация: операторное выражение B задает некоторое вхождение v, а новая переменная A идентифицируется с вхождением, внтури которого расположено v и достижимым из v только через операции того же списка.

2. Идентифицирующий терм "обобщмножитель(A B)" позволяет идентифицировать переменную B с вхождением, расположенным внутри вхождения, заданного операторным выражением A, и достижимым из него только через операции "минус", "умножение", "дробь", "модуль", а также через основания степеней.

3. Идентифицирующий терм "обобщслагаемое(A B)" позволяет идентифицировать переменную B с вхождением, расположенным внутри вхождения, заданного операторным выражением A, и достижимым из него только через операции "умножение", "плюс", "минус", причем в этом смысле тупиковым.

4. Идентифицирующий терм "тригаргумент(A B)" позволяет идентифицировать переменную B с вхождением аргумента тригонометрической операции в терм, определяемый операторным выражением A.

5. Идентифицирующий терм "кратныйаргумент(A B C)" позволяет идентифицировать переменную C с целым числом (термом), при домножении на который тригонометрический аргумент - значение операторного выражения B - обращается в тригонометрический аргумент - значение операторного выражения B.

6. Идентифицирующий терм "частное(A B C)" позволяет идентифицировать новую переменную C с частным целых чисел, являющихся значениями операторных выражений A и B.

7. Идентифицирующий терм "видмногочлена(A B C)" проверяет, что терм, вхождение которого определяется операторным выражением A, при раскрывании скобок преобразуется к многочлену от переменной - значения операторного выражения B, и идентифицирует новую переменную C со степенью этого многочлена (десятичным числом).

Элементы задачи либо обращения к пакетному оператору

1. Идентифицирующий терм "посылка(A B)" идентифицирует переменную A с посылкой задачи B. Указание на задачу B может отсутствовать. Если прием с данным фильтром применяется при сканировании задачи, то в этом случае берутся посылки текущей задачи; если же прием относится к пакетному оператору, то берутся посылки, указанные в обращении к оператору.

2. Идентифицирующий терм "условие(A B)" идентифицирует переменную A с условием задачи B. Указание на задачу B может отсутствовать, и тогда берется текущая задача.

3. Если требуется идентифицировать A с еще не идентифицированной в приеме посылкой (условием) текущей задачи, то используется терм "новаяпосылка(A)" (соответственно, "новоеусловие(A)").

4. Идентифицирующий терм "неизвестная(A B)" идентифицирует переменную A с неизвестной задачи B. Если указание на задачу B отсутствует, то берется текущая задача. Если B - логический символ "вход", то текущая задача приема есть задача на исследование, и тогда берется ее внешняя задача на описание. Данный терм может применяться также в приемах нормализаторов и анализаторов. Тогда B отсутствует; в случае нормализатора для определения списка неизвестных используется комментарий (неизвестные ...) из списка комментариев обращения к нормализатору. Компилятор обеспечивает создание таких комментариев при обращениях к нормализаторам, формат которых имеет элемент "неизвестные".

5. Идентифицирующий терм "цель(A $B_1 \ldots B_n$)" идентифицирует набор ($B_1 \ldots B_n$) с целью ($C_1 \ldots C_n$) задачи A. При отсутствии A берется текущая задача. Некоторые элементы B_i набора ($B_1 \ldots B_n$) суть операторные выражения, определяющие соответствующие разряды C_i; другие суть новые переменные, идентифицируемые с C_i. Требуется наличие хотя бы одной новой переменной (при этом возможен случай $n = 1$ - тогда B_1 просто идентифицируется с произвольной целью задачи A). По умолчанию новые переменные идентифицируются с термами (предполагается, что на соответствующем разряде набора

- цели также находится терм). Если требуется уточнить тип объекта, с которым должна быть идентифицирована переменная x, то соответствующее B_i может иметь вид "переменная(x)" (здесь x идентифицируется с переменной) либо вид "терм(x)" (здесь при идентификации x преобразуется к виду терма, даже если в соответствующем разряде цели находился не терм, а символ).

6. Для идентификации переменной A с надзадачей текущей задачи, имеющей тип B, используется идентифицирующий терм "надзадача(A B)". Если B есть логический символ 0, то A идентифицируется с произвольной надзадачей текущей задачи.

7. Идентифицирующий терм "внешусловие(A)" идентифицирует переменную A с условием внешней задачи на описание, блоком анализа которой служит текущая задача на исследование.

8. Идентифицирующий терм "обл(A)" идентифицирует переменную A с утверждением из области текущего вхождения в задачу.

Комментарии задач и пакетных операторов

1. Идентифицирующий терм "комментариипосылки(i $A_1 \ldots A_n$)" идентифицирует набор ($A_1 \ldots A_n$) с комментарием к терму задачи, идентифицированному с i - м антецедентом (i - символьный номер) теоремы, либо, при $i = 0$, к текущему терму задачи. Некоторые из A_j - новые переменные, которым присваиваются соответствующие разряды комментария, остальные - операторные выражения, значения которых должны совпадать с соответствующими позициями комментария. Если новая переменная x должна быть идентифицирована с переменной, то A_j берется вида "переменная(x)"; иначе, по умолчанию, новые переменные идентифицируются как термы.

2. Идентифицирующий терм "комментарий($A_1 \ldots A_n$)" идентифицирует набор ($A_1 \ldots A_n$) с комментарием к текущей задаче (если тип ее - "исследовать", то с комментарием к посылкам задачи; для приема пакетного оператора здесь берется комментарий из обращения к этому оператору). Некоторые из A_j - новые переменные, которым присваиваются соответствующие разряды комментария, остальные - операторные выражения, значения которых должны совпадать с соответствующими позициями комментария. Если новая переменная x должна быть идентифицирована с переменной, то A_j берется вида "переменная(x)"; иначе, по умолчанию, новые переменные идентифицируются как термы.

3. Идентифицирующий терм "примечание($A_1 \ldots A_n$)" идентифицирует набор ($A_1 \ldots A_n$) с комментарием к списку посылок текущей задачи. Возможно использование данного терма в приеме пакетного оператора (комментарий и в этом случае берется к списку посылок текущей задачи). Некоторые из A_j - новые переменные, которым присваиваются соответствующие разряды комментария, остальные - операторные выражения, значения которых должны совпадать с соответствующими позициями отбираемого комментария. Если новая переменная x должна быть идентифицирована с переменной, то A_j берется вида "переменная(x)"; иначе, по умолчанию, новые переменные идентифицируются как термы.

Контекст точки привязки

1. Идентифицирующий терм "внешоперанд(A)" идентифицирует новую переменную A с еще не вовлеченным в идентификацию операндом той ассоциативно - коммутативной операции, с подмножеством операндов которой идентифицирован преобразуемый терм в приеме замены.

2. Идентифицирующий терм "внешвхождение(A)" идентифицирует новую переменную A при помощи оператора "внешвхождение" с вхождением, встречающимся вне текущего вхождения в обрабатываемый нормализатором терм либо в один из термов списка B комментария "внешвхождение B" к этому нормализатору. Такие идентифицирующие термы применяются в нормализаторах, обеспечивающих выявление повторных вхождений подтермов специального вида (например, подтермов с неизвестными). Эти повторные вхождения прослеживаются во всей группе термов, получаемой добавлением обрабатываемого нормализатором терма к списку B указанного выше комментария.

3. Идентифицирующий терм "другоевхождение(A)" идентифицирует новую переменную A при помощи оператора "другоевхождение" с отличным от текущего вхождением в обрабатываемый нормализатором терм либо в один из термов списка B комментария "внешвхождение B". Применяется в нормализаторах, обеспечивающих выявление повторных вхождений подтермов специального вида.

4. Идентифицирующий терм "внешконтекст(A)" идентифицирует новую переменную A с вхождением B_2 из комментария "внешконтекст B_1 B_2" к текущему нормализатору. Последний комментарий вводится в тех случаях, когда применение нормализатора должно учитывать внешний контекст преобразуемого терма. У него B_2 - вхождение преобразуемого нормализатором терма в некоторый "базовый" терм - условие или посылку задачи, обратившейся к нормализатору, либо терм, преобразуемый внешним нормализатором.

5. Идентифицирующий терм "внешкорень(A)" идентифицирует новую переменную A с термом, вхождение B_2 в который определяется комментарием "внешконтекст B_1 B_2" к текущему нормализатору.

Наборы

1. Идентифицирующий терм "входит(A B)" идентифицирует новую переменную A с элементом набора, определяемого операторным выражением B.

2. Идентифицирующий терм "символ(A B)" идентифицирует новую переменную B с объектом, расположенным по вхождению, определяемому операторным выражением A.

3. Идентифицирующий терм "заголовок(A B)" идентифицирует новую переменную B с логическим символом либо переменной - первым элементом терма либо набора, определяемого операторным выражением A.

4. Идентифицирующий терм "постпозиция(A B)" идентифицирует новую переменную A с вхождением в набор либо терм, расположенным не раньше вхождения в этот набор или терм, являющегося значением операторного выражения B.

5. Идентифицирующий терм "новпозиция(A B)" идентифицирует новую переменную A с вхождением в набор либо терм, расположенным строго после вхождения в этот набор или терм, являющегося значением операторного выражения B.

6. Идентифицирующий терм "ключ(A B C)" идентифицирует новую переменную C с элементом набора, являющегося значением операторного выражения A, и представляющим собой либо логический символ, определяемый операторным выражением B, либо набор, начинающийся с такого логического символа.

7. Идентифицирующий терм "биключ(A B C D)" идентифицирует новую переменную D с элементом набора, являющегося значением операторного выражения A, и представляющим собой набор, первый элемент которого есть логический символ - значение операторного выражения B, а второй элемент определяется значением операторного выражения C.

8. Идентифицирующий терм "разряд(A B C)" идентифицирует новую переменную C с вхождением в набор либо терм, определяемый операторным выражением A логического символа либо переменной, определяемых операторным выражением B.

9. Идентифицирующий терм "серия(A_1 B_1 ... A_n B_n)" идентифицирует новые переменные $A_1, \ldots, A_n$ с элементами наборов - значений операторных выражений $B_1, \ldots, B_n$ - имеющими один и тот же номер $i; i = 1, 2, \ldots$.

10. Идентифицирующий терм "контрсерия(A_1 $B_1 \ldots A_n$ B_n)" идентифицирует новые переменные $A_1, \ldots, A_n$ с вхождениями в наборы, расположенными левее вхождений в эти наборы - значений операторных выражений $B_1, \ldots, B_n$ - на одно и то же число позиций $i; i = 0, 1, \ldots$.

Логические конструкции, используемые в идентифицирующих термах

1. Идентифицирующий терм "список(A $B_1 \ldots B_n$)" идентифицирует значение новой переменной A со значением одного из операторных выражений $B_1, \ldots, B_n$.

2. Идентифицирующий терм "равно(A B)" идентифицирует значение новой переменной A со значением операторного выражения B. Если A имеет вид "значение(F X)", то идентифицируется терм, определяющий значения функции F.

3. Идентифицирующий терм "новаяпеременная(A B)" идентифицирует новую переменную B с переменной, не входящей в список переменных, определяемый операторным выражением A.

4. Идентифицирующий терм "перестановка(A B C D)" идентифицирует новые переменные C и D с парой операторных выражений A и B - в прямом либо обратном порядке.

5. Идентифицирующий терм "или($A_1 \ldots A_n$)" позволяет осуществлять идентификацию согласно одному из дизъюнктивных членов $A_i; i = 1, \ldots, n$. Каждый такой член имеет вид B_1 & ... & B_m, где $B_1, \ldots, B_m$ - набор идентифицирующих термов и фильтров, а также указателей идентификации - как для фильтра "контекст(...)". Идентификации подлежат общие для всех $A_1, \ldots, A_n$ новые переменные.

6. Идентифицирующий терм "значения(A B)" позволяет провести идентификацию для утверждения A предметного уровня посредством пакетного синтезатора, предусмотренного для утверждений такого вида. B - терм "посылки(C)", определяющий дополнительные посылки C обращения к синтезатору. Утверждение C - конъюнкция этих посылок - также относится к предметному уровню. Если дополнительные посылки не требуются, то терм B отсутствует.

11.3.15 Операторные выражения

Операторные выражения используются для задания в фильтрах приемов различных объектов "структурного" уровня (термов, наборов, символов и т.п.). Их следует отличать от выражений, задающих объекты "предметного" уровня (числа, точки пространства, функции и т.п.). Последние выражения также могут использоваться в фильтрах, однако это происходит значительно реже и лишь в специально оговоренных случаях.

Простейшие выражения

1. Любой логический символ представляет собой операторное выражение; значением его (за исключением приводимых ниже служебных слов) служит он сам.

2. Любая теоремная переменная, либо вспомогательная переменная, определенная до этого некоторым идентифицирующим термом, представляет собой операторное выражение. Тип значения этого выражения (терм, логический символ, вхождение, и т.п.) определяется компилятором ГЕНОЛОГа автоматически - в зависимости от контекста. Исключительные случаи, когда такое автоматическое доопределение типа необходимо уточнять вручную, указываются при описании различных контекстов использования операторных выражений.

3. Логический символ "теквхожд" - служебный символ, обозначающий заменяемый терм для приемов замены (в зависимости от контекста, этот символ может также обозначать вхождение заменяемого терма; далее аналогичные оговорки опускаем).

4. Логический символ "корень" служебный символ, обозначающий текущий терм задачи; этот терм содержит точку привязки приема.

5. Логический символ "вход" - служебный символ, используемый в фильтрах, сопровождающих обращение к нормализатору (см. ниже раздел, посвященный обращениям к нормализаторам), и обозначающий подлежащий нормализации терм.

6. Логический символ "текущийуровень" - служебный символ, обозначающий текущий уровень текущей задачи.

7. Логический символ "исходнаязадача" - служебный символ, обозначающий исходную задачу, т.е. фиктивную задачу на исследование, при сканировании которой запускается процедура интерфейса.

8. Логический символ "посылки" - служебный символ, обозначающий список утверждений, образующих контекст реализуемого приема.

9. Логический символ "текущаязадача" - служебный символ, обозначающий текущую задачу.

10. Логический символ "внешописать" - служебный символ, обозначающий внешнюю задачу на описание для текущей задачи на исследование.

11. Логический символ "результат" - служебный символ, обозначающий заменяющий терм приема замены. Применяется как в приемах сканирования задачи, так и в приемах нормализаторов.

Численные выражения

1. Операторное выражение "количествооперандов(A)" обозначает число (десятичное) корневых операндов терма, определяемого операторным выражением A. Служебный символ "количествооперандов" используется для обозначения числа операндов заменяемого подтерма.

2. Служебный символ "числонеизвестных" обозначает число неизвестных текущей задачи либо текущего пакетного оператора. Операторное выражение "числонеизвестных(A)" обозначает число (десятичное) неизвестных в терме, определяемом операторным выражением A.

3. Для нахождения числа наборов объектов, удовлетворяющих заданному условию, используется логическая конструкция "число(A)", аналогичная фильтру "контекст(A)". Здесь A - список идентифицирующих термов, указателей идентификации и фильтров, описывающих условия на объекты - значения новых переменных, определяемых идентифицирующими термами из A. Значение операторного выражения представлено в формате десятичного числа.

4. Если нужно не просто определить число наборов объектов, удовлтеворяющих заданному условию, но просуммировать по всем таким наборам значения заданного операторного выражения, то используется логическая конструкция "суммавсех(A t)". Здесь, как и в предыдущем случае, A есть список идентифицирующих термов, указателей идентификации и фильтров, описывающих условия на объекты - значения новых переменных. t есть операторное выражение, определяющее суммируемые величины (десятичные числа).

5. Возможно нахождение наибольшего общего делителя всех целых чисел, определяемых заданным операторным выражением для всевозможных наборов объектов, удовлетворяющих заданному условию. Здесь используется конструкция "нод(A t)"; A - список идентифицирующих термов, указателей идентификации и фильтров, описывающих условия на объекты - значения новых переменных; t - операторное выражение, определяющее указанные целые числа.

   Аналогичная конструкция "нок(A t)" используется для нахождения наименьшего общего кратного.

   Если нужно определить наибольший общий делитель либо наименьшее общее кратное двух чисел, определяемых операторными выражениями B и C, то используются, соответственно, операторные выражения "нод(B C)" и "нок(B C)". Чтобы не было путаницы с указанными выше конструкциями "нод(A t)" и "нок(A t)", список A должен начинаться с идентифицирующего терма либо с фильтра.

6. Для сложения, вычитания и умножения чисел, определенных операторными выражениями A и B, используются операторные выражения "плюс(A B)", "вычитание(A B)" и "умножение(A B)". Здесь речь идет о десятичных числах. Для выполнения этих же операций над числами в символьной записи используются операторные выражения "плюссимв(A B)", "вычитсимв(A B)" и "умножсимв(A B)". Заметим, что все перечисленные операции - строго двуместные; если число операндов более двух, то нужно использовать выражение многоместной операции через двуместные.

   Чтобы изменить знак десятичного числа, определенного операторным выражением A, используется операторное выражение "минус(A)"; аналог для символьного формата чисел здесь отсутствует.

7. Номер (в формате десятичного числа) логического символа, определенного операторным выражением A, задается операторным выражением "номерсимв(A)". Чтобы определить номер переменной, задаваемой операторным выражением A, используется выражение "номерпеременной(A)". Этот номер представлен уже в символьном формате.

8. Операторное выражение "числзначение(A)" имеет своим значением число в десятичном формате, определяемое константным числовым термом - значением операторного выражения A.

9. Операторное выражение "длинанабора(A)" имеет своим значением константный числовой терм, задающий длину набора, определяемого операторным выражением A. Чтобы определить десятичное число разрядов набора либо терма, определяемого операторным выражением A, используется операторное выражение "Длина(A)".

10. Операторное выражение "числочленов(A B)" имеет своим значением число корневых операндов терма, определенного выражением A, если заголовок этого терма - логический символ B. Иначе это выражение имеет своим значением число 1.

11. Для нахождения числа делителей целого числа - значения операторного выражения A - используется операторное выражение "числоделителей(A)".

Переменные и логические символы

1. Для обозначения переменной, номер которой (символьное число) определяется операторным выражением A, используется операторное выражение "икс(A)". Для обозначения логического символа, номер которого (в формате десятичного числа) определяется операторным выражением A, используется операторное выражение "симвномер(A)".

2. Если требуется ввести набор переменных, не входящих в список переменных, определяемый операторным выражением A, и имеющий такую же длину, как набор - значение операторного выражения B, то используется операторное выражение "кортежпеременных(A B)".

3. Связывающая приставка терма, определяемого операторным выражением A, обозначается как "связприставка(A)".

4. Если операторное выражение A определяет терм либо набор термов, то выражение "параметры(A)" определяет список всех свободных переменных терма A (термов набора A). Выражение "списокпеременных(A)" определяет в этой же ситуации список всех вообще переменных, входящих в A (либо в термы набора A).

5. Если операторное выражение A обозначает терм, то выражение "Неизвестные(A)" обозначает список всех неизвестных, входящих в этот терм.

Наборы

1. Набор, образованный объектами - значениями операторных выражений $A_1, \ldots,$ A_n - обозначается "набор($A_1 \ldots A_n$)".

2. Для создания набора, перечисляющего без повторений объекты, удовлетворяющие заданному условию, используется операторное выражение "перечисление(A B)". Здесь A - список идентифицирующих термов, указателей идентификации и фильтров, описывающих условия на новые переменные (аналогично фильтру "контекст(A)"); B - операторное выражение с этими переменными, определяющее перечисляемые объекты. Если тип значения выражения B допускает различные варианты, то по умолчанию это значение определяется как логический символ либо символ переменной; чтобы явно указать, что рассматриваемый объект должен представлять собой терм, B берется вида "терм(C)", где C собственно и определяет данный терм.

3. Для создания набора, перечисляющего с возможными повторениями объекты, удовлетворяющие заданному условию, используется операторное выражение "выписка(A B)" - аналогично выражению "перечисление(A B)".

4. Если требуется сформировать набор заданной длины, на каждой позиции которого находится один и тот же объект, то используется операторное выражение "бланк(A B C)". Здесь операторное выражение C пределяет некоторое символьное число, причем длина формируемого набора на C больше длины набора A. Операторное выражение B указывает объект, помещаемый на каждой позиции формируемого набора.

5. Конкатенация наборов либо символов, определяемых операторными выражениями $A_1, \ldots, A_n$, обозначается "конкатенация($A_1 \ldots A_n$)".

6. Операторные выражения "вычеркивание(A B)", "пересечениесписков(A B)", "объединениесписков(A B)", как и в ЛОСе, определяют, соответственно, результат исключения из набора, определяемого операторным выражением A, элементов набора, определяемого операторным выражением B (как и в ЛОСе, при исключении учитывается кратность вхождений - из A исключается лишь столько вхождений заданного элемента, сколько их есть в B); пересечение этих наборов (в A сохраняются только те элементы, которые входят в B) и их объединение (к концу набора A присоединяются все не входившие в него элементы набора B).

7. Результат присоединения к началу набора, определяемого операторным выражением A, объекта, определяемого операторным выражением B, обозначается

"префикс(B A)"; результат присоединения B к концу A - "суффикс(A B)". Результат вставки в набор A перед позицией, определяемой выражением B, значения выражения C, обозначается "вставка(A B C)".

8. Результат отбрасывания первого элемента набора, определяемого операторным выражением A, обозначается "окончание(A)". Результат исключения в наборе A разряда либо группы разрядов, определяемых выражением B, обозначается "исключение(A B)".

9. Копия набора, определяемого операторным выражением A, обозначается "копия(A)".

10. Первый элемент терма либо набора, определяемого операторным выражением A, обозначается "начало(A)"; последний элемент набора A обозначается "конец(A)". Объект, расположенный на $i+1$ - м месте, считая от начала набора A, обозначается "левпозиция(A I)", где i - сивмольное число, являющееся значением выражения I. Объект, расположенный на $i+1$ - м месте, считая с конца набора A, обозначается "правпозиция(A I)".

11. Объект, расположенный в наборе, определяемом выражением B на позиции, соответствующей позиции объекта - значения выражения C в наборе, определяемом выражением A, обозначается "таблзначение(A B C)".

12. Набор объектов X, таких, что в наборе a имеется пара (b X), обозначается "Ключ(A B)"; здесь A, B - операторные выражения, имеющие значения a, b.

Термы

1. Чтобы задать терм T, полученный из некоторого терма A предметного уровня подстановкой в него вместо "теоремных" переменных идентифицированных с ними подтермов, используется операторное выражение "терм(A)". Если в A имеются подтермы, к которым согласно описанию приема должный применяться цепочки нормализаторов либо вспомогательных задач, то при построении T эти подтермы подвергаются данной обработке.

2. Иногда терм, который нужен в качестве значения операторного выражения, уже имеется в теореме. Чтобы не переписывать этот терм, используя приведенную выше конструкцию "терм(...)", можно сослаться на его вхождение в теорему при помощи так называемого указателя вхождения в теорему - терма "фикс(n_0 $n_1 \ldots n_k$)". Если теорема приема имеет вид "длялюбого($x_1 \ldots x_p$ если $A_1 \ldots A_m$ то A_0)", то n_0 есть номер i того утверждения A_i, в котором расположено рассматриваемое вхождение. При этом ($n_1 \ldots n_k$) - последовательность номеров операндов в A_i, определяющая путь от корня A_i к данному вхождению (операнды нумеруются начиная с 1, слева направо; сначала выполняется переход к n_1 - му операнду корневой операции A_i, затем к n_2 - му операнду данного операнда, и т.д.). Если заголовок теоремы приема отличен от квантора общности, то указатель вхождения перечисляет номера операндов прямо с корневой операции теоремы. Указатель вхождения в теорему терма A заменяет операторное выражение "терм(A)" и сам может рассматриваться как операторное выражение. Оно является даже более сильным, чем "терм(A)", так как в тех фильтрах, где нужно вхождение терма A, а не сам этот терм, в

качестве значения выражения и берется такое вхождение (то, с которым было идентифицировано соответствующее вхождение A в теорему). Заметим, что набирать вручную последовательность номеров $n_0, \ldots n_k$ из указателя вхождения не нужно - в интерфейсе редактора приемов предусмотрена возможность автоматического ввода этой последовательности после выделения клавишами курсора подтерма теоремы.

3. Терм, сформированный из операндов, перечисляемых в наборе, являющемся значением операторного выражения B, при помощи заголовка - значения выражения A, обозначается "сборка(A B)". Если в случае одноэлементного набора B заголовок A не навешивается, а результатом служит сам элемент набора B, то используется операторное выражение "унисборка(A B)".

4. Если операторные выражения $A_1, \ldots, A_n$ определяют некоторые термы $a_1, \ldots, a_n$, а операторное выражение B - логический символ b, то операторное выражение "запись(B $A_1 \ldots A_n$)" определяет терм $b(a_1 \ldots a_n)$.

5. Если операторное выражение A определяет некоторый терм $f(t_1 \ldots t_n)$ либо вхождение первого символа такого терма, то операторное выражение "первыйтерм(A)" определяет терм t_1, операторное выражение "второйтерм(A)" - терм t_2, операторное выражение "предпоследтерм(A)" - терм t_{n-1} и операторное выражение "последнийтерм(A)" - терм t_n. Наконец, операторное выражение "набороперандов(A)" определяет набор термов $(t_1, \ldots, t_n)$.

6. Если операторное выражение A определяет вхождение первого символа некоторого подтерма, то сам этот подтерм задается выражением "подтерм(A)". Чтобы определить тот терм, вхождение в который задает выражение A, используется выражение "базавхождения(A)".

7. Если операторное выражение A определяет вхождение первого символа некоторого терма, а операторное выражение B - вхождение корневого операнда этого терма, то выражение "исключениеоперанда(A B)" определяет результат исключения в терме A его операнда B (если остается единственный операнд корневой операции, то корневая операция отбрасывается).

8. Если операторное выражение A определяет терм; операторное выражение B - вхождение либо набор вхождений в A (упорядоченных слева направо), а операторное выражение C - соответственно, терм либо набор термов (той же длины, что и B), то выражение "заменатермов(A B C)" обозначает результат замены в A вхождений B на термы C.

Вхождения

1. Если операторное выражение A обозначает вхождение в терм либо в набор, то выражение "буква(A)" обозначает объект, расположенный по этому вхождению.

2. Если операторное выражение A определяет набор, то выражение "левыйкрай(A)" обозначает вхождение первого элемента набора A, а "правыйкрай(A)" - вхождение последнего элемента набора A. Если, далее, операторное выражение B определяет символьный номер i, то выражение "окрестность(A B)" обозначает вхождение $i+1$ - го элемента набора A.

3. Если операторное выражение A определяет терм $f(a_1 \ldots a_n)$ либо вхождение этого терма, то выражения "первыйоперанд(A)", "второйоперанд(A)", "предпоследоперанд(A)", "последнийоперанд(A)" обозначают вхождения операндов a_1, a_2, a_{n-1}, a_n соответственно.

4. Если операторное выражение A определяет вхождение в набор, то выражение "левсосед(A)" определяет вхождение предыдущего элемента набора (если A - вхождение первого элемента, то берется то же самое вхождение A). Выражение "правсосед(A)" обозначает вхождение следующего элемента набора (если A - вхождение последнего элемента, то берется оно само).

5. Если операторное выражение A определяет вхождение в набор, являющийся значением операторного выражения B, причем операторное выражение C также определяет набор, то выражение "соотвпозиция(A B C)" обозначает вхождение в набор C разряда, имеющего тот же номер, что и разряд A набора B.

6. При идентификации антецедента вида $P(A\ B)$, где P - симметричный и транзитивный предикатный символ, может использоваться указатель "равно(i)" (i - символьный номер данного антецедента). Такой антецедент может быть усмотрен либо из явно присутствующего в контексте утверждения $P(A\ B)$, либо из двух утверждений $P(A\ C)$, $P(B\ C)$. В последнем случае C - "стандартное обозначение" выражений A, B, а также, быть может, ряда других встречающихся в контексте выражений (в первом случае роль "стандартного обозначения" играет выражение B). Появление группы утверждений $P(\ldots)$, связывающих $A, B, \ldots$ с их "стандартным обозначением", должно быть заблаговременно обеспечено специальной стандартизацией. Чтобы в описании приема можно было анализировать не только выражения A и B, но и выражение C, используется операторное выражение "выражение(i)", значением которого служит вхождение первого символа подтерма C (фактически это вхождение второго операнда в использованное при идентификации утверждение $P(A\ C)$).

7. Если i - й антецедент теоремы был идентифицирован с посылкой либо условием задачи, то операторное выражение "вхождениепосылки(i)" определяет вхождение этой посылки либо условия, соответственно, в список посылок либо условий. Здесь i - символьный номер.

Задачи

1. Если операторное выражение A обозначает задачу, то операторные выражения "цели(A)", "списокусловий(A)", "списокпосылок(A)" обозначают, соответственно, список целей, список условий (используется только в ситуации, когда заблаговременно установлено, что A - задача на описание) и список посылок задачи A. Возможно отбрасывание явного указания на задачу A - тогда рассматриваемое операторное выражение становится логическим символом. В этой ситуации по умолчанию берется текущая задача.

2. Если операторное выражение A обозначает задачу, то операторные выражения "неизвестные(A), "переменные(A)" обозначают, соответственно, список всех неизвестных задачи A (в него не включается заголовок "неизвестные" цели, перечисляющей неизвестные) и список всех переменных, встречающихся в задаче A. Возможно опускание явной ссылки на задачу, и тогда берется текущая задача.

3. Если операторное выражение A обозначает задачу, то операторное выражение "комментарии(A)" обозначает список комментариев к задаче A (которая не должна иметь типа "исследовать"). Возможно опускание явной ссылки на задачу A, и тогда берется текущая задача. Данное операторное выражение (без ссылки на A) применяется также для приемов пакетных операторов - тогда оно имеет своим значением список комментариев обращения к данному оператору. Операторное выражение "комментариипосылок(A)" определяет аналогичным образом список комментариев к посылкам задачи A.

4. Чтобы обозначить список комментариев к заданному условию задачи, определяемой операторным выражением A, используется выражение "комментарииусловия(A B)". Здесь B - либо символьный номер антецедента теоремы, идентифицированного с рассматриваемым условием, либо логический символ "корень" (указатель на условие, в котором находится текущее вхождение сканирования задачи), либо операторное выражение, имеющее своим значением рассматриваемое условие. Возможно опускание явной ссылки на задачу A, и тогда берется текущая задача.

5. Операторное выражение "комментариипосылки(A B)" совершенно аналогично выражению "комментарииусловия(A B)" - оно обозначает список комментариев к посылке B задачи A.

6. Если операторное выражение A обозначает задачу, то выражение "максимальныйуровень(A)" обозначает максимальный уровень этой задачи. Возможно отсутствие явного указания на задачу A, и тогда берется текущая задача.

7. Если операторное выражение A определяет задачу на доказательство либо на преобразование, то выражение "условие(A)" обозначает условие этой задачи. Возможно отсутствие явного указания на задачу A, и тогда берется текущая задача.

Обращения к справочникам

Результат обращения к справочнику определяется операторным выражением "справка(A B $C_1 \ldots C_n$)". Здесь A - тип справочника (логический символ); B - логический символ, к программе которого происходит обращение (может задаваться произвольным операторным выражением); $C_1, \ldots, C_n$ - операторные выражения, определяющие сопутствующие данные обращения.

Определение типа объекта

Операторное выражение "типданных(A)" определяет тип объекта, заданного операторным выражением A: 0 - логический символ либо переменная, 1 - терм, 2 - вхождение, 3 - ошибка (ссылка на пустую зону), 4 - набор, не являющийся термом.

Условное выражение

Операторное выражение "вариант(A B C)" определяет при истинном фильтре A значение, задаваемое операторным выражением B, а при ложном - операторным выражением C.

11.3.16 Очередность проверки фильтров и дополнительные действия при проверке

Организация слежения за ситуацией, в которой ложный на текущий момент фильтр может оказаться истинным

Если условие фильтра A не выполнено, но оно может оказаться выполненным при возникновении в списке посылок задачи одного из утверждений списка B, то при появлении таких посылок целесообразна повторная попытка применения приема. Ее можно организовать, например, понизив вес текущего терма (при сканировании которого была инициирована попытка применения приема) до заданной величины P. Указанные действия - организация слежения за появлением при решении задачи посылок списка B и соответствующее переключение внимания - будут выполняться, если вместо фильтра A поместить в описании приема фильтр "пересмотр(A B P)".

Проверка истинности фильтра в конце программы приема

Проверка истинности некоторых фильтров приема оказывается достаточно трудоемкой. В таких случаях ее выполнение целесообразно предпринимать лишь в самом конце программы приема, после применения более дешевых "отсекающих" средств. Чтобы указать компилятору на необходимость размещения программы фильтра A в конце программы приема, этот фильтр преобразуется к виду "конец(A)". Таким образом можно размечать лишь "корневые" фильтры, не расположенные внутри более сложного фильтра.

Проверка истинности фильтра непосредственно перед обработкой заданного антецедента

В некоторых случаях проверку истинности фильтра следует выполнять не в начале программы приема (так как она сравнительно трудоемкая), но все же до существенно более трудоемкой обработки некоторого антецедента (например, передаваемого проверочному оператору). В этих случаях для сообщения компилятору пожелания о данном размещении фильтра A он преобразуется к виду "проверка(A i)", где i - символьный номер рассматриваемого антецедента. Таким образом размечаются лишь "корневые" вхождения фильтров.

11.4 Типы указателей приемов

Перечислим типы используемых в ГЕНОЛОГе указателей приема - различных информационных элементов, уточняющих компилятору необходимую версию программы приема.

11.4.1 Указатели идентификации

В подавляющем большинстве случаев логические конструкции, используемые в теореме приема, не встречаются в задаче в чистом виде. Для усмотрения возможности применения данной теоремы обычно приходится применять специальные преобразования встречающихся в задаче термов, "подгоняя" их под испольуемые в теореме

выражения. При описании приема на ГЕНОЛОГе такие преобразования должны быть явно заданы - с помощью так называемых указателей идентификации. Компилятор ГЕНОЛОГа создает на основе этих указателей программу усмотрения замаскированных возможностей применения теоремы, существенно расширяя тем самым область применимости приема.

Вырожденные значения переменных

1. Если теоремная переменная x встречается в идентифицируемой части теоремы в качестве операнда таких операций $f(x\ Q)$, которые сводятся к Q при специальном "единичном"значении a переменной x, то использование указателя "единица($a\ x$)" обеспечивает трансляцию приема, при которой идентифицируемый с $f(x\ Q)$ терм задачи t, не имеющий заголовка f, будет автоматически представляться в виде $f(a\ t)$; при этом x будет идентифицироваться с a, а Q - с t. Возможно объединение нескольких указателей "единица($a\ x_1$)", ... "единица($a\ x_n$)" в один указатель "единица($a\ x_1 \ldots x_n$)". Для использования данного указателя предварительно должен быть создан соответствующий паре f, a прием справочника "единица". Пример использования указателя "единица" - для идентификации квадратного трехчлена $ax^2 + bx + c$, у которого коэффициенты a, b могут оказаться вырожденными - равными 1. В этом случае вместо указанных в записи общего вида квадратного трехчлена произведений ax^2 и bx будут иметься некоторые выражения, не имеющие вида произведений. Чтобы компилятор мог воспринять их как произведения на единицу, вводится указатель "единица(1 a b)".

2. Иногда в теореме используется ассоциативно - коммутативная операция, некоторые операнды которой при идентификации могут отсутствовать - в этом случае определенным параметрам таких отсутствующих членов присваиваются наперед заданные константные значения (например, нулевое значение коэффициента в отсутствующем слагаемом). Чтобы обозначить такую ситуацию, используется указатель "подстановка($v\ x\ a$)". Здесь v - указатель вхождения в теорему операнда ассоциативно-коммутативной операции, который может отсутствовать; x - переменная (она должна иметь единственное вхождение в идентифицируемую часть); a - логический символ, с однобуквенным термом (a) которого идентифицируется x при отсутствии операнда v. Пример использования указателя "подстановка" - при задании вида кубического многочлена $ax^3 + bx^2 + cx + d$ уточняется, что члены второй либо первой степени могут отсутствовать. Соответственно, вводятся указатели "подстановка($K_1\ b\ 0$)" и "подстановка($K_2\ c\ 0$)", где K_1, K_2 - указатели вхождения в теорему членов второй и первой степени. Такие указатели можно использовать одновременно с указателями "единица" для a, b, c. Различные указатели, затрагивающие идентификацию одной и той же переменной, обычно можно вводить одновременно - кроме случаев, когда они явно противоречат друг другу.

3. Если переменная x идентифицируется как "общая часть" нескольких вхождений ассоциативно-коммутативной операции f, то можно разрешить идентифицировать ее в случае пустой общей части с единицей этой операции. Для этого используется указатель "пустоеслово(x)". Разрешается объединять несколько таких указателей в один указатель "пустоеслово($x_1 \ldots x_n$)". Обычно данный указатель не нужен - вместо него берется указатель "единица($a\ x$)". Однако,

если в теореме имеется более двух операндов операции f, и нужно предусмотреть возможность вырожденной идентификации данной операции с выражением, не имеющим заголовка f (когда все операнды, кроме одного, обращаются в единицу), то указатель "пустоеслово" дополняет указатель "единица".

Учет внешней одноместной операции либо связки

1. Если одноместная операция f удовлетворяет тождеству $f(f(X)) = X$, то указатель "отрицание(f $x_1 \dots x_n$)" определяет такую идентификацию переменных $x_i; i = 1, \dots, n$, при которой терм задачи t, идентифицируемый с $f(x_i)$, автоматически представляется в виде $f(f(t))$. Заблаговременно должен быть создан прием справочника "отрицание" для f. Пример применения данного указателя - идентификация в алгебре логики выражения, содержащего одновременно и переменную, и ее отрицание. Если выражение вида $a \vee \neg a \cdot b$ при идентификации снабдить указателем "отрицание(отр a)", то становится возможной идентификация a с $\neg c$, а отрицания a - с c. Здесь "отр" - логический символ для булева отрицания $\neg$.

2. Если переменная x имеет единственное вхождение в таком идентифицируемом подтерме $t(x)$ теоремы приема, что для одноместной операции f выполнено тождественно $f(t(X)) = t(f(X))$, то указатель "заменазнака(f x)" определяет идентификацию терма $t(x)$, при который идентифицируемый с ним терм вида $f(t(A))$ автоматически преобразуется к виду $t(f(A))$, т.е. внешний знак f передается переменной x. Подтерм $t(x)$ определяется по идентифицируемому вхождению переменной x как максимальный подтерм, для которого возможна указанная передача знака f по цепочке вложенных операций (используются приемы справочника "заменазнака" для этих операций, которые должны быть заблаговременно созданы). Возможна идентификация с передачей знака f при наличии нескольких непересекающихся подтермов $t_1(x), \dots, t_m(x)$. Несколько указателей "заменазнака(f x_1)", ..., "заменазнака(f x_n)" могут быть объединены в один указатель "заменазнака(f $x_1 \dots x_n$)". Пример использования указателя "заменазнака" - идентификация квадратного трехчлена $ax^2 + bx + c$, у которого перед членами второй и первой степени могут стоять знаки "минус". В этом случае указатель "заменазнака(минус a b)"обеспечивает передачу данных минусов коэффициентам a, b.

3. Если консеквент приема замены имеет вид "не(A)", то по умолчанию будет идентифицироваться вхождение утверждения A, заменяемое приемом на константу "ложь". В тех случаях, когда требуется идентифицировать именно вхождение утверждения "не(A)", заменяемое на константу "истина", применяется указатель идентификации "прямойответ". Пример применения данного указателя - приемы усмотрения истинности условий задачи на описание, имеющих вид отрицания равенства нулю. Такие вспомогательные условия обычно возникают при сопровождении основных условий по области допустимых значений. В некоторый момент решения надобность в них отпадает, и тогда предпринимается попытка отбросить данные условия, предварительно убедившись в том, что они вытекают из прочих условий. Без указателя "прямойответ" данный прием предпринимал бы также попытку усмотрения ложности имеющихся в условиях уравнений с нулем в правой части, что нецелесообразно.

Перестановки операндов

1. Если заменяемый терм приема тождественной либо эквивалентной замены (в том числе приема нормализатора) имеет вид $f(A_1\ A_2)$, причем при идентификации допустима перестановка операндов A_1, A_2 с одновременной перестановкой операндов B_1, B_2 заменяющего терма $f(B_1\ B_2)$, то используется указатель "дробь". Заменяющий терм может и не иметь вида $f(B_1\ B_2)$; тогда используется искусственное его представление в таком виде, использующее единицу операции f. Пример использования указателя "дробь" - тригонометрическое сокращение с помощью тождества $a(\sin b)^n/c(\tan b)^n = a(\cos b)^n/c$. Если ввести данный указатель, то сокращение будет выполняться и при перестановке числителя со знаменателем.

2. Указатель "дробь($v\ w_1 \dots w_n$)" обобщает указатель "дробь". У него v - указатель вхождения в теорему идентифицируемого терма $f(A_1\ A_2)$, операнды которого A_1 и A_2 при идентификации могут быть переставлены; $w_1, \dots, w_n$ - список указателей вхождений двуместных операций либо отношений, при идентификации либо формировании которых осуществляется синхронная с v перестановка операндов. Пример использования данного указателя - декомпозиция строгого неравенства для произведения по теореме $0 \le a \to (0 < ab \leftrightarrow 0 < a\ \&\ 0 < b)$. Указатель "дробь(фикс(0 1)фикс(0 2 2))" выделяет вхождения в теорему неравенства $0 < ab$ и неравенства $0 < b$. При его наличии прием будет выполнять также и декомпозицию неравенств вида $ab < 0$ на $0 < a, b < 0$.

3. Указатель "циклупорядочение(v)" используется в тех случаях, когда идентификация терма $f(A_1 \dots A_n)$, расположенного по указателю вхождения v, выполняется с возможными циклическими перестановками операндов $A_1, \dots, A_n$. Если допускается лишь однократная циклическая перестановка, при которой перечисление операндов начинается со второго операнда A_2, то используется указатель "второйоперанд(v)". Пример использования указателя "циклупорядочение" - идентификация утверждения "четырехугольник($ABCD$)", которую можно начинать с любой из расположенных по циклу вершин A, B, C, D. Пример использования указателя "второйоперанд" - идентификация утверждения "параллелограмм($ABCD$)" в той ситуации, когда достаточно различать лишь два случая - выделение пары параллельных сторон AB, CD либо пары BC, AD.

4. Указатель "контрсерия(v)" используется в тех случаях, когда идентификация терма $f(A_1 \dots A_n)$, расположенного по указателю вхождения v, выполняется с возможным изменением порядка операндов на противоположный. Если же, кроме этого, возможны также любые циклические перестановки операндов, то используется указатель "подобны(v)". Пример использования указателя "контрсерия" - идентификация утверждения "трапеция($ABCD$)", если в приеме нужно обеспечить возможности идентификации, симметричные относительно вертикального для трапеции направления. При этом одновременно рассматриваются два различных обозначения одной и той же трапеции - $ABCD$ и $DCBA$. Пример использования указателя "подобны" - идентификация утверждения "четырехугольник($ABCD$)" в тех приемах, где нужно учесть не только возможные варианты обозначения $BCDA$, $CDAB$, ..., но и варианты $DCBA$, $CBAD$, ...

5. Указатель "множество(v)" используется в тех случаях, когда идентификация

терма $f(A_1 \dots A_n)$, расположенного по указателю вхождения v, происходит с произвольной перестановкой операндов. Если данный терм определяет набор, идентифицируемый без учета порядка элементов (этот набор может задаваться явным перечислением элементов - "набор($A_1 \dots A_n$)" либо косвенным образом - "префикс(A B)", "конкатенация(набор(A_1 A_2)B)", где для идентификации выделяются один либо два элемента набора), то используется указатель "список(v)". Пример использования указателя "множество" - идентификация выражения "плоскость(ABC)" без учета порядка точек A, B, C. Пример использования указателя "список(фикс(0 1 1))" - применение тождества "равно(перечень (конкатенация(набор(a a) b)) перечень (префикс (a b)))" для исключения повторяющихся элементов в списке, определяющем конечное множество. Этот указатель выделяет вхождение выражения "конкатенация(набор(a a)b)", которое идентифицируется с набором из задачи так, что различные позиции, на которых встречаются одинаковые элементы a, - произвольны.

6. При идентификации операндов коммутативной операции по умолчанию учитывается возможность перестановки операндов. Кроме того, рассматриваются символы со специальной симметрией (см ниже), идентификация которых также по умолчанию производится с допустимыми перестановками операндов. Если же такую перестановку требуется заблокировать - для коммутативной неассоциативной операции либо операции со специальной симметрией - то используется указатель "коммутативно(v)"; v - указатель вхождения в теорему этой операции.

Альтернативные заголовки подтермов

1. В некоторых случаях теорема приема может быть модифицирована при одновременной замене заголовков операций либо отношений, расположенных на заданных вхождениях. Эти вхождения выделяются как в идентифицируемой части теоремы, так и в тех ее частях, которые определяют новые термы, создаваемые приемом. Возможность модификации определяется указателем "альтернатива($C_1 \dots C_n$)", где каждое C_i - группа операндов вида $A_1 \dots A_m$ B, причем $A_1, \dots, A_m$ - указатели вхождений в теорему термов, которые в модифицированной теореме все имеют альтернативный заголовок B. Требуется, чтобы альтернативные заголовки имели одновременно все термы по указателям вхождений из $C_1, \dots, C_n$. Если имеется несколько различных указателей "альтернатива(...)", то модификации теоремы согласно этим указателям осуществляются независимо (т.е. возможны любые их комбинации). При этом различные группы термов, выделенные в теореме данными указателями, не должны пересекаться друг с другом. В качестве примера рассмотрим теорему $0 \le \mathrm{arctg}(a) - \mathrm{arctg}(b) \leftrightarrow 0 \le a - b$, определяющую прием исключения арктангенсов в неравенствах. Чтобы прием обрабатывал как нестрогие, так и строгие неравенства, вводится указатель "альтернатива(фикс(0 1)фикс(0 2)меньше)".

2. Если в теореме выделяется вхождение операции либо отношения, для которого при идентификации допускается более одного альтернативного заголовка, то используется указатель идентификации "вариант(A $B_1 \dots B_n$)", где A - указатель данного вхождения в теорему; $B_1, \dots, B_n$ - список всех альтернативных заголовков. В качестве примера рассмотрим теорему "вершина(A, E) & линвторпорядка(E) $\rightarrow A \in E$", определяющую прием вывода следствий в задачах

по аналитической геометрии. Чтобы прием мог срабатывать на кривых второго порядка конкретных типов, вводится указатель "вариант(фикс(2)эллипс гипербола парабола)".

3. Для некоторых ассоциативно-коммутативных операций f предусмотрена специальная процедура, которая может при идентификации вводить искусственное представление терма в виде $f(A_1 \dots A_n)$, даже если он первоначально не имел заголовка f. Например, степенное выражение при идентификации может рассматриваться как произведение двух других степенных выражений. Такие процедуры определяются справочником "пересечениесписков" (см. об этом справочнике в начале следующего подраздела, посвященного идентификации операндов ассоциативно - коммутативных операций). Чтобы заблокировать проверку наличия у идентифицирующего терма заголовка f, в таких случаях рекомендуется применять указатель идентификации "пересечениесписков($K_1 \dots K_m$)", где $K_1, \dots, K_m$ - указатели вхождений в теорему идентифицируемых операций f. Отметим, что иногда такая блокировка выполняется компилятором ГЕНОЛОГа автоматически - например, при идентификации термов $f(x\ T)$, где переменная x может принимать вырожденное "единичное" значение. Как правило, специальная блокировка нужна для операций f с более чем двумя операндами.

4. Если необходимо при идентификации выражения "частнпроизв($A_1\ A_2\ A_3$)" для частной производной автоматически приводить к виду "частнпроизв($B\ 1\ C$)" выражения "производная($B\ C$)" для обычной производной, то используется указатель "частнпроизв".

Идентификация операндов ассоциативно - коммутативной операции

1. При идентификации операндов ассоциативно - коммутативных операций может использоваться специальная процедура "пересечения" списков операндов двух операций, определяющая общую их переменную. Фактически это пока сделано лишь для вещественного и комплексного умножений - здесь "пересечение" понимается как определение некоторой версии наибольшего общего делителя двух одночленов. Однако, в компиляторе ГЕНОЛОГа предусмотрена общая конструкция, позволяющая при необходимости подключать и другие процедуры такого типа. Компилятор использует для определения данной процедуры по символу f ассоциативно - коммутативной операции справочник "пересечениесписков". Эта же процедура определяет "остаточные" после определения общей части фрагменты (в случае умножения - частные от деления одночленов на их наибольший общий делитель). Такое усиление идентификации вводится по умолчанию. В тех редких случаях, когда оно не нужно и лишь замедляет работу приема, применяется указатель идентификации "набороперандов($K_1 \dots K_n$)". Здесь $K_1, \dots, K_n$ - указатели вхождений в теорему некоторых ассоциативно - коммутативных операций, обрабатываемых компилятором без применения указанных процедур.

2. Приведем еще два способа отключения указанной выше усиленной идентификации. Первый из них связан с идентификацией логического символа A - операнда ассоциативно - коммутативной операции f. Указатель идентификации "логсимвол($A\ f$)" означает, что A должно быть усмотрено "в чистом виде"

как операнд этой операции. Если в случае умножения не применять данный указатель, то прием будет предпринимать попытку деления соответствующего одночлена на константу A. Второй способ - при идентификации переменной x из пересечения наборов операндов нескольких одноименных ассоциативно - коммутативных операций использовать указатель "операнды(x)" вместо явного перечисления в указателе "набороперандов(...)" вхождений этих операций.

3. Если требуется идентифицировать переменную X с результатом $f(B_1 \dots B_m)$ выделения среди операндов $A_1, \dots, A_n$ ассоциативно - коммутативной операции f подмножества всех операндов B_i, удовлетворяющих заданному условию, то используется указатель "перечень(X $P(X)$)", где P - условие на операнд B_i, в котором этот операнд обозначен посредством самой переменной X (это условие сформулировано на том же языке, на котором определяются фильтры приемы). При использовании данного указателя предполагается, что переменная X имеет ровно одно вхождение в идентифицируемую часть - в качестве операнда некоторой ассоциативно - коммутативной операции f. В качестве примера рассмотрим теорему $a + b = c \leftrightarrow a = c - b$, используемую для перенесения в правую часть уравнения всех известных слагаемых b. Указатель "перечень(a не(известно(a)))" определяет идентификацию переменной a с суммой всех слагаемых, содержащих неизвестные; тогда сумма остальных слагаемых идентифицируется с b.

4. Если переменная X имеет несколько вхождений в идентифицируемую часть, причем все они суть операнды различных ассоциативно - коммутативных операций с одним и тем же заголовком f, то возможна идентификация ее в виде $f(B_1 \dots B_m)$, где $B_1, \dots, B_m$ - пересечение всех подмножеств операндов B указанных операций, удовлетворяющих условию $P(B)$. Для этого используется указатель "выписка(X $P(X)$)". В качестве примера рассмотрим теорему $ab + ac = a(b + c)$, применяемую в нормализаторе выявления повторяющихся вхождений выражений с неизвестными и обеспечивающую вынесение за скобку лишь известных общих множителей слагаемых. Вынесение за скобку неизвестного множителя в таком нормализаторе может разрушить произведение с неизвестными сомножителями, встречающееся повторно в другой части преобразуемого терма, и поэтому нежелательно. Здесь используется указатель "выписка(a известно(a))".

5. Возможна идентификация переменной X с коэффициентом, возникающим после приведения нескольких "подобных членов". Эта переменная должна иметь единственное вхождение в идентифицируемую часть теоремы, причем внутри терма $f(\dots g(X\ A)\dots)$, где f, g - ассоциативно - коммутативные операции; A - некоторый терм. При идентификации значением X становится терм вида $f(B_1 \dots B_n)$, где $g(A\ B_1), \dots, g(A\ B_n)$ - все операнды операции $f(\dots g(X\ A)\dots)$, множество g - членов которых включает множество g - членов терма A. Указатель идентификации здесь имеет вид "группировка(X)". В качестве примера рассмотрим идентификацию квадратного трехчлена $ad^2 + bd + c$ при разложении на множители. Сначала идентифицируется выражение d^2, и тем самым - основание степени d, а затем группируются в ad^2 все слагаемые с d^2 и в bd - все слагаемые с d. Используются указатели "группировка(a)", "группировка(b)".

6. В случае обычных сложения и умножения идентификация коэффициента X, возникающего после приведения подобных членов, может быть усилена таким

образом, чтобы коэффициенты извлекались также из дробных слагаемых (например, коэффициент $2/3$ из слагаемого $2A/3$). Переменная X должна иметь единственное вхождение в идентифицируемой части - в слагаемом вида AX либо AX/B. Она идентифицируется после идентификации A и B, как сумма коэффициентов при A (соответственно, при A/B) всех слагаемых рассматриваемой суммы, представимых в таком виде. Указатель идентификации в данном случае имеет вид "коэффициент(X K)", где K - указатель вхождения в теорему слагаемого с переменной X.

7. Иногда встречается несколько другая ситуация - требуется выделить группу "подобных членов", общий коэффициент K которых уже известен. В этом случае внутри идентифицируемой части теоремы выделяется терм вида $g(\ldots f($ X $A)\ldots)$, где f, g - ассоциативно - коммутативные символы. Перед идентификацией операнда $f(X\ A)$ терм, идентифицированный с X, представляется в виде $g(t_1 \ldots t_n)$, и указанный операнд идентифицируется с группой операндов $f(t_1\ A), \ldots, f(t_n\ A)$ операции g. Указатель идентификации здесь имеет вид "группировки(X K)", где K - указатель вхождения в теорему операнда $f(X\ A)$. В качестве примера рассмотрим теорему $ab + bd = (a + d)b$, применяемую при разложении на множители в ситуации, когда выражение b, идентифицированное как сомножитель слагаемого ab, имеет вид суммы $b_1 + \ldots + b_n$, а выражение bd составляется при идентификации из слагаемых $b_1 d, \ldots, b_n d$. Используется указатель "группировки(b фикс(0 1 2))".

8. Если вместо операнда $f(X\ A)$ в предыдущем случае рассматривается операнд вида $f(X)$, который (уже после идентификации X с $g(t_1 \ldots t_n)$) должен быть идентифицирован с группой $f(t_1), \ldots, f(t_n)$ операндов операции g (при формировании этих операндов двойные применения операции f отбрасываются, т.е. предполагается истинным тождество $f(f(x)) = x$, то используется указатель идентификации "нормзнака(X f g)". В качестве примера рассмотрим теорему $a+b = 0\ \&\ a-b = 0 \leftrightarrow a = 0\ \&\ b = 0$, применяемую для тривиального упрощения системы двух уравнений. Сначала идентифицируется a, как общая часть групп слагаемых левых частей этих уравнений. Это позволяет идентифицировать b из первого уравнения - как сумму оставшихся слагаемых, а затем идентифицировать $-b$ во втором уравнении как сумму этих же слагаемых, взятых с обратными знаками. Используется указатель "нормзнака(b минус плюс)".

9. Если переменную X требуется идентифицировать с отдельным операндом ассоциативно - коммутативной операции f, не прибегая к каким-либо группировкам, то используется указатель идентификации "операнд(X K)". Здесь K - указатель вхождения в теорему данной операции. Если в идентифицируемой части теоремы переменная X имеет единственное вхождение (именно, как операнда рассматриваемой операции), то возможно указание вместо K символа операции f. В качестве примера рассмотрим теорему $b \le 0 \rightarrow -b|a| = |ab|$, используемую для вынесения из-под модуля неположительных сомножителей. Если не ввести указатель "операнд(b фикс(0 2 1))", то компилятор может идентифицировать с сомножителем не b, а переменную a, в то время как b, для которого и проверяется неположительность, будет идентифицироваться с произведением оставшихся сомножителей.

10. Если переменная X встречается в качестве операнда нескольких ассоциативно

- коммутативных операций с общим заголовком f, то она по умолчанию идентифицируется как результат применения f к списку всех общих операндов этих операций, оставшихся после идентификации более явно обозначенных операндов. Можно, однако, выделить лишь подмножество M указанных ассоциативно - коммутативных операций, из рассмотрения которых и будет идентифицироваться X - как пересечение только их остаточных операндов. Для этого используется указатель идентификации "пересечение(X $K_1 \dots K_n$)", где $K_1, \dots, K_n$ - указатели вхождений в теорему операций подмножества M. В качестве примера рассмотрим теорему $ca^2 - 2cab + cb^2 + da - db = (a - b)(d + ac - cb)$, применяемую для разложения на множители. Переменную c здесь достаточно идентифицировать из рассмотрения слагаемых ca^2, cb^2, что и достигается с помощью указателя "пересечение(c фикс(0 1 1) фикс(0 1 3))".

11. Иногда при идентификации ассоциативно - коммутативной операции $f(A_1 \dots A_n)$ бывает нужно идентифицировать некоторое A_i только с первым операндом соответствующей операции из идентифицируемого терма. Например, это может быть полезным при заменах рекурсивного типа - для отсечения попыток применить рекурсию с участием внутреннего операнда, если она не удалась для первого операнда. Здесь используется указатель идентификации "первыйтерм(K)"; K - указатель вхождения в теорему операнда A_i. В качестве примера рассмотрим теорему "нечетнаяфункция($\lambda_x(f(x), h(x))$) & нечетнаяфункция ($\lambda_x(g(x), h(x))$) $\rightarrow$ нечетнаяфункция($\lambda_x(f(x) + g(x), h(x))$)", применяемую для усмотрения нечетной функции, являющейся суммой нечетных функций. Нет смысла перебирать всевозможные перестановки слагаемых - достаточно провести рекурсию, рассматривая их лишь последовательно слева направо. Для этого служит указатель "первыйтерм(фикс(0 1 1))".

12. Если в приеме замены заголовком заменяемого терма служит конъюнкция либо дизъюнкция, то по умолчанию она идентифицируется так, чтобы эту конъюнкцию либо дизъюнкцию можно было усмотреть из внешнего квантора общности (возможно, при перенесении некоторых антецедентов в консеквент) либо из внешнего отрицания двойственной логической связки. Чтобы заблокировать такой режим идентификации, используется указатель "дизъюнктоперанд". В качестве примера рассмотрим теорему $a \in b \cup c \leftrightarrow a \in b \vee a \in c$, используемую для группировки известных выражений b, c при неизвестном a в условии задачи на описание. Такая группировка внутри квантора общности либо конъюнкции отрицаний принадлежности малополезна, и для отсечения этих случаев служит указатель "дизъюнктоперанд".

Одновременное изменение знаков операндов ассоциативно - коммутативной операции

Если идентифицируется выражение вида $f(A_1 \dots A_n)$, где f есть символ ассоциативно - коммутативной операции, причем одноместная операция g удовлетворяет тождеству $g(g(x)) = x$, то при идентификации можно разрешить одновременное изменение "знака" g у всех операндов $A_1, \dots, A_n$. В тех случаях, когда указанное выражение является заменяемой частью некоторого тождества, при формировании заменяющего выражения также будет происходить одновременная замена знака g у всех f - членов. В противном случае данная альтернативная идентификация на формировании заменяющих термов никак не сказывается. Указателем идентификации служит терм

"знаксуммы(g K)", где K есть указатель вхождения в теорему выражения $f(A_1 \dots A_n)$. Возможно применение этого указателя внутри фильтра "контекст(...)" - тогда в качестве K берется само выражение $f(A_1 \dots A_n)$. В качестве примера рассмотрим теорему $a^2+b^2-2ab=(a-b)^2$, применяемую для разложения на множители. Чтобы прием позволял разложить на множители также выражение $2ab-a^2-b^2$, вводится указатель "знаксуммы(минус фикс(0 1))".

Идентификация кванторов и описателей

1. В задачах обычно применяется стандартизация, при которой кванторная импликация с конъюнктивным консеквентом преобразуется в конъюнкцию нескольких кванторных импликаций. Все эти кванторные импликации имеют один и тот же список антецедентов. Чтобы при идентификации автоматически выполнять "сборку" ранее расформированных кванторных импликаций, используется указатель "консеквент(A)", где A - указатель вхождения в теорему консеквента идентифицируемой кванторной импликации. Этот консеквент будет идентифицироваться с конъюнкцией консеквентов всех кванторных импликаций из области идентификации, имеющих заданный список антецедентов (к моменту идентификации кванторной импликации все переменные ее антецедентов должны быть уже идентифицированы). Единственный прием, где пока используется такой указатель - доказательства индукцией по длине набора. К моменту применения этого приема в посылках задачи должны иметься кванторные импликации, определяющие априори известные свойства i - го элемента набора, которые и группируются указанным выше образом.

2. Если теорема приема замены представляет собой эквивалентность с квантором K в заменяемой части, то можно усилить идентификацию, разрешая выделять K либо отрицание K из квантора с более длинной связывающей приставкой путем группировки всех подкванторных утверждений, зависящих от переменных некоторого подмножества этой приставки. Такое усиление обеспечивается указателем "кванторнаясвертка". В качестве примера рассмотрим теорему $a \subseteq b \leftrightarrow \forall c(c \in a \rightarrow c \in b)$, применяемую для перехода от кванторной импликации к условию включения множеств. При наличии указателя "кванторнаясвертка" прием будет выполнять такой переход в утверждении вида $\forall xy(x \in y \ \& \ y \in a \rightarrow x \in b)$, преобразуя его в утверждение $\forall y(y \in a \rightarrow y \subseteq b)$ с помощью перегруппировки $\forall y(y \in a \rightarrow \forall x(x \in y \rightarrow x \in b))$.

3. По умолчанию компилятор ГЕНОЛОГа организует идентификацию заменяемого квантора с попыткой усмотреть его отрицание в противоположном кванторе. Если такая идентификация нежелательна, то используется указатель "внешнийквантор(A)", где A - указатель вхождения данного квантора в теорему. Примеры использования указателя - приемы редактирования параметрических описаний решений тригонометрических уравнений либо неравенств, где изначально предполагается использование именно квантора существования и нет смысла тратить время на анализ возможных вхождений в задачу кванторов общности.

4. Если в теореме приема встречается квантор либо описатель, то компилятор обеспечивает проверку невхождения переменных его связывающей приставки в термы, идентифицированные с подкванторными переменными - обычными

либо функциональными, т.е. выражениями вида $f(x)$, не зависящими от этой связывающей приставки. Эту проверку можно отменить, используя указатель "обобщподст(A)", где A - указатель вхождения рассматриваемого квантора либо описателя в теорему. В качестве примера рассмотрим теорему приема $\neg(b = 0) \to set_x(a/b + c = 0\ \&\ A(x)) = set_x(a + bc = 0\ \&\ A(x))$, применяемую для устранения знаменателей в уравнениях кривых либо плоскостей. Чтобы не перегружать теорему обозначениями $a(x), b(x), c(x)$, вместо них используются a, b, c, но зато введен указатель "обобщподст(фикс(0 1))".

5. Указанную выше отмену проверки невхождения переменных связывающей приставки можно сделать частичной. Указатель идентификации "содержится($x\ y_1 \ldots y_n$)" выделяет те переменные $y_1, \ldots, y_n$, встречающиеся под квантором либо описателем по переменной x, которые могут идентифицироваться с термами, содержащими переменную, идентифицированную с x.

6. Другой способ сделать отмену проверки невхождения переменных связывающей приставки частичной - указатель "Входит(x)", определяющий ту переменную x связывающих приставок, для которой не предпринимается проверка невхождения идентифицированных с ней переменных в термы, идентифицированные с прочими переменными под кванторами и описателями.

7. При идентификации кванторов и описателей иногда бывает удобно выделить ряд конкретных переменных в связывающей приставке, а все оставшиеся переменные сгруппировать в набор, обозначаемый в теореме приема единственной (условной) переменной x. Такой набор, в частности, может оказаться пустым. Данный режим вводится указателем идентификации "кортежпеременных(x)". Заметим, что список выделяемых "штучным образом" переменных может быть пустым; в случае его непустоты такие переменные идентифицируются в связывающей приставке без учета порядка. В качестве примера рассмотрим теорему $set_x((f(x) \vee g(x))\ \&\ h(x)) = set_x(f(x)\ \&\ h(x)) \cup set_x(g(x)\ \&\ h(x))$, используемую для перехода от дизъюнкции в условии принадлежности множеству к объединению множеств. Прием имеет указатель "кортежпеременных(x)", что позволяет ему обрабатывать описания множеств наборов произвольной длины.

8. Чтобы идентифицировать теоремную переменную x с произвольным элементом связывающей приставки, без учета порядка ее переменных, применяется указатель "элемент(x)".

9. В теореме приема допускается использовать запись вида $\forall_x(f(x))$. Если сопроводить ее указателем "импликация(f)", то выражение $f(x)$ идентифицируется со вспомогательным термом "то($A_0\ A_1 \ldots A_n$)", где A_0 - консеквент; $A_1, \ldots, A_n$ - антецеденты идентифицирующей кванторной импликации, расположенной в задаче. В заменяющем терме выражение $f(B)$ может быть использовано только непосредственно под квантором, где оно расформируется на антецеденты и консеквент. Данный указатель применяется лишь в приемах вывода теорем, обычно в сочетании с указателем "вхождение(f)".

10. Если внутри теоремы приема встречается кванторная импликация $\forall_x(A_1\ \&\ \ldots\ \&\ A_n \to A_0)$, у которой некоторое A_i представляет собой функциональную переменную $F(x)$ либо обычную переменную F, и нужно идентифицировать его с отдельным антецедентом идентифицирующей кванторной импликации,

то используется указатель "антецедент(F P)". Здесь P - указатель вхождения в теорему данной кванторной импликации.

11. Иногда бывает удобно выделить в идентифицируемом кванторе единственную переменную x связывающей приставки, а остальные переменные (которые могут иметься либо отсутствовать) вообще не указывать. Если в теореме формируются новые кванторные конструкции с x, то все отличные от x переменные исходной связывающей приставки переносятся в них по умолчанию. Для такого режима служит указатель "связка(x)".

Идентификация функциональных переменных

В теореме приема могут встречаться подтермы вида $f(x), f(x,y), f(x,y,z), \ldots$, где $f, x, y, z, \ldots$ - переменные. Во внутреннем логическом представлении (в скобочной записи) они соответствуют выражениям "значение(f x)", "значение(f набор(x y))", "значение(f набор(x y z))", ... Такие выражения называем функциональными переменными. Для идентификации функциональной переменной предусмотрены перечисляемые ниже указатели; при отсутствии этих указателей выражения "значение(f ...)" идентифицируются с имеющими заголовок "значение" термами задачи обычным образом.

1. Наиболее часто встречающийся случай - идентификация терма $f(x_1 \ldots x_n)$ с произвольным подтермом задачи T (здесь n может равняться 1, причем переменные $x_1, \ldots, x_n$ попарно различны). Если переменные $x_1, \ldots, x_n$ согласно другим фрагментам теоремы окажутся идентифицированы с некоторыми переменными задачи $y_1, \ldots, y_n$, то далее T начинает играть роль "шаблона": остальные подтермы теоремы вида $f(t_1 \ldots t_n)$, если они есть, при идентификации либо формировании новых термов получаются как результат подстановки в T вместо переменных $y_1, \ldots, y_n$ термов, определяемых по результатам идентификации термами $t_1, \ldots, t_n$. Указатель на идентификацию данного типа имеет вид "отображение(f)". Рекомендуется объединять несколько различных указателей такого типа в общий указатель "отображение(f $g \ldots h$)". Заметим, что если в теореме f встречается только в виде $f(x)$, то x может (до идентификации $f(x)$) быть идентифицировано не с переменной, а со списком переменных (например, со связывающей приставкой квантора либо описателя). В качестве примера рассмотрим теорему $c = \lim_{x\to a} f(x)$ & число(c) & $d = \lim_{x\to a} g(x)$ & число(d) $\to \lim_{x\to a}(f(x) + g(x)) = c + d$, используемую при вычислении предела суммы, если пределы слагаемых существуют и конечны. Наличие указателя "отображение(f g)" позволяет идентифицировать $f(x), g(x)$ с произвольными выражениями, не обязательно даже содержащими переменную x.

2. Если при идентификации терма $f(t)$ все переменные аргумента t уже идентифицированы так, что этот аргумент задает терм T "в переменных задачи", то можно выделить в текущем (идентифицируемом с $f(t)$) терме A все вхождения $v_1, \ldots, v_n$ подтерма T, и считать, что остальные вхождения в теорему термов $f(p)$ задают результаты замены в A вхождений $v_1, \ldots v_n$ на определяемый аргументом p терм P. Такой режим вводится указателем "заменатермов(K)", где K - указатель вхождения в теорему терма $f(t)$. В качестве примера рассмотрим теорему $f(y) = a \to (a$ при $x = y$, иначе $f(x)) = f(x)$, используемую для исключения условного выражения, у которого особый случай $x = y$ сводится

к общему случаю. Указатель "заменатермов(фикс(0 1 3))" позволяет рассматривать выражение, идентифицированное с $f(x)$, как шаблон для построения $f(y)$.

3. Если необходимо потребовать, чтобы идентифицируемый с теоремным термом $f(t)$ терм задачи A содержал переменную X только внутри вхождений терма T, определяемого аргументом t, то используется указатель "новаргумент(f x N)". Предполагается, что к моменту идентификации $f(t)$ все переменные из t уже идентифицированы, а теоремная переменная x идентифицирована с переменной задачи X. N - либо логический символ "фикс", либо название нормализатора, применяемого к A перед попыткой выделения вхождений терма T. Такой нормализатор в качестве входного комментария получает набор (новаргумент T X); он пытается преобразовать "неявные" вхождения терма T в терм A в явные. Если N = "фикс", то A используется без преобразований. После проведения идентификации терма $f(t)$ остальные вхождения в теорему термов $f(p)$ определяют результаты замены всех вхождений T в преобразованный нормализатором N терм A на терм P, определяемый аргументом p. В качестве примера рассмотрим теорему

$$\int \frac{(1-x^2)^{\frac{a-1}{2}} f(x)}{g(x)} dx = \lambda_x(h(x), \text{число}(x)) \to \int \frac{(\sin x)^a f(\cos x)}{g(\cos x)} dx =$$

$$= \lambda_x(-h(\cos x), \text{число}(x)),$$

позволяющую вычислять неопределенный интеграл путем замены $\cos x = y$. Указатели "новаргумент(f x извлечение)", "новаргумент(g x извлечение)" определяют попытки преобразования выражений A, B, идентифицируемых с $f($ $\cos x)$, $g(\cos x)$, к виду, в котором переменная интегрирования x встречалась бы только как $\cos x$. Для этого применяется нормализатор "извлечение", имеющий (в том числе) ряд несложных приемов перехода к косинусам.

4. Указатель "новаргумент", введенный в предыдущем пункте, допускает обобщение на случай идентификации "многоместного" теоремного терма $f(t_1 \dots t_n)$. Здесь он уже имеет вид "новаргумент(f набор($x_1 \dots x_m$) N)". Проверяется, что идентифицируемое с указанным термом выражение A содержит переменные, идентифицированные с переменными $x_1, \dots, x_m$, только внутри вхождений термов $t_1, \dots, t_n$, все переменные которых заранее идентифицированы. В случае $m = 1$ символ "набор" перед переменной x_1 отбрасывается. После идентификации формирование термов $f(p_1 \dots p_n)$ осуществляется на основе выделения в A вхождений термов $t_1, \dots, t_n$.

5. Еще одно обобщение указателя "новаргумент" связано с идентификацией, допускающей переобозначения связанных переменных. Это обобщение используется крайне редко (пока - лишь в приемах, связанных с решением дифференциальных уравнений порядка выше первого). По сравнению с предыдущим обобщением, здесь добавляется еще список $a_1, \dots, a_n$, каждый элемент которого - либо переменная, либо 0. Если a_i - переменная, то при выделении вхождения выражения t_i в терме A разрешается переобозначать ее внутри этого вхождения на переменную, связанную внешним по отношению к данному вхождению квантором либо описателем. Если $a_i = 0$, то никакие переобозначения внутри t_i

не допускаются. После идентификации формирование термов $f(p_1 \dots p_n)$ происходит с переобозначением переменных a_i на те же связанные переменные. Указатель, вводящий данный режим идентификации, имеет вид "функаргумент(f набор($x_1 \dots x_m$) набор($a_1 \dots a_n$) N)". В качестве примера ситуации, где он может понадобиться, рассмотрим идентификацию терма $f(g'(x))$ с термом, имеющим вид второй производной $g''(x)$. Нормализатор N преобразует вторую производную к виду повторной производной $g'(g'(x))$. На логическом языке эта запись выглядит как "производная(отображение(y число(y) производная(отображение(z число(z) $g(z)$)y))x)". Внутренняя производная здесь берется в точке, обозначенной некоторой вспомогательной переменной y, связанной внешней производной. Поэтому при идентификации этой производной с $g'(x)$ понадобится переобозначение x на y.

6. В ряде случаев требуется идентифицировать функциональную переменную с символом операции либо отношения. Здесь терм $f(t)$ либо $f(t_1 \dots t_n)$ (в скобочной записи последний представлен как "значение(f набор($t_1 \dots t_n$))") идентифицируется с термом $F(T)$ либо $F(T_1 \dots T_n)$, где F - логический символ. После этого остальные "теоремные" термы вида $f(r)$,$f(r_1 \dots r_n)$ определяют термы $F(R)$,$F(R_1 \dots R_n)$. Указателем на данный режим идентификации служит "символ(f)". Обычно этот указатель используется в приемах текстового анализатора либо в приемах логического вывода базы теорем.

7. Если выражение $f(A)$ должно идентифицироваться с многочленом от A, то используется указатель идентификации "сммногочлен(f S)". Здесь S указывает на способ определения выражения A при идентификации. Если S - символ "неизвестные", то A выявляется из рассмотрения всех содержащих неизвестные задачи или пакетного оператора множителей одночленов; если S есть теоремная переменная, то A выявляется из всех содержащих идентифицированную с S переменную множителей одночленов; если S имеет вид "терм(T)", то A определяется операторным выражением T. Наконец, S вообще может отсутствовать, и тогда A выявляется из рассмотрения всех неконстантных множителей одночленов. Во всех перечисленных случаях (кроме случая явного указания A посредством записи "терм(T)") идентификация $f(A)$ определяет как многочлен, так и выражение A. После такой идентификации остальные встречающиеся в теореме термы вида $f(B)$ определяют многочлены с теми же коэффициентами, что и $f(A)$, примененные к выражению B. Если для выделенных указателями "сммногочлен(a x)", "сммногочлен(f x)"переменных a, f имеется антецедент теоремы $a = \lambda_x(bx^c + f(x))$, причем этот антецедент - идентифицирующий (см. следующий раздел), то переменная c будет идентифицироваться со степенью многочлена, переменная b - с коэффициентом при старшем члене, а функциональная переменная $f(x)$ - с суммой оставшихся членов. В этом случае указатель "отображение(f)" не нужен. Заметим, что здесь и далее в указателях ГЕНОЛОГа под многочленами понимаются термы, имеющие вид многочленов, а не формальные многочлены, рассматриваемые в алгебре.

8. В некоторых случаях переменную f, идентифицированную обычным образом с термом A (т.е. с использованием такого ее вхождения в теорему, которое не имеет вида $f(t)$), бывает необходимо использовать после этого как функциональную переменную. Выражения $f(r)$ либо $f(r_1 \dots r_n)$ при этом будут определять результаты замены в A некоторых переменных y либо $y_1, \dots, y_n$ на

термы R либо $R_1, \ldots, R_n$, соответствующие $r, r_1, \ldots, r_n$. Чтобы ввести такой режим, используется указатель идентификации "функция($f\ x$)" либо "функция(f набор($x_1 \ldots x_n$))". Теоремные переменные $x, x_1, \ldots, x_n$ должны быть идентифицированы до формирования выражений $f(r), f(r_1 \ldots r_n)$ независимым образом; они определяют указанные выше переменные задачи $y, y_1, \ldots, y_n$.

9. Иногда бывает нужно идентифицировать с заданным теоремным выражением t какой-то произвольный подтерм заданного терма задачи T , безотносительно к тому, где этот подтерм расположен внутри T. Тогда в теореме используется функциональная запись $P(x, t)$, где P - вспомогательная переменная, в сочетании с указателем "вхождение(P)". Здесь переменная x заранее идентифицирована с переменной либо со списком переменных. Теоремный терм $P(x, t)$ идентифицируется с термом задачи T, а t - с некоторым вхождением v в T. В заменяющих фрагментах теоремы запись $P(A, B)$ далее используется для обозначения результата замены выделенного внутри T вхождения v на B, с одновременной подстановкой вместо всех остальных (не расположенных внутри v) вхождений переменной либо переменных x терма либо термов A. Чтобы в этой ситуации сослаться на результат подстановки терма A вместо x в конъюнкцию всех утверждений из области вхождения v, используется запись $P(A)$. Если в теореме не используется нетождественная подстановка A вместо x (а такие невырожденные подстановки появляются обычно в приемах вывода теорем), то вместо $P(x, t)$ может быть использована запись $P(t)$.

10. Указатель "функподст($f\ g$)" позволяет идентифицировать теоремные термы вида $f(g(A), B)$ либо $f(g(A))$. Предварительно переменная g должна быть идентифицирована с некоторой переменной G. Проверяется, что идентифицирующий терм T содержит переменную G только в виде "значение($G \ldots$)". После идентификации формирование заменяющих термов $f(C, B)$ либо $f(C)$ происходит путем замены всех вхождений выражений "значение($G \ldots$)" в T на C. Данная идентификация применяется в приемах вывода теорем.

11. Если нужно идентифицировать терм вида $f(A, B)$ с термом вида $F(X, Y)$ либо $G(F(X, Y))$, где F, G - логические символы операций либо отношений, то применяется указатель "Бинарнаяоперация(f)". A, B идентифицируются с X, Y без учета порядка этих операндов. Переменная f, рассматриваемая отдельно, считается идентифицированной с символом F.

12. Если нужно идентифицировать терм вида $f(A_1 \ldots A_n)$, у которого f - некоторая операция либо отношение, причем лишь единственный операнд A_i содержит заданную переменную x, то в теореме приема можно использовать запись $F(A)$, сопровождаемую указателем "внешфункция(F, X)". Здесь переменная F идентифицируется с логическим символом f, переменная X заранее должна быть идентифицирована с x, а выражение A будет идентифицироваться с тем операндом, который содержал x.Соответственно, термы вида $F(B)$ будут формироваться как результаты замены найденного операнда A_i на B.

Идентификация операций над конечными семействами и кванторных импликаций для конечных областей

1. Пусть одноместная операция F над семейством объектов возникла из двуместной ассоциативно-коммутативной операции f (например, "суммавсех" либо

"произведениевсех"из двуместных сложения и умножения). Выражение F(отображение(i и(целое(i) меньшеилиравно(1 i)меньшеилиравно(i n))$P(i)$)), определяющее результат применения данной операции к конечному семейству (формульный редактор прорисовывает такое выражение, например, в случае суммирования, как

$$\sum_{i=1}^{n} P(i)),$$

можно идентифицировать с термом задачи вида $f(A_1 \ldots A_n)$ (в случае суммирования - с суммой $A_1 + \ldots + A_n$). Для этого используется указатель "развертка(K)", где K - указатель вхождения в теорему данного выражения $F(\ldots)$. Предпринимается последовательная идентификация $P(i)$ с $A_1, \ldots, A_n$. При этом еще не идентифицированные переменные g, встречающиеся внутри терма $P(i)$ в виде $g(i)$, оказываются идентифицированы с термами "набор($B_1 \ldots B_n$)"; здесь B_j - терм, с которым была идентифицирована $g(i)$ при рассмотрении A_j. Несколько различных указателей "развертка(K_1)", ..., "развертка(K_m)" можно объединять в общий указатель "развертка($K_1 \ldots K_m$)". Чтобы при идентификации допустить вырожденный случай $n = 0$, используется дополнительный указатель "пустоеслово(K)". В качестве примера рассмотрим теорему

$$\forall_i(i \in \{1, \ldots, n\} \rightarrow \text{конечное}(B(i))) \ \& \ A = \bigcup_{i=1}^{n} B(i) \rightarrow \text{конечное}(A),$$

применяемую в проверочном операторе усмотрения конечного множества. Указатель "развертка(фикс(2 2))" позволяет идентифицировать второй антецедент с произвольным равенством вида $A = B_1 \cup \ldots \cup B_n$; при этом одновременно идентифицируются число операндов n и набор B.

2. Указатель "развертка(K)" можно использовать не только для уточнения режима идентификации, но и для того, чтобы новый терм согласно выражению F(отображение(...)) формировался как $f(A_1 \ldots A_n)$. Допускается объединение таких указателей формирования новых термов с указателями идентификации в общую запись "развертка(...)". Примером может служить теорема

$$\text{периметр}(\text{фигура}(x)) = \sum_{i=1}^{n} l(x(i)x(i(\mathrm{mod}n) + 1)),$$

используемая для перехода к выражению периметра многоугольника, заданного набором x своих вершин, в виде сумм длин его сторон. Здесь используется указатель "развертка(фикс(0 2))", обеспечивающий формирование "конкретной" суммы длин.

3. Следующий способ применения указателя "развертка(K)" - для идентификации терма "отображение(i принадлежит(i номера(1 n)) $P(i)$)" с термом "набор($A_1 \ldots A_n$)", определяющим конечный набор, либо для формирования второго терма по первому после идентификации из других источников всех его переменных. Заметим, что формульный редактор прорисовывает терм "отображение(...)" в виде $\lambda_i(P(i), i \in \{1, \ldots, n\})$.

4. Особо выделен случай двуместной коммутативно - ассоциативной связки "и" - аналогом этой связки для семейства утверждений служит кванторная импликация "длялюбого(i если принадлежит(i номера(1 n)) то $P(i)$)". Формульный

редактор прорисовывает ее в виде $\forall_i(i \in \{1,\ldots,n\} \to P(i))$. Если такая импликация представляет собой антецедент теоремы приема, то указатель идентификации "развертка(K)" определяет режим поиска в контексте идентификации группы утверждений $A_1,\ldots,A_n$, идентифицируемых последовательно с $P(i)$. Функциональные переменные $g(i)$ из $P(i)$ при этом идентифицируются так же, как в случае термов F(отображение(...)). Указатель "развертка(K)" может ссылаться и на определяющую новый терм кванторную импликацию приведенного выше вида, которая в этом случае порождает конъюнкцию $P(1)$ & ... & $P(n)$. Наконец, рассматриваемая кванторная импликация, являющаяся антецедентом теоремы, может быть выделена, кроме указателя "развертка(K)", также некоторым указателем обработки антецедента (см. следующий раздел), определяющим режим проверки этого антецедента при помощи пакетного оператора либо вспомогательной задачи. В этом случае к моменту проверки должны быть идентифицированы все переменные антецедента, и проверка будет заключаться в последовательности обращений к указанным оператору либо задаче для $i = 1,\ldots,n$.

5. Возможна идентификация многочлена, определенного с помощью выражения "суммавсех(отображение(i и(целое(i)меньшеилиравно(1 i)меньшеилиравно(i n))умножение(значение(A i)степень(x плюс(i минус(1))))))". Это выражение прорисовывается формульным редактором как

$$\sum_{i=1}^{n} A(i)x^{i-1}.$$

Здесь используется указатель идентификации "Многочлен(K)", где K - указатель вхождения в теорему рассматриваемого выражения. Переменная A при этом идентифицируется с термом "набор($C_1 \ldots C_n$)", задающим коэффициенты многочлена C_i, упорядоченные по возрастанию степеней; переменная n - с увеличенной на единицу степенью многочлена (термом). Все переменные выражения x должны быть определены до идентификации данного многочлена.

Идентификация матриц

Матрицы задаются термами "строки(...)" либо "столбцы(...)", перечисляющими элементы матрицы, соответственно, по строкам либо по столбцам. Чтобы идентифицировать с такой матрицей выражение $\lambda_{ij}(A(i,j), i \in \{1,\ldots,m\}$ & $j \in \{1,\ldots,n\})$, используется указатель "матрица(S)", где S - указатель вхождения данного выражения в теорему. Такой же указатель применяется, если на основе приведенного выше выражения нужно создать новый терм, обозначающий матрицу. Допускается объединение нескольких указателей "матрица(...)" в один, перечисляющий сразу все рассматриваемые указатели вхождений в теорему S.

Если нужно идентифицировать матрицу из коэффициентов выражений заданного вида, встречающихся, например, в некоторой сумме, то для идентификации с суммой (или подобной многоместной операцией) используется выражение вида $F(\lambda_j(g(A(i,j), B), P(j)))$, сопровождаемое указателем "развертка(...)", где F - обобщение двуместной ассоциативно-коммутативной операции f (аналога сложения) на наборы произвольной длины. Добавление указателя "коэфф(A)"обеспечивает идентфикацию A как матрицы из коэффициентов $A(i,j)$. При отсутствии в сумме некоторых членов $g(...)$ значение $A(i,j)$ идентифицируется с нулем операции f.

Дополнительная идентификация

Некоторые из теоремных переменных, а также некоторые переменные, встречающиеся в фильтрах приема, могут быть идентифицированы за счет "внешних" по отношению к тексту теоремы средств. Для такой дополнительной идентификации используется указатель "контекст($A_1 \dots A_n$)", устроенный совершенно так же, как одноименный фильтр. Единственным отличием от фильтра здесь является то, что переменные, значения которых определяются в $A_1, \dots, A_n$ посредством идентифицирующих термов, имеют своей областью действия не только данный указатель "контекст(...)", как это было в случае фильтра, а все описание приема. Эти переменные, в частности, могут встречаться и в тексте теоремы, где они рассматриваются как уже идентифицированные. Такая переменная может отсутствовать в идентифицирующей части теоремы, но входить в ту ее часть, на основе которой формируются новые термы. В качестве простого примера рассмотрим теорему $\neg(i - j = 0) \to \text{card}(\text{set}_x(l(x) = 2 \;\&\; \text{слово}(x) \;\&\; x(i) = a \;\&\; P(x(j)))) = \text{card}(\text{set}_x(P(x)))$, используемую для сведения числа пар x, у которых одна компонента фиксирована, а другая удовлетворяет некоторому условию P, к числу объектов, удовлетворяющих условию P. Здесь выражение $x(j)$ при идентификации $P(x(j))$ должно быть уже идентифицировано. Для этого служит указатель "контекст(позиция(b теквхожд)вид(b значение(x j))не(равно(i j)))", который усматривает вхождение данного выражения внутри заменяемого терма, определяя таким образом значение j. Будет ли переменная x идентифицирована по данному указателю, или непосредственно из теоремы (где она легко усматривается из связывающей приставки в $\text{set}_x(\dots)$) - остается на усмотрение компилятора.

Формирование вспомогательных термов при идентификации

Если к моменту идентификации некоторого теоремного терма A все его переменные уже идентифицированы, то A определяет некоторый терм B, полученный из A подстановкой вместо переменных идентифицированных с ними термов. Тогда появляется возможность не "накладывать" при идентификации терм A на соответствующий терм задачи C, а сформировать вспомогательный терм B и затем проверить совпадение термов C и B. Если подтермы терма A были снабжены указателями нормализации, то фактически C будет сравниваться не с B, а с результатом применения к B тех или иных дополнительных преобразований. Данный режим идентификации включается при наличии указателя "подтерм(K)", где K - либо указатель вхождения в теорему терма A, либо сам этот терм. В качестве примера рассмотрим теорему "таблица$\{; \lambda_i(\text{конст}(a(i), b(i)), i \in \{1, \dots, n\})\}$+ таблица$\{; \lambda_i(\text{конст}(a(i), c(i)),$ $i \in \{1, \dots, n\})\} =$ таблица$\{; \lambda_i(\text{конст}(a(i), b(i) + c(i)), i \in \{1, \dots, n\})\}$". На ней основан прием сложения двух функций, заданных таблицами на одном и том же множестве точек $a(i), i = 1, \dots, n$. Эти точки идентифицируются по первому слагаемому, а при идентификации второго слагаемого выражения $a(i)$ рассматриваются как уже известные и просто сравниваются с идентифицирующими термами. Для такой идентификации служит указатель "подтерм(фикс(0 1 2 1 1 3 1))", ссылающийся на вхождение выражения $a(i)$ во второе слагаемое.

Идентификация квазиодноместных операций с помощью вспомогательных процедур

В некоторых разделах встречаются часто повторяющиеся одноместные операции $t(x)$, возникающие на основе многоместных при подстановке вместо части их операндов констант. Например, в элементарной алгебре таковыми являются константные натуральные степени аргумента x: $x^2, x^3, x^4, \ldots$. Для идентификации таких "квазиодноместных" операций $t(A)$ могут быть использованы специальные процедуры. Эти процедуры определяются справочником "вид" по заголовку терма $t(x)$, а сам справочник используется компилятором ГЕНОЛОГа таким образом, что по умолчанию всегда, когда это возможно, указанные специальные процедуры применяются. В частности, при идентификации константных натуральных степеней применяется процедура "выделениестепени".

Для блокировки использования справочника "вид" при идентификации подтермов, расположенных по указателям вхождения в теорему $K_1, \ldots, K_n$, применяется указатель идентификации "вид($K_1 \ldots K_n$)".

Если необходимо передать вспомогательным процедурам, вводимым справочником "вид" для идентификации подтермов по указателям вхождений в теорему $K_1, \ldots, K_n$, некоторый комментарий, то используется указатель "выделениестепени($K_1 \ldots K_n$ $A_1 \ldots A_m$)", где набор $A_1 \ldots A_m$ определяет данный комментарий (A_1 - логический символ, являющийся заголовком комментария; возможен случай $m = 0$).

Выделение общей части нескольких термов при помощи вспомогательных процедур

Если идентифицируется группа термов $B_1, \ldots, B_n$, содержащих общую переменную x, причем каждый из них, помимо x, имеет ровно одно вхождение единственной переменной (возможно, функциональной, т.е. выражения вида $f(t)$), не встречающейся более в идентифицируемой части теоремы, то для выделения "общей части" x этих термов, а также идентификации остальных входящих в них переменных, может быть применена реализованная на ЛОСе вспомогательная процедура P. Эта процедура должна иметь формат P(x1 x2 x3), где значением входной переменной x1 служит набор термов, идентифицируемых с $B_1, \ldots, B_n$, значением выходной переменной x2 - набор термов, идентифицированных с отличными от x переменными термов $B_1, \ldots, B_n$, а значением выходной переменной x3 - терм, идентифицированный с x. Указатель идентификации, определяющий данный режим, имеет вид "общаястепень(P x $B_1 \ldots B_n$)". Примером вспомогательной процедуры указанного типа служит процедура "общаястепень(x1 x2 x3)", предназначенная для представления группы выражений в виде степеней с общим показателем x (каждое выражение B_i представляется этой процедурой в виде произведения степеней; у каждой степени определяется рациональный коэффициент показателя, и находится частное наибольших общих делителей числителей и знаменателей этих коэффициентов). Эта процедура используется, например, в приеме с теоремой

$$0 \le b \;\&\; 0 \le d \;\&\; 0 \le a \;\&\; 0 \le e \;\&\; 0 < c \to (ab^c - ed^c = 0 \leftrightarrow a^{\frac{1}{c}}b - e^{\frac{1}{c}}d = 0),$$

осуществляющем извлечение корня степени c из обеих частей равенства. Величина c определяется как общий множитель показателей степени всех неизвестных сомножителей частей этого равенства. Таких сомножителей в каждой части может быть

несколько, причем каждый из них имеет вид степени. Для нахождения c вводится указатель "общаястепень(общаястепень c степень(b c) степень(d c))".

Идентификация наборов

Если в теореме встречается терм вида "префикс(A B)", то компилятор автоматически обеспечивает такой режим идентификации, при котором этот терм мог бы идентифицироваться как с выражением вида "набор(...)" (из него выбирается первый элемент для A, а остаток набора определяет B), так и с явным выражением вида "префикс(...)". Это же относится к термам вида "конкатенация(набор(A B ... C)D)" и "суффикс(A B)". Никаких специальных указателей при этом не используется. Фактически выделение нескольких первых либо последнего элемента набора обеспечивают операторы ЛОСа: "элементнабора", "элементномер", "Элементномер", "остатокнабора" - для уточнения подробностей идентификации либо ее развития следует обращаться к программам этих операторов.

Если в указанных выше случаях термы "префикс(A B)", "конкатенация(набор(A B ... C)D)", "суффикс(A B)" выделены указателем "список(K)" (K - указатель вхождения такого терма в теорему приема), то порядок расположения элементов в наборе при идентификации игнорируется. В частности, при идентфикации терма "префикс(A B)" будут последовательно предприниматься попытки идентифицировать с A каждый из элементов набора. В качестве примера использования указателя "список" рассмотрим теорему $b \in a \to \{b; c\}\backslash a = \{; c\}\backslash a$, на которой основан прием "вычитания" множеств, заданных конечными списками. Заметим, что обозначениям формульного редактора $\{b; c\}$ и $\{; c\}$ в скобочной записи соответствуют выражения "перечень(префикс(b c))" и "перечень(c)" соответственно. Чтобы поиск элемента b, отбрасываемого из перечня ввиду его принадлежности вычитаемому множеству a, происходил по всем элементам перечня, в приеме введен указатель "список(фикс(0 1 1 1))", выделяющий вхождение подтерма "префикс(b c)".

Группировка всех слагаемых неравенства в одной части

Для идентификации неравенства вида $0 < A$ либо $A < 0$ может быть использован режим с автоматической группировкой всех слагаемых в одной части (такой же режим может быть введен и для идентификации нестрогих неравенств). Здесь используется указатель "перегруппировка(B_1 ... B_n)", где $B_1, \ldots, B_n$ - указатели вхождений в теорему всех неравенств, требующих данного режима идентификации. В качестве примера рассмотрим теорему $0 < a \ \& \ c < 0 \ \& \ 0 < ax + b \ \& \ 0 < cx + d \to 0 < ad - bc$, на которой основан прием, выводящий из двух неравенств их линейную комбинацию, в которой исключена переменная x. Чтобы избежать создания приемов для различных случаев размещения x слева либо справа от знака неравенства, введен указатель "перегруппировка(фикс(3)фикс(4))".

Использование равенств из контекста идентификации

Если в теореме приема указано некоторое выражение A, которое при идентификации сравнивается с выражением B, не имеющим такого вида, то идентификация, все же, могла бы состояться, если бы в текущем контексте нашлась посылка $A = B$. Чтобы такие попытки усиления идентификации предпринимались, прием снабжается

указателем "смравно(P)", где P - указатель идентифицируемого вхождения выражения A в теорему приема.

Уточнение порядка, в котором происходит идентификация

Через указатели можно в некоторой степени влиять на порядок, в котором прием будет идентифицировать значения переменных. Это может оказаться полезным для получения менее трудоемкой (с бóльшим отсечением на ранних шагах) процедуры идентификации. К числу таких указателей, уточняющих порядок идентификации, относится указатель "определено(K x)". Он блокирует идентификацию подтерма - операнда ассоциативно - коммутативной операции, расположенного по указателю вхождения K, до тех пор, пока не будет идентифицирована переменная x. Некоторые другие указатели, от которых тоже зависит порядок идентификации, связаны с очередностью обработки антецедентов теоремы; они вводятся в следующем разделе.

11.4.2 Указатели обработки антецедентов теоремы

Антецеденты $A_1, \ldots A_n$ теоремы приема $\forall_{x_1 \ldots x_m}(A_1 \ \& \ \ldots \ \& \ A_n \rightarrow A_0)$ предполагаются занумерованными слева направо последовательными натуральными числами $1, 2, \ldots, n$. Эти номера используются в указателях обработки антецедентов для ссылок на антецеденты.

Непосредственная идентификация антецедентов

Если для обработки антецедента не предусмотрена специальная процедура, в соответствии с приводимыми далее указателями обработки, то прием пытается идентифицировать его с некоторым утверждением, содержащимся в контексте идентификации. Это называется непосредственной идентификацией антецедента.

Обычно контекст идентификации совпадает с областью вхождения в задачу точки привязки приема. Если прием был инициирован при рассмотрении вхождения в посылку, то такой контекст идентификации включает в себя все остальные посылки задачи; если прием был инициирован при рассмотрении вхождения в условие, то контекст идентификации включает в себя все остальные условия, а также все посылки задачи. Кроме того, в контекст идентификации будут входить относящиеся к области текущего вхождения подутверждения текущего терма задачи.

Можно уточнять подмножество контекста идентификации, из которого берется идентифицирующее заданный антецедент утверждение. Указатель "списокпосылок($i_1 \ldots i_n$)" перечисляет номера $i_1, \ldots, i_n$ тех антецедентов, которые идентифицируются с посылками задачи; указатель "списокусловий($i_1 \ldots i_n$)" - номера тех антецедентов, которые идентифицируются с условиями задачи.

Проверка истинности антецедента с помощью проверочного оператора

1. Указатель обработки "блокпроверок($i_1 \ldots i_n$)" указывает номера $i_1, \ldots, i_n$ тех антецедентов, для проверки истинности которых будут использоваться пакетные проверочные операторы. Компилятор сам определяет заголовки этих операторов, используя справочник "легковидеть". Каждый проверочный оператор

сопровождается указанием вида T утверждений, для проверки истинности которых он предназначен. При указании на обработку антецедента проверочным оператором необходимо, чтобы уже имелся оператор для проверки утверждений данного вида. Простейший пример использования проверочных операторов для обработки антецедентов - основанный на теореме $0 \leq a \ \& \ 0 \leq b \rightarrow (a + b = 0 \leftrightarrow a = 0 \ \& \ b = 0)$ прием декомпозиции уравнения с нулем в правой части и левой частью, представимой в виде суммы двух слагаемых одного знака. Для быстрого усмотрения неотрицательности слагаемого a и суммы остальных слагаемых b используется проверочный оператор "усмменьшеилиравно". Чтобы компилятор создал обращения к нему, вводится указатель "блокпроверок(1 2)".

2. Для некоторых типов утверждений предусмотрена усиленная версия проверочного оператора. Заголовок такого усиленного проверочного оператора определяется при помощи справочника "проверка", аналогичного справочнику "легковидеть". Для ссылки на антецеденты, обрабатываемые усиленными проверочными операторами, используется указатель "проверка($i_1 \ldots i_n$)"; $i_1, \ldots, i_n$ - номера этих антецедентов.

3. В ряде случаев бывает целесообразно в зависимости от контекста корректировать заголовок антецедента, указанный в теореме. Например, вместо проверки строгого неравенства может оказаться достаточной проверка нестрогого неравенства. Чтобы прием мог выбирать необходимый проверочный оператор для такого антецедента (в указанном случае - оператор "усмменьше" либо "усмменьшеилиравно"), используется указатель "замещение($P\ i\ A$)", который означает, что при выполнении условия P заголовок антецедента с номером i, обрабатываемого проверочным оператором, заменяется на A.

4. Иногда в качестве антецедента теоремы используется переменная (обычная либо функциональная, вида $f(x)$), которая идентифицируется из других фрагментов теоремы с некоторым утверждением. Организовать непосредственную проверку истинности такого утверждения проверочным оператором невозможно, так как априори вид его не известен. Однако, здесь можно воспользоваться оператором "очевидно(...)", который с помощью справочника "легковидеть" пытается определить по виду предъявленного ему для проверки утверждения необходимый проверочный оператор и переадресует ему дальнейшую проверку. Чтобы предпринять такую обработку для антецедентов с номерами $i_1, \ldots, i_n$, применяется указатель "очевидно($i_1 \ldots i_n$)". В качестве примера рассмотрим теорему "семействомножеств($\lambda_a(f(a), g(a))$) \& $g(x) \rightarrow \mathrm{set}(f(x))$)", на которой основан прием, усматривающий множество в элементе семейства множеств. Первый антецедент идентифицируется непосредственно; для обработки второго введен указатель "очевидно(2)".

Реализация антецедента с помощью пакетного синтезатора

Если группа антецедентов с номерами $i_1, \ldots, i_n$ должна обрабатываться одним общим пакетным синтезатором, то используется указатель обработки "значения($i_1 \ldots i_n$)". Компилятор определяет заголовок применяемого синтезатора с помощью справочника "значения". Как и в случае проверочного оператора, синтезатор предназначен для обработки утверждений лишь специального вида, причем обработка заключается в

нахождении значений выходных переменных, при которых эти утверждения становятся истинными. Вид списка $T_1, \ldots, T_n$ обрабатываемых утверждений определяется в описании формата синтезатора (см. справочник "синтезатор"). Среди переменных утверждений $T_1, \ldots, T_n$ выделены входные - они должны быть уже идентифицированы к моменту обработки указанных антецедентов, а также выходные - их значения будут определяться в результате применения синтезатора. Заметим, что на "выходных" позициях антецедентов, обрабатываемых синтезатором, могут стоять только обычные переменные (недопустимы даже функциональные переменные - выражения вида $f(x)$). В то время, как ссылки на антецеденты, обрабатываемые проверочными операторами, объединяются в общий указатель "блокпроверок(...)", каждое обращение к синтезатору требует своего указателя "значения(...)". В качестве примера обработки антецедентов синтезаторами рассмотрим прием, основанный на теореме $b \le a \ \& \ a \le c \ \& \ d = [b] \ \& \ 0 < d - c + 1 \rightarrow [a] = d$. Этот прием предпринимает попытку определения значения целой части путем получения верхней и нижней численных оценок b, c выражения a. Собственно нахождение этих оценок в нем выполняется синтезаторами "нижняяоценка" (первый антецедент) и верхняяоценка" (второй антецедент). Шаблон обращения к первому из них имеет вид $b \le a$, где a - входная переменная, b - выходная; шаблон обращения ко второму имеет вид $a \le c$, где a - входная переменная, c - выходная. Компилятор сам определяет, какой из синтезаторов для какого антецедента нужно использовать - идентифицированной на момент рассмотрения этих антецедентов является только переменная a, которая и рассматривается как входная. Указатели обработки первых двух антецедентов в данном приеме имеют вид "значения(1)", "значения(2)".

Идентифицирующий антецедент

Некоторые антецеденты теоремы приема могут вводить вспомогательный терм T и затем идентифицировать его с заданным термом B. Они называются идентифицирующими антецедентами. Самый распространенный вид идентифицирующего антецедента - "$A = B$". В этом случае терм T определяется, с учетом указателей нормализации, заданных в приеме, по терму A. К моменту обработки антецедента все переменные терма A должны быть уже идентифицированы. Расположение терма A в левой либо правой части равенства несущественно. Если терм B имеет хотя бы одну не идентифицированную переменную, то он будет идентифицироваться с термом T так же, как, например, заменяемая часть какого - либо тождества идентифицируется с подтермом текущего терма задачи. В простейшем случае, если B является переменной, то эта переменная будет идентифицирована с T. Таким образом в теоремах приемов часто вводятся вспомогательные обозначения для громоздких либо встречающихся несколько раз выражений. Если же все переменные терма B идентифицированы, то он, как и A, определяет некоторый терм T'. Тогда обработка антецедента сводится к сравнению T и T'.

Идентифицирующий антецедент может также иметь вид дизъюнкции нескольких равенств - $A_1 = B_1 \vee \ldots \vee A_n = B_n$. Здесь предпринимаются альтернативные попытки идентификации термов $B_1, \ldots, B_n$ с термами, определенными по $A_1, \ldots, A_n$. Как и выше, порядок размещения частей равенств несущественен. Наконец, возможно применение идентифицирующего антецедента "принадлежит(x номера(m k))", который последовательно идентифицирует переменную x с десятичными записями чисел $m, m+1, \ldots, k$. Такой антецедент применяется только в случаях, когда переменные

m, k идентифицируются с десятичными записями целых чисел. Для выделения идентифицирующих антецедентов используется указатель обработки "идентификатор($i_1 \ldots i_n$)", перечисляющий все их номера $i_1, \ldots, i_n$. В качестве примера рассмотрим теорему для нахождения корней квадратного уравнения:

$d = b^2 + 4ac \rightarrow ax^2 + bx = c \leftrightarrow \neg(a = 0)\ \&\ 0 \leq d\ \&\ (x = \frac{\sqrt{d}-b}{2a} \vee x = -\frac{\sqrt{d}+b}{2a}) \vee bx = c\ \&\ a = 0$.

Ее единственный антецедент вводит обозначение d для дискриминанта, которое далее используется в теореме несколько раз. В действительности, с учетом нормализаторов приема, в данном антецеденте происходит и вычисление дискриминанта - упрощение его и попытка разложения на множители. Указатель обработки антецедента здесь имеет вид "идентификатор(1)".

Если идентифицирующий антецедент $A = B$ применяет к сформированному по терму A утверждению T нормализаторы либо вспомогательные задачи на описание, то это утверждение может принять вид дизъюнкции нескольких подутверждений - "подслучаев". Чтобы рассматривать такие подслучаи по отдельности, вводится специальный режим идентификации терма B не со всей дизъюнкцией T, а с отдельными ее членами. Такой режим допустим для приема, выполняющего эквивалентную замену, причем итоговый заменяющий терм будет формироваться как дизъюнкция тех вариантов "теоремного" заменяющего терма, которые соответствуют отдельным подслучаям. Данный режим может быть применен сразу к нескольким перечисляющим подслучаи идентифицирующим антецедентам с номерами $i_1, \ldots, i_n$. Соответствующий указатель обработки антецедентов имеет вид "дизъюнктчлен($i_1 \ldots i_n$)". Он вводится в дополнение к указателю "идентификатор(...)", также выделяющему эти антецеденты. Примеры его использования можно найти в подразделе базы приемов, посвященном дифференциальным уравнениям. Идентифицирующий антецедент применяется для обращения к задаче на описание, решающей некоторое вспомогательное дифференциальное уравнение; по различным подслучаям решения этого вспомогательного уравнения определяются решения основного уравнения, и далее они объединяются в общую дизъюнкцию.

Иногда бывает необходимо формировать некоторый вспомогательный терм A, используемый приемом, в два этапа: сначала ввести новую переменную x и преобразовать с помощью нормализаторов либо вспомогательных задач терм $t(x)$ к требуемому виду $r(x)$ (например, разложить на множители), а затем для получения A подставить в $r(x)$ вместо x заданное выражение P. Чтобы обеспечить такой режим, вводятся дополнительный антецедент $x = P$ и указатель "свертка(x F)". Антецедент $x = P$ снабжается при этом стандартным указателем "идентификатор(i)". Переменная x будет идентифицироваться как новая переменная, причем нормализаторы, обрабатывающие термы с x, снабжаются дополнительными посылками - конъюнктивными членами утверждения F. По окончании применения таких нормализаторов в их результирующий терм вместо x будет подставляться P. Утверждение F может отсутствовать. Примером использования указателя "свертка" может служить теорема $h = bd\ \&\ i = bf\ \&\ g = \sqrt{b}e\ \&\ j = ag^3 + cg^2 + hg + i \rightarrow a\sqrt{b}e^3 + ce^2 + d\sqrt{b}e + f = j/b$, позволяющая выполнять разложение на множители выражения $a\sqrt{b}e^3 + ce^2 + d\sqrt{b}e + f$ сведением к разложению на множители многочлена $ag^3 + cg^2 + hg + i$. Новая переменная g этого многочлена введена идентифицирующим антецедентом $g = \sqrt{b}e$. Если бы вместо переменной здесь было сразу подставлено обозначаемое ею выражение, то при преобразованиях вид многочлена оказался бы утерян, и необходимые приемы не сработали бы. Прием использует указатель "свертка(g число(g))".

Программно реализуемый антецедент

Если переменные антецедента были идентифицированы с термами "константного" характера, обозначающими объекты, для представления которых предусмотрены нелогические структуры данных (например, десятичные числа), то проверка либо реализация такого антецедента могут выполняться процедурами, обрабатывающими не сами константные термы, а обозначаемые ими объекты, заданные нелогическими структурами данных. Такие антецеденты называем программно реализуемыми. Указатель, выделяющий программно реализуемые антецеденты, имеет вид "программа($i_1 \dots i_n$)", где $i_1, \dots, i_n$ - все их номера. В программно реализуемых антецедентах используются выражения и утверждения, для которых компилятор ГЕНОЛОГа может формировать процедуры непосредственного вычисления. К их числу, в частности, относятся выражения, построенные при помощи числовых констант, операций сложения, изменения знака, умножения, возведения в константную натуральную степень, модуля, взятия максимума либо минимума двух чисел, наибольшего общего делителя и наименьшего общего кратного.

Приведем наиболее часто встречающиеся типы программно реализуемых антецедентов:

1. Антецедент для деления целого числа m на целое ненулевое число n с остатком: $m = an + b$. Остаток здесь берется от 0 до $|n| - 1$.

2. Антецедент, перечисляющий все целочисленные делители m, k целого числа n: $n = mk$.

3. Антецедент, перечисляющий все представления целого числа n в виде произведения целого m на заданную натуральную степень p целого k: $n = mk^p$.

4. Антецедент для проверки числового равенства либо присвоения: $a = b$.

5. Антецедент для проверки неравенства $a < b$ либо $a \leq b$.

6. Отрицание, дизъюнкция либо конъюнкция нескольких программно реализуемых антецедентов проверочного типа также является программно реализуемым антецедентом.

7. Антецедент для нахождения целочисленных решений m квадратного уравнения с целыми коэффициентами a, b, c: $am^2 + bm = c$.

8. Антецедент для перечисления всех целых чисел i из промежутка от m до n: $i \in \{m, \dots, n\}$.

9. Антецедент для выделения наибольшей натуральной степени n целого числа a, на которую делится целое число b; определяется также частное c от деления b на эту натуральную степень: $b = a^n c$.

10. Антецедент для перечисления всех простых либо целочисленных делителей m целого n (если в фильтрах указано "простое(m)", то перечисляются простые делители, иначе - целочисленные): $m|n$.

11. Антецедент для проверки четности целого числа n: $n - even$.

В качестве примера рассмотрим теорему "x – целое & y – целое & $|a| \leq |b|$ & $b = ap + q \rightarrow ax + by = c \leftrightarrow \exists_z(x = z - py$ & z – целое & $az + qy = c)$", используемую для решения линейного целочисленного уравнения с двумя неизвестными. Она соответствует одному шагу алгоритма Евклида. В фильтрах приема указано, что a, b, c идентифицируются с целочисленными константами, и это позволяет реализовать третий и четвертый антецеденты с помощью операторов ЛОСа, применяемых к целым числам (для четвертого антецедента выполняется деление b на a с остатком). Указатели обработки антецедентов приема - "блокпроверок(1 2)" и "программа(3 4)".

Вместо указателя "программа(...)", программно реализуемый антецедент вида $x = t$ может быть выделен указателем "выч(...)". Это происходит в тех случаях, когда все переменные выражения t оказываются идентифицированы с константными термами, причем переменную x нужно идентифицировать с численной константой, полученной при приближенном вычислении значения t с помощью машинной арифметики для чисел формата "с плавающей запятой". Например, так делается в приемах анализатора "чертеж", позволяющего строить эскизы к планиметрическим задачам.

Список типов программно реализуемых антецедентов постоянно пополняется. Основные типы задаются приемами справочника "вычисл". Чтобы получить информацию о программной реализации антецедента, содержащего некоторый ключевой логический символ, находим в оглавлении приемов раздел данного символа, входим в его подраздел "Справочники", и далее - в подраздел "Вычисл". Анализируя приемы справочника "вычисл", определяем, какие типы программно реализуемых антецедентов существуют и на какие форматы данных они рассчитаны. Теорема справочника "вычисл" имеет вид "вычисл($s\ T\ A_1 \ldots A_n\ R$)", где s - ключевой логический символ; T - шаблон антецедента, содержащий символ s; $A_1, \ldots, A_n$ - указатели форматов входных и выходных переменных; R - оператор ЛОСа, вставляемый компилятором ГЕНОЛОГа в программу приема для обработки антецедента. Точно таким же образом справочник "вычисл" определяет компиляцию подвыражений программно реализуемых антецедентов. В этом случае некоторое A_i имеет вид "выход(f)"; f - формат значения выражения T.

Отдельный программно реализуемый антецедент приема сканирования задачи может представлять собой обращение к вычислительному пакету, реализованному на ГЕНОЛОГе и предназначенному для выполнения каких-либо трудоемких вычислений вне логического процесса. Таким образом обеспечивается подключение к логическому программированию "обычных" вычислений.

Проверка истинности антецедента при помощи решения задачи на доказательство

Для проверки истинности антецедента с помощью задачи на доказательство могут применяться различные указатели, уточняющие уровень используемых для нее средств. Эта проверка предпринимается лишь в исключительных случаях, так как она обычно гораздо более трудоемка, чем обращение к проверочному оператору. Решение вспомогательной задачи на доказательство может вызвать неоправданно большую паузу в решении основной задачи - если нет веских оснований считать, что проверяемое условие вообще выполнено. Поэтому рекомендуется сопровождать обращения к таким задачам ограничителями трудоемкости (см. указатель "Обрыв(...)").

1. Если максимальный уровень используемой задачи на доказателство равен 4, причем она снабжается комментарием "извлекается" и решается с заблокированным сканированием посылок (их веса подняты до величины, превосходящей максимальный уровень), то применяется указатель "легковидеть(i)"; i - номер антецедента (этот и другие приводимые ниже однотипные указатели U обработки обычно группируются в указатель $U(i_1 \dots i_m)$, ссылающийся сразу на несколько антецедентов).

2. Если максимальный уровень задачи на доказательство равен 5, то применяется указатель "усматривается(i)".

3. Наконец, наиболее сильное средство проверки - использование задачи на доказательство, максимальный уровень которой полагается совпадающим с максимальным уровнем текущей задачи. Здесь используется указатель обработки "следствие(i)". Простейший пример - прием, основанный на теореме $0 < a-b \to \neg(a = b)$ и используемый для попытки усмотреть неразрешимость уравнения $a = b$ в условии задачи на описание путем доказательства неравенства. Этот прием применяется на достаточно большом уровне, когда решение задачи обычными средствами уже зашло в тупик. Первый антецедент в нем выделен указателем "следствие(1)", причем трудоемкость попытки доказательства неравенства (может быть, придется еще проверить и противоположное неравенство) ограничена сверху с помощью указателя "лимит(3000000)".

4. Особо выделен случай, когда решается задача на доказательство, причем она фактически сводится к проверке истинности заданного антецедента. Тогда вспомогательная задача формируется при помощи оператора "спуск" (специально предназначенного для формирования задач на описание либо доказательство, к которым решаемая задача может быть сведена путем замены некоторого условия). Максимальный уровень ее полагается равным максимальному уровню текущей задачи на доказательство. Данный режим проверки вводится при помощи указателя "доказать(i)". Этот же указатель можно использовать в пакетных операторах, где он имеет несколько иной смысл - просто указывает на обращение к вспомогательной задаче на доказательство с максимальным уровнем обращения, равным 8. В качестве примера рассмотрим теорему

$$\forall_x(g(x) \to 0 < f(x)) \to 0 < \prod_{x,g(x)} f(x),$$

на которой основан прием, предпринимающий попытку доказательства положительности конечного произведения путем установления положительности его общего члена. Первый антецедент, выделенный указателем "доказать(1)", обрабатывается путем решения задачи на доказательство, полученной из текущей задачи на доказательство путем соответствующей замены условия. Заметим, что сама текущая задача при этом не изменяется, пока не будет решена вспомогательная - лишь после этого условие основной задачи будет заменено на константу "истина".

Антецеденты, реализуемые путем обращения к вспомогательной задаче на описание

Применение приема вывода следствий в задаче на исследование может быть связано с необходимостью предварительно вычислить значения группы вспомогательных

выражений. Для такого вычисления обычно используется задача на описание, имеющая цель "известно ...", перечисляющую те известные параметры, через которые должны быть определены вычисляемые выражения. Чтобы создать данную задачу на описание и обратиться к ее решению, в теореме приема вводятся антецеденты $x_1 = t_1, \ldots, x_n = t_n$, перечисляющие подлежащие вычислению выражения $t_1, \ldots, t_n$ и те переменные $x_1, \ldots, x_n$, которым присваиваются вычисленные значения (т.е. выражения, не содержащие неизвестных контекста применения приема). Эти антецеденты сопровождаются указателем "вычисл(A B)". Здесь A - список их номеров; B - список дополнений: "посылки(P)" - указание конъюнкции P дополнительных посылок, используемых в задаче; "удалениепосылок(Q)" - Q есть список названий таких разделов, что содержащие их понятия посылки текущей задачи не переносятся во вспомогательную задачу на описание. Известные параметры вспомогательной задачи берутся из списка известных параметров текущей задачи на исследование.

Антецеденты, реализуемые идентифицирующими операторами

В некоторых предметных областях приходится идентифицировать антецеденты, для которых контекст идентификации не содержит явных прототипов, однако такие прототипы могут быть извлечены из утверждений данного контекста с помощью крайне небольшого числа простых логических переходов. Высокая частота попыток такой идентификации и расположение их в самом начале программ приемов делают нежелательным применение здесь обычных проверочных операторов и синтезаторов. Для ускорения идентификации в таких ситуациях применяются так называемые идентифицирующие операторы - специальные реализованные непосредственно на ЛОСе вспомогательные процедуры, выполняющие указанные выше логические переходы.

Например, для перечисления всех точек, выделенных на текущий момент на некоторой прямой, применяется идентифицирующий оператор "точкипрямой", который усматривает точки из различных источников: точки из названия самой прямой; точки из явно содержащихся в посылках условий принадлежнотси этой прямой; точки из условий принадлежности интервалам и отрезкам, концы которых принадлежат данной прямой.

Ускоренное, по сравнению с пакетными операторами, функционирование идентифицирующих операторов достигается не только за счет малого числа и простоты применямых в них логических переходов, но и за счет использования специальных адресных структур данных, группирующих посылки специального вида, необходимые для выполнения таких переходов. Эти структуры данных технически оформлены в виде комментариев к посылкам "списокпосылок ...", "выражение ..."и "отр ..." (подробнее о них см. информацию о логических символах - заголовках этих комментариев). Они автоматически корректируются в процессе преобразований списка посылок - эту работу выполняют процедуры, осуществляющие преобразования списка посылок, т.е. "вывод", "заменавхождения", и т.п.

Применение специальных адресных структур для быстрого поиска нужных посылок сближает по своей эффективности применяемое логическое представление с сетевым, где непосредственно указывались бы объекты, находящиеся в заданном отношении с рассматриваемым объектом. Такое сближение усиливается использованием буфера, в котором сохраняются результаты нескольких десятков последних обращений к идентифицирующему оператору (см. комментарий к посылкам "Буфер"). Фактически этот буфер становится в некотором смысле самоорганизующимся "расширенным

контекстом", содержащим сотни готовых ответов на типовые вопросы, относящиеся к текущей ситуации. Во избежание ошибок управляющего характера, буферы идентифицирующих операторов сбрасываются после каждого преобразования посылок, которое могло бы изменить результат обращения.

Указатель обработки антецедентов с номерами $i_1, \ldots, i_n$ посредством идентифицирующих операторов имеет вид "усм($i_1 \ldots i_n$)".

Ввод нового идентифицирующего оператора осуществляется следующим образом:

1. Определяется, какого вида группа антецедентов $T_1, \ldots, T_n$ будет обрабатываться при помощи нового оператора. В этой группе антецедентов выделяются входные термы $A_1, \ldots, A_m$ (эти термы должны быть к началу идентификации уже определены - либо как выходные значения ранее примененных идентифицирующих операторов, либо через идентифицированные значения их переменных). Выделяются также выходные термы $B_1, \ldots, B_k$, которые будут определяться идентифицирующим оператором. Здесь допустим случай $k = 0$.

2. На ЛОСе создается программа идентифицирующего оператора $f(x_1 \ldots x_{m+k+1})$, входные данные которой суть: x_1 - текущая задача на исследование либо на доказательство, при сканировании которой применяется идентифицирующий оператор; $x_2, \ldots x_{m+1}$ - термы $A_1, \ldots, A_m$. Выходные переменные $x_{m+2}, \ldots,$ x_{m+k+1} перечисляют наборы термов $B_1, \ldots, B_k$ (в случае $k = 0$ оператор выполняет проверку истинности соответствующих $T_1, \ldots, T_n$).

3. Создаются приемы справочников "усм" и "См", основанные на псевдотеореме вида "усм(f вход($A_1 \ldots A_m$) выход($B_1 \ldots B_k$) усм(T_1 & $\ldots$ & T_n) $C_1 \ldots C_p$)". Здесь $C_1, \ldots, C_p$ - информационные элементы, используемые компилятором ГЕНОЛОГа при формировании обращений к оператору f. Вид этих элементов приведен в информации о логическом символе "усм". Отметим, в частности, элемент "уровень(U)", который определяет уровень приоритетности при применении компилятором данного идентифицирующего оператора (чем меньше U, тем выше приоритет).

После этого компилятор ГЕНОЛОГа готов к использованию нового идентифицирующего оператора.

При вводе комплекта новых идентифицирующих операторов следует учитывать, что идентификация одних и тех же утверждений в разных приемах может происходить в различном порядке - с различными комбинациями определенных к моменту идентификации подтермов. Отсутствие идентифицирующего оператора, обслуживающего какую-либо из таких комбинаций, может ухудшить качество компиляции или вообще вызвать отказ компилятора.

Основной раздел, в котором были использованы идентифицирующие операторы - элементарная геометрия. Перечислим логические символы, для которых в их подразделах оглавления базы приемов указаны справочники "усм" и "См", а также созданные для этих символов идентифицирующие операторы:

1. "Прямая". Введены операторы: "прямыеточки" (перечисление прямых, проходящих через данную точку); "точкипрямой" (перечисление точек прямой); "точкапрямой" (проверка принадлежности точки прямой); "общаяточка" (определение общих точек двух прямых); "общаяпрямая" (определение прямых,

проходящих через две точки); "лежатнапрямой" (определение прямых, проходящих через три заданные точки - так как нет гарантий, что эти точки действительно различны или что не совпадают две по-разному обозначенные прямые, он выдает результаты в режиме перечисления); "точкилуча" (перечисление точек луча, определяемого по двум точкам); "точкалуча" (проверка принадлежности третьей точки лучу, определяемому двумя первыми точками); "пооднусторону"; "поразныестороны" (перечисление точек, лежащих на плоскости, соответственно, по одну либо по разные стороны от данной точки относительно данной прямой); "Однасторона"; "Разныестороны" (проверка расположения двух точек по одну либо по разные стороны относительно данной прямой). Заметим, что последние два идентифицирующих оператора в итоге оказались практически неиспользуемыми, так как их вытеснили хотя и более медленные, но на порядок более мощные проверочные операторы "разныестороны" и "однасторона".

2. "Плоскость". Введены операторы: "точкиплоскости" (перечисление точек плоскости); "точкаплоскости" (проверка принадлежности точки плоскости); "плоскоститочки" (перечисление плоскостей, проходящих через заданную точку); "общаяплоскость" (проверка принадлежности двух прямых одной плоскости).

3. "Отрезок". Введены операторы: "точкиотрезка" (перечисление точек отрезка); "точкаотрезка" (проверка принадлежности точки отрезку); "конецотрезка" (определение противоположного конца отрезка, если заданы точка отрезка и первый конец). Заметим, что идентифицирующие операторы для работы с отрезком сами по себе весьма слабые; чтобы эффективно определять взаимное расположение точек на прямой с помощью этих операторов, необходимо заблаговременно в явном виде выводить следствия - утверждения о принадлежности или непринадлежности точек отрезкам и интервалам.

4. "Расстояние". Введены операторы "расстояния" (перечисление точек, для которых введено в рассмотрение расстояние до данной точки); "смрасстояние" (проверка того, что расстояние между двумя точками введено в рассмотрение); "равныедлины" (перечисление пар точек C, D, таких, что расстояние от C до заданной точки A равно расстоянию от D до заданной точки B); "равноудалена" (перечисление пар точек B, C и прямых BC, таких, что расстояние от B до заданной точки A равно расстоянию от B до C, причем точка A лежит на прямой BC.

5. "Угол". Введены операторы: "углывершины" (перечисление пар точек B, C, для которых введен в рассмотрение угол с заданной вершиной A); "усмугол" (проверка того, что заданный угол ABC был введен в рассмотрение); "имяугла" (по заданной вершине угла A, двум прямым, проходящим через A и двум точкам, определяющим направления лучей на этих прямых, перечисляются пары точек B, C, определяющих заданный этими лучами угол BAC).

6. "Параллельны". Введены операторы: "параллелпрямые" (перечисление прямых, параллельных данной); "усмпараллельны" (проверка параллельности двух прямых); "параллели" (перечисление пар параллельных прямых, первая их которых проходит через точку A, а вторая - через точку B); "параллелплоскости" (перечисление плоскостей, параллельных данной).

7. "Перпендикулярно". Введены операторы: "перпендикуляры" (перечисление прямых, перпендикулярных данной); "перпендикулярны" (проверка перпендикулярности двух прямых); "перпендикпрямые" (перечисление пар прямых, параллельных двум заданным перпендикулярным прямым. В этом и ряде других случаев входным данным для идентифицирующего оператора служит не терм, а вхождение в терм условия перпендикулярности некоторых прямых - см. подробнее информационный элемент "перпендикулярно" из описания формата идентифицирующих операторов); "прямыеуглы" (перечисление пар прямых, параллельных двум заданным перпендикулярным прямым, и точек их пересечения); "Перпендикулярны" (проверка перпендикулярности прямой и плоскости); "Перпендикуляры" (перечисление прямых, перпендикулярных плоскости); "Перпендикпрямые" (перечисление пар "прямая - плоскость", параллельных заданной паре перпендикулярных прямой и плоскости); "Прямыеуглы" (перечисление пар "прямая - плоскость", параллельных заданной паре перпендикулярных прямой и плоскости, а также точек их пересечения).

8. "Окружность". Введены операторы: "точкиокружности" (перечисление точек окружности); "окружноститочки" (перечисление окружностей, проходящих через заданную точку); "точкаокружности" (проверка принадлежности точки заданной окружности).

В качестве примера использования идентифицирующих операторов рассмотрим теорему $l(AB) = l(BC)$ & прямая$(BD) \perp$ прямая(AC) & $D \in$ прямая$(AC) \to l(AD) = l(DC)$. На этой теореме основан прием, усматривающий медиану в высоте BD равнобедренного треугольника ABC. Применение приема начинается с обнаружения равенства $l(AB) = l(BC)$ либо пары равенств $l(AB) = a$,$l(BC) = a$, позволяющих идентифицировать точки A, B, C. Прием имеет указатель "усм(2 3)", так что далее используется идентифицирующий оператор, перечисляющий для идентификации прямой BD прямые, перпендикулярные прямой AC. Идентифицирующий оператор проверки принадлежности точки прямой отбирает ту из них, на которой лежит точка B. Наконец, применяется идентифицирующий оператор, находящий точку D пересечения прямых BD и AC.

Уточнение списка посылок, участвующих в проверке либо реализации антецедента

1. При обработке пакетным оператором либо вспомогательной задачей антецедента с номером i могут быть введены дополнительные посылки $A_1, \ldots, A_n$. Для этого используется указатель "занесениепосылки(i и$(A_1 \ldots A_n)$)". Наиболее часто дополнительные посылки вводятся для исключения кванторного антецедента теоремы. Например, на теореме

   $\forall_y(f(y) \to \neg(h(g(y)))) \to$ непересек$(\mathrm{set}_x(\exists_y(f(y)\ \&\ x = g(y))), \mathrm{set}_z(h(z)))$

   можно было бы создать прием проверки непересечения двух множеств, заданных описателями "класс". Однако, эта теорема имеет кванторный антецедент, и вместо нее можно взять теорему

   $\neg(h(g(y))) \to$ непересек$(\mathrm{set}_x(\exists_y(f(y)\ \&\ x = g(y))), \mathrm{set}_z(h(z))),$

   полученную сохранением лишь консеквента кванторного антецедента, однако добавить указатель "занесениепосылки(1 $f(y)$)". Результат будет тот же, что и

в первом случае, но запись теоремы сокращается, и экономится срабатывание приема доказательства истинности кванторной импликации, перебрасывающего ее антецеденты в посылки задачи.

2. Если при обработке пакетным оператором либо вспомогательной задачей антецедента с номером i требуется исключить посылки, удовлетворябщие заданному условию, то применяется указатель "удалениепосылок(i X $P(X)$)". Здесь X - вспомогательная переменная, обозначающая исключаемую посылку; $P(X)$ - условие на эту посылку. Обычно удаление посылок предпринимается в тех случаях, когда есть основания предполагать наличие в контексте громоздких посылок, которые заведомо не нужно использовать при проверке утверждения рассматриваемого типа.

3. Если в приеме нормализатора обработка антецедента проверочным оператором должна производиться с использованием списка посылок этого нормализатора, пополненного утверждениями из контекста текущего вхождения, то используется указатель "облнорм".

Специальная идентификация антецедентов, имеющих своим заголовком транзитивное бинарное отношение

Для посылок, заголовком которых служит транзитивное бинарное отношение, обычно предусматривается стандартизация, сжимающая все многообразие извлекаемых по транзитивности следствий к возможно меньшему числу базисных утверждений. В этой ситуации для идентификации антецедента с одним из элементов "замыкания по транзитивности" этих базисных утверждений служат специальные указатели.

Если рассматриваемое транзитивное отношение является отношением эквивалентности E, то обычно антецедент $E(X\ Y)$ идентифицируется либо с явно содержащимся в контексте идентификации утверждением $E(A\ B)$, либо с собираемым из двух таких утверждений $E(A_1\ A_2)$, $E(A_3\ A_2)$ новым утверждением $E(A_1\ A_3)$. Последнее объясняется тем, что для всех находящихся в отношении E рассматриваемых в задаче объектов выбирается некоторый эталонный объект (в данном случае его роль играет A_2), и связь с ним других объектов задается с помощью утверждений вида $E(A\ A_2)$. Указателем на данную идентификацию служит "равно(i)", где i - номер антецедента $E(X\ Y)$. Примером ее может служить обработка антецедента $l(AB) = l(BC)$ в приводившемся выше примере с высотой равнобедренного треугольника.

Для транзитивного отношения E, не являющегося отношением эквивалентности, при идентификации может быть использован оператор "транзитпереход", перечисляющий с учетом транзитивности все термы, находящиеся в отношении E с заданным термом на основе явно присутствующих в контексте идентификации утверждений $E(A\ B)$. Идентификация антецедента $E(X\ Y)$ начинается здесь после того, как были идентифицированы все переменные одного из операндов X, Y. Данный режим вводится указателем "транзитпереход($i_1 \ldots i_n$)", где $i_1, \ldots, i_n$ - номера всех реализуемых таким образом антецедентов. Например, в основанном на теореме
$b \subseteq$ прообраз$(f, a) \rightarrow$ образ$(f, b) \subseteq a$
приеме усмотрения включения прослеживаются по цепочкам включения множества b, явно указываемые в посылках, и таким образом идентифицируется правая часть включения $b \subseteq$ прообраз(f, a).

Очередность обработки антецедентов при применении приема

Для ускорения работы программы приема иногда приходится определенным образом упорядочивать обработку антецедентов: отодвигать в конец программы априори трудоемкие вычисления, и наоборот, переносить в начало быстрые проверки, позволяющие усиливать отсечение нереализуемых ситуаций.

Чтобы приблизить к началу программы обработку антецедента с номером i, используется указатель "начало($i\ x_1 \dots x_n$)". В этом случае обработка данного антецедента будет выполняться сразу же, как только идентифицируются значения переменных списка $x_1, \dots, x_n$.

Указатель "конец($i_1 \dots i_n$)" отодвигает обработку антецедентов с номерами $i_1, \dots, i_n$ к концу программы приема.

В тех случаях, когда антецедент с номером i непосредственно идентифицируется с некоторым утверждением из контекста идентификации, можно заблокировать его обработку до того момента, пока не окажутся идентифицированы заданные переменные $x_1, \dots, x_n$. Для этого используется указатель "подчинено($i\ x_1 \dots x_n$)".

Слежение за появлением требуемого антецедента

Если проверка истинности антецедента при помощи пакетного оператора оказалась неудачной, то можно организовать слежение за появлением этого антецедента "в чистом виде" в контексте идентификации, с переключением внимания, обеспечивающим в этом случае повторную попытку применения приема. Здесь используется указатель обработки "См(i)", где i - номер антецедента.

Ограничение трудоемкости обработки антецедента

Для установки лимита в N шагов работы интерпретатора ЛОСа, отводимых на обработку антецедента с номером i, применяется указатель "Обрыв($i\ N$)". Если эта обработка завершается менее чем за N шагов, то данный лимит сбрасывается, иначе - происходит откат, такой же, как при нелимитированной неудачной попытке обработки.

Отмена ограничений на использование утверждений текущего контекста для непосредственной идентификации

Если некоторые антецеденты теоремы приема были непосредственно идентифицированы с утверждениями $A_1, \dots, A_n$ из контекста идентификации, и после этого произошло обращение к проверочному оператору либо пакетному синтезатору, то утверждения $A_1, \dots, A_n$ не будут при работе данного оператора использоваться в непосредственной идентификации антецедентов. Они регистрируются для этого в комментарии "исключение ...", передаваемом оператору. Это делается по умолчанию для предотвращения зацикливаний в рекурсивных обращениях пакетных операторов. Однако, такое ограничение можно отменить для группы непосредственно идентифицируемых антецедентов приема пакетного оператора, имеющих номера $i_1, \dots, i_m$. Если вводится указатель "повтор($i_1 \dots i_m$)", то данные антецеденты будут идентифицироваться с утверждениями из текущего контекста без учета комментария "исключение ...".

Выделение антецедентов, используемых для замещения условия при обратном выводе

Ранее (см. раздел "Типы заголовков приемов") были введены приемы с заголовком "подборзначений". Они осуществляют попытку подбора примера путем замены идентифицированного с консеквентом A_0 теоремы приема $\forall_{x_1 \ldots x_m}(A_1 \ \& \ \ldots \ \& \ A_k \rightarrow A_0)$ условия задачи на описание на группу антецедентов с номерами $i_1, \ldots, i_n$, выделенных указателем "подборзначений($i_1 \ldots i_n$)". Остальные антецеденты при этом обрабатываются обычным образом. В качестве примера рассмотрим теорему $0 < a \ \& \ x = a/2 \rightarrow x \in (0, a)$, используемую для указания точки, принадлежащей интервалу $(0, a)$. Она позволяет переходить от условия $x \in (0, a)$ к условию $x = a/2$, после того, как проверено, что a положительно. Прием имеет фильтр, проверяющий, что неизвестная x не входит в другие условия (кроме, быть может, условия "число(x)"). Первый его антецедент снабжен указателем "блокпроверок(1)", второй - "подборзначений(2)".

Принятие истинности антецедента по умолчанию в зависимости от контекста

Ряд антецедентов в определенных контекстах, уточняемых фильтрами приема, могут оказаться избыточными (но не во всех случаях применения приема - иначе их можно было бы вообще отбросить). Чтоы в этих ситуациях сэкономить время на проверках или избежать отказа из-за возможной некомплектности списка посылок, имеющихся в определяемой фильтром ситуации, можно применять указатель "пассив(i A)", где i - номер антецедента, проверка которого отключается, A - фильтр, определяющий условие отключения проверки.

Унифицирующий антецедент

В приемах вывода теорем используются антецеденты вида $A = B$, обрабатываемые путем обращения к процедуре унификации. Процедура обрабатывает сразу все такие равенства, унифицируя их левые части с правыми. Равенства содержат выражения $f(t_1 \ldots t_k)$, соответствующие уже идентифицированным функциональным переменным $f(x_1 \ldots x_k)$. Здесь каждый содержащий еще не идентифицированные переменные терм t_i представляет собой новую переменную, и значения всех таких новых переменных определяются процедурой унификации. Допускается использование для одной и той же переменной x_j нескольких различных t_j. Перед обращением к процедуре унификации в таких случаях выполняется соответствующая расклейка переменной x_j на несколько различных новых переменных, вместо которых и будут подставляться при унификации соответствующие различные t_j. Унифицирующие антецеденты выделяются указателем "унификация($i_1 \ldots i_n$)", где $i_1, \ldots, i_n$ - номера этих антецедентов.

11.4.3 Указатели, уточняющие тип основного преобразования

1. Если прием замены, применяемый при сканировании задачи, должен заменять не только текущее вхождение преобразуемого терма, но все его вхождения в термы задачи, для которых удается усмотреть корректность данной замены, то используется указатель "заменавхождений".

2. Если замена выполняется в задаче на доказательство либо на описание, причем нет полной уверенности в том, что она не заведет задачу в тупик, то можно организовать ввод вспомогательной задачи - копии текущей задачи, и выполнить данную замену лишь во вспомогательной задаче. Тогда, после получения отказа на такую вспомогательную задачу, решение текущей задачи будет продолжаться без выполнения рассматриваемой замены; если же на вспомогательную задачу будет получен ответ, то он будет выдан как ответ текущей задачи. Данный режим обеспечивается указателем "попытказамены".

3. Если при выполнении приема "замещениеусловий", передающего во внешнюю задачу на описание найденные в ее блоке анализа следствия, требуется сразу же вернуться к сканированию этой внешней задачи, то применяется указатель "обрывзадачи(A)". Здесь A - фильтр, определяющий условия, при которых необходим такой переход. Он может отсутствовать, и тогда указатель имеет вид "обрывзадачи".

4. Если выполняется замена согласно тождеству "вариант($P\ t_1\ t_2) = t$" (например, при усмотрении модуля), то имеется возможность автоматически обобщить это тождество до "вариант($P\ F(t_1)\ F(t2)) = F(t)$", введя указатель "извлечениеварианта". Прием будет пытаться представить находящиеся "под вариантом" термы в виде $F(t_1)$ и $F(t_2)$, используя процедуру "извлечениеварианта". Заметим, что все входящие в t_1, t_2 переменные должны быть идентифицированы из других частей теоремы (например, из условия P).

5. Если теорема приема проверочного оператора имеет вид $\forall_{y_1 \ldots y_n}(P(\ldots a \ldots)\ \&\ P(\ldots b \ldots) \rightarrow P(\ldots h(a\ b) \ldots))$, где h - ассоциативно - коммутативная операция, то есть прием предпринимает разбиение группы операндов операции h на два подмножества, то возможно предпринять разбиение этой группы операндов сразу на одноэлементные подмножества и проверять $P(\ldots a \ldots)$ последовательно для всех одиночных операндов a. Такой режим вводится указателем "дистрибразвертка(K)", где K - указатель вхождения в теорему операции $h(a\ b)$.

6. Если прием тождественной либо эквивалентной замены, либо прием нормализатора основан на тождестве (либо эквивалентности) типа $P(h(a\ b)) = f(P(a)P(b))$, с ассоциативно - коммутативными f, g, то возможно ввести режим одновременной обработки всех операндов операции h либо f, в зависимости от направления замены. В случае "разгруппировки" операции h применяется указатель "набор(первыйтерм)", иначе (при группировке операндов операции f) - указатель "набор(второйтерм)".

7. Обычно при компиляции тождества, заголовок заменяемой части которого есть ассоциативно - коммутативный символ f, вводится новая переменная x, добавляемая к f - членам обеих частей тождества. Эта переменная идентифицируется с остаточными членами операции f. Если же требуется идентифицировать заменяемую часть целиком со всем списком f - членов преобразуемого терма, не допуская остатка последних, то вводится указатель "модификатор" (в компиляторе модификатором называется упомянутая выше переменная x).

8. В некоторых предметных областях (например, в геометрии) возникает необходимость в определенных преобразованиях теоремы перед компиляцией приема. Такие преобразования обобщают определяемое теоремой описание ситуации,

уменьшая потребность в средствах специальной идентификации. В качестве примера здесь можно было бы привести прием, основанный на теореме Пифагора. В этом приеме рассматривается прямоугольный треугольник ABC с прямым углом B. Последнее формулируется как условие перпендикулярности прямых AB и BC. Однако, применяемая стандартизация условий параллельности и перпендикулярности прямых выбирает для каждого класса параллельных прямых лишь один эталонный экземпляр, и все условия параллельности и перпендикулярности для этого класса формулируются с его участием. В результате может оказаться так, что в задаче не найдется условия перпендикулярности для требуемых прямых, а будет условие перпендикулярности для каких-то двух других прямых, DE и FG. Соответственно, в теореме вводятся новые переменные D, E, F, G, а условие перпендикулярности для AB,BC преобразуется в условие перпендикулярности для DE,FG и условия параллельности AB с DE и BC с FG. Такого рода преобразования теоремы выполняются до тех пор, пока не будут устранены все нежелательные связи между объектами, ограничивающие общность описания ситуации. Преобразования эти выполняются по умолчанию (их обеспечивает процедура "развязка", к которой обращается компилятор). Однако, их можно отключить (иногда это необходимо), вводя указатель "развязка".

9. Если теорема приема имеет вид $A \to P(t_1 \dots t_n)$, где отношение P однозначно определяет значение t_i по значениям $t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n$, то возможно автомати ческое преобразование этой теоремы при компиляции к виду

$$A \to (P(t_1 \dots t_{i-1}\ x\ t_{i+1} \dots t_n) \leftrightarrow x = t_i).$$

Это делается при наличии указателя "однозначно", причем факт однозначного определения t_i распознается компилятором с помощью справочника "однозначно". В качестве примера приведем теорему "натуральное(n) $\to$ наибольший(n, set_m(натуральное(m) & $m|n$))", по которой создается прием, заменяющий утверждения вида "наибольший(A, set_m(натуральное(m) & $m|n$))" на "$A = n$".

10. Если имеется множество альтернативных приемов, которые должны срабатывать на одном и том же уровне, но среди них нужно отобрать один самый "лучший", то применяется специальный механизм, связанный с использованием указателя "оценка($u_1 A_1 \dots A_n v$)". Такая ситуация складывается, например, при выборе той формулы для площади треугольника, которая будет наиболее полезна в решаемой планиметрической задаче. Если промедлить со вводом формулы, то решатель отвлечется на посторонние действия. Поэтому распределить формулы для площади по различным уровням сканирования оказывается невозможным - они должны относиться к одному и тому же уровню. Однако, можно использовать приемы, выписывающие эти формулы, в специальном "двухуровневом режиме". Каждый из них имеет два уровня срабатывания: предварительный уровень u_1 и основной уровень u_2, больший предварительного. На предварительном уровне предпринимается лишь выбор оценки v предпочтительности применения приема (чем меньше оценка, тем выше предпочтительность). Эта оценка сохраняется в комментарии ($A_1 \dots A_n v$) к посылкам задачи. Здесь компоненты $A_1, \dots A_n$ - общие для всех альтернативных применений приемов (например, представляющие собой ссылку на треугольник, для которого нужно выписать формулу площади), а в компоненте v отбирается минимальное значение оценок, сообщаемых приемами при сканировании на

уровне u_1. Таким образом, при сканировании на уровне u_2 уже можно отобрать наилучший прием: им окажется тот, чья повторно вычисляемая оценка v совпадет с оценкой w, извлекаемой из комментария $(A_1 \dots A_n w)$. Если таких приемов будет несколько, то сработает первый из них. Чтобы заблокировать срабатывание прочих приемов с той же оценкой, сработавший прием может ввести специальный комментарий. Для создания описанного двухуровневого режима срабатывания приема достаточно ввести приведенный выше указатель "оценка(...)". Заметим, что перед A_1 может быть размещен терм "условие(U)", где U - фильтр, определяющий случай для выбора оценки. Чтобы находить оценку для нескольких различных непересекающихся подслучаев, вводятся несколько соответствующих им условных указателей "оценка(...)".

11.4.4 Указатели, играющие роль дополнительных фильтров либо отменяющие ограничения на применение приема

Если прием использует нормализатор A, выполняющий преобразование терма t, причем применение приема теряет смысл при сохранении терма t неизменным, то вводится указатель "различны(A t)", блокирующий в этих случаях срабатывания приема.

Обычно применение приема блокируется, если он пытается преобразовать утверждение, используемое для сопровождения по области допустимых значений других термов задачи. Однако, указатель "сопровождение" позволяет отменить это ограничение. Применять его следует лишь в исключительных случаях, когда либо продолжение сопровождения по о.д.з. становится ненужным, либо усмотрение условий на о.д.з. легко может быть осуществлено и на основе преобразованного утверждения - например, если предпринимаемое преобразование должно синхронным образом изменить и сопровождаемые данным утверждением термы задачи.

11.4.5 Указатели, уточняющие способ формирования новых термов

1. Если квантор существования в заменяющем терме приема замены либо приема нормализатора берется по конечной области, то его бывает целесообразно представить в виде конечной дизъюнкции. В этом случае используется указатель "или(A $B_1 \dots B_n$)", где A - указатель вхождения в теорему приема данного квантора существования; $B_1, \dots, B_n$ - список указателей вхождений конъюнктивных членов подкванторного утверждения, определяющих перечисление дизъюнктивных членов и компилируемых так, как компилируются программно реализуемые антецеденты. В качестве примера рассмотрим теорему "$x < n$ & натуральное$(x) \to \exists_m(m \in \{1, \dots, n-1\}$ & $x = m)$", используемую для вывода условия $x = 1 \vee \dots \vee x = n - 1$ задачи на описание, имеющей неизвестное выражение x, принимающее натуральные значения и ограниченное сверху натуральной константой n. Такое условие инициирует далее разбор случаев. Прием имеет указатель "или(фикс(0)фикс(0 2 1))", у которого первый указатель вхождения в теорему относится к вхождению квантора существования, а второй - к вхождению утверждения $m \in \{1, \dots, n-1\}$, определяющего перечисление натуральных значений m.

2. Предположим, что формируется терм вида $f(t)$ для ранее идентифицированной

с некоторым термом $T(x)$ функциональной переменной $f(x)$, причем можно считать достаточно вероятным появление внутри $T(x)$ подтермов $R_1(x), \ldots,$ $R_n(x)$, для которых $R_1(t), \ldots, R_n(t)$ могут быть упрощены до термов $Q_1, \ldots, Q_n$. Тогда можно еще до подстановки t вместо x в $T(x)$ заменить все вхождения $R_1(x), \ldots, R_n(x)$ в $T(x)$ на $Q_1, \ldots, Q_n$ (выделяя в первую очередь все вхождения $R_1(x)$, затем - не пересекающиеся с ними вхождения $R_2(x)$, и т.д.), и лишь после этого оставшиеся (не накрытые $R_1(x), \ldots, R_n(x)$) вхождения x заменить на t. Такой режим формирования $f(t)$ обеспечивается указателем "мультизамена(K $R_1(x)$ $Q_1 \ldots R_n(x)$ Q_n)", где K - указатель вхождения в теорему приема терма $f(t)$. В решателе он используется для ускоренного упрощения выражений, возникающих после замены переменной при интегрировании (подстановки Эйлера и др.).

3. Как правило, вводимые в приеме новые термы t обрабатываются цепочкой нормализаторов, а иногда - также и вспомогательными задачами на преобразование либо описание. В тексте приема не сохраняются координаты того вхождения в теорему терма t, который должен быть подвергнут обработке - считается, что она распространяется на все его вхождения, как в теорему, так и в фильтры приема. Однако, имеется возможность часть вхождений t в теорему и в фильтры формировать с заблокированной обработкой нормализаторами (это бывает полезно, если t встречался в идентифицируемой части теоремы и был отождествлен с некоторым термом задачи T - для ссылок на этот "исходный" вид терма T). Здесь применяется указатель "копия(K)", где K - указатель того вхождения t в теорему, для которого обработка нормализаторами блокируется. Если требуется заблокировать обработку нормализаторами вхождения t в фильтр приема, то в этом фильтре вместо t помещается запись "копия(t)". Наиболее частый случай использования указателя "копия" - блокировка обработки нормализаторами некоторого терма при проверке предварительных условий. Например, в приеме предпринимается попытка разложения на множители некоторой суммы $a + b$, но лишь после того, как установлено, что она неотрицательна. При проверке неотрицательности нет необходимости использовать результат разложения на множители $a+b$, а берется исходный вид этой суммы.

4. Если переменная X была идентифицирована с выражением, заголовком которого служит ассоциативно - коммутативная операция f, то при формировании нового терма вида $f(A \ldots X \ldots B)$ корневые f - операнды X будут объединены в общую группу f - операндов с $A, \ldots, B$. В тех случаях, когда это нежелательно (например, нужно выделить явное вхождение подтерма X, чтобы впоследствии заменить все такие вхождения на новую переменную), применяется указатель "спускоперандов(X)" - он блокирует включение f - операндов идентифицированного с X выражения в группу f - операндов внешней операции.

11.4.6 Указатели, определяющие дополнительные преобразования

Вывод дополнительных посылок и условий

В некоторых случаях целесообразно дополнить выполнение основного преобразования приема (тождественную либо эквивалентную замену, вывод условия либо посылки) занесением в список посылок сопровождающих утверждений, истинность

которых является следствием антецедентов теоремы, а также, быть может, ряда других специально проверяемых утверждений. Для этого используется указатель "посылка(условие(A)$B_1 \dots B_n$ C примечание(K_1) ... примечание(K_m))". Здесь A - фильтр, определяющий условия, при которых занесение посылки целесообразно (он может отсутствовать); $B_1, \dots, B_n$ - сформулированные в терминах переменных теоремы утверждения, истинность которых должна быть дополнительно установлена с помощью проверочных операторов перед занесением новой посылки; C - сама новая посылка (тоже в терминах теоремных переменных); $K_1, \dots, K_m$ - комментарии, которыми сопровождается эта посылка. При выводе C проверяется, что все утверждения, использованные для проверки корректности преобразований приема и для проверки $B_1, \dots, B_n$, относятся к списку посылок текущей задачи. Заметим, что фактически терм "примечание(K_i)" здесь имеет вид "примечание($D_1 \dots D_p$)", так что K_i есть набор операндов ($D_1 \dots D_p$), начинающийся с логического символа D_1 (заголовка комментария); $p \geq 0$. Как и обычно, по умолчанию D_j при $0 < j$ компилируются в виде термов; если требуется вместо этого иметь в комментарии переменную либо логический символ, то используется запись "переменная(D_i)". Типичный случай использования указателя "посылка(...)" - вывод в геометрическом приеме таких дополнительных утверждений, характеризующих чертеж, которые полезны лишь при выполнении специальных условий.

Особый случай - занесение дополнительных посылок, используемых при сопровождении по области допустимых значений. Такие посылки необходимо своевременно формировать в процессе преобразований, чтобы легко было усматривать выполнение условий по о.д.з. на новые термы, введенные приемом. Так как выполнение даже самых простых алгебраических преобразований связано с проверкой выполнения условий по о.д.з., отсутствие сопровождающих утверждений может полностью заблокировать решение задачи. Отличие от общего случая со вводом дополнительных посылок заключается, во-первых, в том, что сопровождающее утверждение может быть занесено не только в список посылок, но и в список условий (если прием преобразует условие задачи на описание), и во-вторых, в том, что может происходить автоматический контроль неиспользуемости для сопровождения по о.д.з. ранее введенных утверждений, с устранением их по мере возможности. Здесь используется указатель "вывод(условие(A)C эквивалентно($i_1 \dots i_n$))"; A - фильтр проверки целесообразности вывода дополнительного утверждения C; $i_1, \dots, i_n$ - номера некоторых антецедентов теоремы, истинность которых является следствием утверждения C и остальных антецедентов теоремы (подтерм "эквивалентно(...)", как и "условие(...)", может отсутствовать). Утверждения задачи, с которыми были идентифицированы указанные утверждения, если они после выполнения преобразования приема не используются в качестве сопровождающих по о.д.з., удаляются. В качестве примера ситуации, требующей сопровождения преобразований вводом дополнительной посылки с помощью указателя "вывод", рассмотрим исключение иррациональности в знаменателе выражения $a/(\sqrt{(}b) - \sqrt{(}c))$ путем домножения числителя и знаменателя на $\sqrt{(}b) + \sqrt{(}c)$. Знаменатель $b - c$ новой дроби может быть существенно видоизменен после приведения подобных членов, и для того, чтобы последующие приемы не испытывали затрудений при усмотрении отличия его от нуля, нужно сразу же вводить посылку $b - c = 0$. Так как нормализаторы применяются ко всем вхождениям терма $b - c$, то выполняемое ими приведение подобных членов будет предпринято и в новом знаменателе, и в сопровождающей посылке.

Если решается задача на описание, то появляется возможность выполнять преобра-

зование подвыражения ее условия, используя в заменяющем терме вспомогательные обозначения, определяемые с помощью вводимых тем же приемом новых условий. Для этой и возможных других целей служит указатель "выводусловия(A $B_1 \dots B_n$)", позволяющий сопровождать основные действия приема занесением в список условий текущей задачи на описание нового условия A. Данное условие снабжается комментариями, вводимыми в термах B_i вида "комментарий(...)"; возможен случай $n = 0$.

Удаление посылок и условий

Если в результате применяемого преобразования пропадает терм, требовавший специального сопровождения по о.д.з., то прием может предпринять попытку исключить утверждения, обеспечивавшие такое сопровождение. В случае исключения посылки для этого вводится указатель "удалениепосылки(условие(A)B)"; в случае удаления условия - указатель "удалениеусловия(условие(A)B)". Здесь B - удаляемое утверждение (в теоремных переменных); A - фильтр, проверяющий целесообразность удаления (он может отсутствовать). Если B действительно является посылкой (соответственно, условием) текущей задачи, то после выполнения определяемой приемом замены проверяется, используется ли оно все еще в качестве сопровождающего по о.д.з. утверждения, и если не используется, то удаляется. Пример использования указателя - при решении уравнения $\mathrm{tg}(x) = a$ предпринимается попытка удалить условие $\neg(\cos(x) = 0)$, обеспечивавшее до этого сопровождение по о.д.з для устраненного тангенса.

Новые переменные, вводимые приемом

Прием может вводить новую переменную - такая переменная не имеет вхождений в идентифицируемые термы приема, но встречается в формируемых приемом новых термах. Если речь идет о связанной переменной, возникающей при вводе нового квантора либо описателя, то никаких специальных указателей для нее не требуется - компилятор обеспечит ввод ее автоматически. В противном случае нужны специальные указатели, уточняющие способ ввода данной переменной. При отсутствии этих указателей компиляция приема не состоится.

1. Простейший указатель ввода новой переменной X имеет вид "новаяпеременная(X)". При его наличии компилятор просто выберет для построения нового терма с X некоторую не встречающуюся в текущем контексте переменную. Обозначение для нее будет выбираться, по мере возможности, с учетом условий на X из антецедентов теоремы и с учетом контекста задачи - для неизвестных будут выбраны $x, y, z, \dots$, для целочисленных переменных - $m, n, k, \dots$, для точек - $A, B, C, \dots$, и т.д. Часто встречающаяся ситуация со вводом новой переменной - использование ее в качестве вспомогательного обозначения. Например, вычисляется производная некоторой функции f, обозначаемая новой переменной g, и в посылки задачи заносятся как утверждение "Производная(f, g)", связывающее g с f, так и равенство, явно определяющее g.

2. Иногда нужно ввести не единственную переменную, а сразу кортеж переменных, длина которого определяется в зависимости от контекста. Например, это бывает при работе с функциями многих переменных. Тогда применяется указатель "переменные(X N)", определяющий, что переменная X идентифицируется

с набором новых переменных, длина которого определяется значением теоремной переменной N, идентифицированной с десятичной записью числа.

3. Если новые переменные $X_1, \ldots, X_n$ должны быть присоединены к списку неизвестных текущей задачи на описание либо исследование, то используется указатель "новыенеизвестные $(X_1 \ldots X_n)$".

4. Если прием вводит дополнительные построения - рассматривает некоторые новые объекты, условия на которые заносятся в список посылок текущей задачи, то особо должен быть учтен случай задачи на исследование - в этом случае предпринимается перенесение основных условий на новые объекты в список посылок внешней задачи на описание. Здесь используется указатель "новыйсимвол$(X\ K_1 \ldots K_n)$", где X - новая переменная; $K_1, \ldots, K_n$ - указатели вхождений в теорему основных утверждений о свойствах X, которые целесообразно перенести в список посылок внешней задачи на описание. Хотя в процессе обучения решателя оказалось, что такая предусмотрительность является лишь формальностью - список посылок внешней задачи на описание фактически полностью отключен от процесса решения - указатель "новыйсимвол(...)" применяется во всех приемах по планиметрии, связанных с дополнительными построениями.

5. При решении задачи на исследование, связанной с задачей на описание вычислительного типа (например, вычислительные задачи в планиметрии), либо при решении задачи на доказательство путем вычислений в посылках, существенным является разделение переменных на неизвестные внешней задачи, неизвестные "внутреннего" характера (это просто обозначения вспомогательных объектов, которые не должны войти в окончательный ответ, например, обозначения точек на чертеже), и известные параметры. Часто для решения задач такого типа применяется прием, вводящий как бы известный новый параметр, через который далее будут выражаться прочие переменные задачи. Впоследствии либо окажется, что удалось найти выражения для неизвестных, в которых такой вспомогательный параметр "сократился", либо будут предприняты дополнительные действия по определению его значения после выражения через него неизвестных задачи. Тогда он сам поменяет статус вспомогательной "известной" на вспомогательную "неизвестную". Для первоначального ввода вспомогательного "известного" параметра X применяется указатель "вспомпараметр$(X$ $K_1 \ldots K_n)$", где $K_1, \ldots, K_n$ - указатели вхождений в теорему тех утверждений о X, добавляемых при вводе этого параметра к списку посылок, которые одновременно должны быть зарегистрированы в списке посылок внешней задачи на описание. Заметим, что при вводе вспомогательного параметра все веса посылок задачи обращаются в 0; это способствует срабатыванию полезных приемов при повторном сканировании ее посылок, независимому от введения дополнительных "известных" параметров. Заметим также, что один прием может вводить несколько вспомогательных параметров (каждый требует своего указателя).

6. Другой способ придать нужную целенаправленность действиям по решению задачи на исследование, имеющей цель "известно", либо задачи на доказательство - ввести вспомогательную неизвестную X, которая в случае задачи на исследование переносится также в список неизвестных внешней задачи на описание, однако найденные ее значения не включаются в ответ такой задачи. Фильтры многих и многих приемов (например, по планиметрии) активно

реагируют на связь с неизвестными внешней задачи тех или иных числовых параметров текущей задачи (расстояния, углы и проч.), так что фактически ввод вспомогательной неизвестной оказывается примерно равноценен решению вспомогательной задачи на нахождение значения такой неизвестной. Впрочем, вычисление ее значения происходит здесь в более гибком режиме - параллельно с прочими действиями по решению основной задачи, что часто гораздо более предпочтительно, чем грубое прерывание общего процесса анализа посылок и локализация внимания только на вспомогательном вычислении. Для указания на то, что прием вывода фактически вводит обозначение для новой неизвестной X, используется указатель "вспомнеизвестная(X)". Прием может вводить сразу несколько вспомогательных неизвестных, связанных между собой утверждениями, присоединяемыми им к списку посылок. В этом случае вводятся несколько отдельных указателей "вспомнеизвестная(...)". В случае задачи на доказательство ввод вспомогательной неизвестной сопровождается понижением до 0 весов всех посылок и веса условия.

Обработка надтермов

Если применяется преобразование тождественной либо эквивалентной замены, то оно может повлечь за собой различного рода нарушения стандартизации вида термов, и как следствие - цепочку срабатываний мелких нормализующих приемов. Такого рода цепочки срабатываний увеличивают число сканирований и могут ощутимо повлиять на время решения задачи. Для борьбы с ними, в первую очередь, применяются различного рода нормализаторы, обеспечивающие необходимую стандартизацию внутри самого заменяющего терма. Однако, эти нормализаторы не выполняют восстановления стандартизации для надтермов заменяемого терма. Чтобы выполнять такую стандартизацию надтермов, в оператор ЛОСа "заменавхождения(...)", используемый для реализации тождественных и эквивалентных замен при сканировании задачи, встроена процедура, последовательно просматривающая надтермы (в направлении их увеличения) и обрабатывающая их посредством пакетных нормализаторов общей стандартизации (эти нормализаторы находятся с помощью справочника "нормализатор"). Эта процедура активизируется, если прием выделен указателем "норм". Примером использования этого режима служит группа приемов, выполняющих сложение и упрощение дробей - для проверки возможности применения последующей стандартизации дробей (деление дроби; деление на дробь; сокращение дроби и т.п.).

Аналогичный механизм стандартизации надтермов может быть включен при помощи указателя "нормализатор". Такой указатель обычно выделяет приемы общей стандартизации. После некоторых замен, нарушающих стандартизацию для надтермов замененного терма, может возникнуть длинная цепочка простых стандартизирующих преобразований. Эти преобразования, в целях уменьшения трудоемкости, выгодно применять вне сканирования. Поэтому, как только к одному и тому же терму задачи применяются подряд два приема, снабженные указателем "нормализатор" (что может служить сигналом о наличии указанной выше длинной цепочки стандартизации), автоматически включается специальная процедура, выполняющая полную обработку этого терма пакетными нормализаторами. Эта же процедура обеспечивает необходимую коррекцию структур данных, выполняющих сопровождение по о.д.з. Обращение к указанной процедуре (заголовком ее служит логический символ "блок-

нормализации") происходит из программы "заменавхождения", обеспечивающей замены при сканировании задачи.

Коррекция содержимого буферов обращений к пакетным операторам

После выполнения преобразования может сложиться ситуация, когда автоматическое использование таких хранящихся в буфере обращений к заданному нормализатору результатов, которые содержат некоторый терм t, нецелесообразно - прием только что "избавился" от этого терма. Здесь применяется указатель "контрольнормализации(A t)", инициирующий исключение из буфера обращений к нормализатору A всех результатов, содержащих подтерм t. Возможно также исключение из буфера всех результатов обращений к A - для этого достаточно не указывать t.

Особые случаи сопровождения по области допустимых значений

1. Если прием выполняет замену некоторого утверждения, которое может использоваться в качестве сопровождающего по о.д.з. (при отключенной блокировке таких замен - на это отключение приходится идти при завершающем редактировании ответа), то рекомендуется сохранять информацию о том, что его исходная версия выводима из новой версии (а также из прочих обеспечивавших корректность замены утверждений из текущего контекста), вводя указатель "стандследствие". Эта информация сохраняется в комментарии "стандследствие ..." к посылкам исходной задачи и может быть использована далее оператором "стандследствие", к которому обращаются все проверочные операторы.

2. Если сопровождающие по о.д.з. утверждения нецелесообразно регистрировать в списках посылок и условий задачи, но тем не менее указания на то, что они являются следствиями конкретных посылок и условий, могут пригодиться, то применяется указатель "вычерк". Прием в этом случае будет сохранять информацию о сопровождающих утверждениях в комментарии "стандследствие ...". Фактически такой указатель был использован только для приема, выполняющего формальное интегрирование путем обращения к нормализатору "нормИнтеграл".

3. Иногда, наоборот, целесообразно зарегистрировать сопровождающие по о.д.з. условия для новых термов, даже не проверяя того, что они суть следствия старых условий и посылок (например, они могут возникнуть при вычислении пределов либо суммировании рядов как необходимые условия существования предела либо суммы ряда). Тогда вводится указатель "внешвывод" - он инициирует автоматическое перенесение всех новых утверждений из накопителя "коррекцияпосылок ..." в контекст замены (например, в список посылок).

4. В определенных случаях сопровождение по о.д.з. может все-таки нарушаться. Это не означает, что в контексте требующего сопровождения терма отсутствуют утверждения, следствием которых являются необходимые условия на о.д.з. - просто усмотрение из них этих условий неочевидно, и в комментариях явных ссылок на эти утверждения как на сопровождающие по о.д.з. нет. Такая ситуация возникает, как правило, лишь в особых случаях - после обращения к вспомогательным задачам на описание. При редактировании ответа данной задачи

все сопровождающие по о.д.з. утверждения могут быть явно разрешены относительно своих параметров (такова обычная стандартизация ответов), и тогда явное сопровождение становится неявным. Если полученный таким образом ответ используется далее в задаче, то явное сопровождение необходимо восстанавливать - без него будут заблокированы многие важные приемы. Восстановление обеспечивается специальными приемами, снабженными указателем "коррекцияодз". Эти приемы суть обычные приемы вывода в посылках либо условиях; применение их начинается с усмотрения в задаче требующего сопровождения по о.д.з. терма A (например, квадратного корня) - он определяется указателем "контрольвывода(A)". Кроме регистрации в посылках либо условиях сопровождающих утверждений, прием осуществляет соответствующую коррекцию комментария "сопровождение"к текущей задаче.

Пополнение ответа задачи

Теорема приема "ответзадачи" явно перечисляет список условий на неизвестные, которые выдаются в качестве ответа. Иногда этот ответ целесообразно пополнить некоторыми утверждениями, отсутствующими в указанном списке, но являющимися его следствиями. Тогда применяется указатель "плюс(условие(A)$B_1 \dots B_n$)", где A - фильтр, определяющий целесообразность присоединения к ответу утверждений $B_1, \dots, B_n$ (фильтр может отсутствовать).

Расширение списков переменных, указанных в целях задачи

Вводимые при решении задачи новые переменные иногда бывает нужно зарегистрировать в тех или иных целях текущей задачи. Для этого можно использовать указатель "установка(A X)". Переменная X заносится тогда в список $B_1 \dots B_n$ цели, имеющей вид $(A\ B_1 \dots B_n)$. Этот же указатель можно применять и в ситуации, когда заносимый в цель $(A \dots)$ объект - не переменная, а терм. Тогда вместо X в указателе помещается запись "терм(T)", где T - заносимый терм.

Для достаточно часто возникающей необходимости пополнять новыми переменными цель "обозначение $X_1 \dots X_k$" (она перечисляет вспомогательные переменные, которые не должны входить в ответ задачи на описание либо преобразование) предусмотрен специальный указатель "обозначения($x_1 \dots x_p$)", где $x_1, \dots, x_p$ - добавляемые к цели "обозначение ..." переменные.

11.4.7 Указатели, уточняющие точку привязки приема

Приемы сканирования задачи инициализируются при рассмотрении вхождения в задачу некоторого логического символа. Это вхождение называется точкой привязки приема; для уточнения ее вида служат следующие указатели:

1. Если прием определяет вывод следствий в посылках либо условиях задачи, то инициализация попытки его применения не всегда связана с обнаружением при сканировании задачи явного вхождения одного из антецедентов теоремы приема. Иногда она может быть вызвана обстоятельствами весьма косвенного характера - обнаружением где - либо в задаче некоторого выражения либо утверждения T, при работе с которым могут понадобиться формулируемые

приемом свойства входящих в T объектов. Чтобы инициализация приема начиналась с обнаружения при сканировании терма T, используется указатель "контрольвывода(T)". В качестве примера рассмотрим прием вывода следствий в посылках, записывающий соотношение для площади равнобедренного треугольника:

$l(AB) = l(AC)$ & $D \in$ прямая(BC) & $l(BD) = l(CD) \rightarrow 2S($фигура$(ABC)) = l(AD)l(BC)$.

Указатель "контрольвывода(площадь(фигура(набор(ABC))))" определяет в качестве точки привязки приема вхождение выражения $S($фигура$(ABC))$, которое в антецедентах теоремы не встречается.

2. Чтобы явно указать логический символ A, к которому должна быть отнесена программа приема - тогда его срабатывание будет начинаться с усмотрения при сканировании некоторого вхождения символа A - используется указатель "точкапривязки(A)".

3. Как правило, заменяемый терм теоремы для приема замены должен быть невырожденным, не сводящимся к обычной либо функциональной переменной, иначе трудно определить точку привязки. Однако, иногда бывает нужно идентифицировать такой вырожденный терм однозначным образом - с условием текущей задачи на преобразование. В этом случае применяется указатель "корень".

4. Если точку привязки требуется выбрать в антецеденте теоремы, имеющем номер i, то вводится указатель "теквхожд(i)". Фактически оказалось, что он стал использоваться лишь в сочетании с указателем "усм(i)" для антецедента, имеющего вид утверждения о перпендикулярности. Исключение составляет единственный случай, когда из соображений переключения внимания в приеме замены понадобилось точку привязки выбрать в обычном (идентифицируемом с посылкой задачи) антецеденте.

5. Чтобы выделить в приеме "ответзадачи" тот конъюнктивный член теоремы, который желательно использовать в качестве точки привязки, применяется указатель "смответ(i)", где i - символьный номер данного конъюнктивного члена; нумерация начинается с 1.

11.4.8 Указатели, определяющие преобразования комментариев

Ввод нового общего комментария в случае применения приема

Чтобы при срабатывании приема вводился новый комментарий к текущей задаче (в случае задач на исследование - комментарий к посылкам задачи) либо комментарий к пакетному оператору, используется указатель "замечание($A_1 \ldots A_n$)". A_1 - логический символ, являющийся заголовком комментария; $A_2, \ldots, A_n$ - операторные выражения для элементов комментария. Типы допустимых здесь выражений - те же, что при задании комментариев в фильтрах приемов. Возможен случай $n = 1$, и тогда комментарий состоит из логического символа A_1. Комментарий вводится лишь непосредственно перед обращением к процедуре ЛОСа, выполняющей преобразования приема. Однако, в определенных случаях блокировка срабатывания может

произойти уже внутри последней процедуры - если преобразование оказалось вырожденным, либо нарушающим сопровождение по о.д.з, либо вследствие учета каких-либо общих установок на решение задачи. Если в таких случаях важно не допустить ввода комментария, то перед A_1 в указанной выше записи добавляется символ "результат". Комментарий формируется с отбрасыванием этого символа, и вводится только после фактического выполнения преобразования. Наконец, перед A_1 возможно поместить терм "условие(B)", где B - фильтр, и тогда комментарий будет вводиться только при истинности условия B.

Если применяется прием нормализатора, но комментарий требуется ввести не для нормализатора, а для текущей задачи, то перед A_1 в указателе "замечание(...)" помещается логический символ "текзадача".

По умолчанию, разрешается вводить множество копий одного и того же комментария. Иногда это используется для контроля глубины цепочки однотипных преобразований (например, применений правила Лопиталя для вычисления пределов). В тех случаях, когда требуется предотвратить дублирование комментариев, для ввода нового комментария нормализатора либо анализатора, вместо "замечание(...)" применяется указатель "замечусловие($A_1 \dots A_n$)".

Если требуется ввести комментарий к посылкам текущей задачи (при применении приема сканирования задачи), то используется указатель "комментариипосылок($A_1 \dots A_n$)". Чтобы гарантировать ввод комментария только в случае фактического применения преобразования приема, перед A_1 помещается логический символ "результат".

Ввод комментария к новому либо преобразованному терму задачи

Если требуется снабдить комментарием новую посылку либо условие, сформированные при выводе следствий, либо преобразованный приемом терм задачи, либо новое утверждение, введенное анализатором, то используется указатель "примечание($A_1 \dots A_n$)" либо "примечание(условие(B $A_1 \dots A_n$))". Набор $A_1, \dots, A_n$ определяет комментарий; B - фильтр, указывающий дополнительное условие для ввода этого комментария.

Ввод комментария к терму задачи, идентифицированному с заданным антецедентом

Новым комментарием можно снабдить посылку либо условие задачи, идентифицированные с заданным антецедентом теоремы приема. Для этого используется указатель "примечпосылки(i $A_1 \dots A_n$)". i - номер антецедента; набор $A_1, \dots, A_n$ определяет вводимый комментарий. Формат задания комментария - такой же, как в случае "замечание(...)".

Проверка отсутствия комментария с последующим его вводом

Комментарии могут играть роль "защелки", пропуская внутрь приема при первой попытке применения и блокируя повторные попытки. В таких случаях в процессе применения приема сначала предпринимается проверка отсутствия заданного комментария, а впоследствии, в зависимости от типа примененного указателя, его ввод. Используются указатели следующих типов:

1. Простейший тип защелки - ввод комментария (к задаче, нормализатору либо анализатору) непосредственно после проверки его отсутствия. Здесь применяется указатель "ключ($A_1 \dots A_n$)", где A_1 - заголовок комментария; $A_2, \dots, A_n$ - его элементы. Формат термов $A_2, \dots, A_n$ при работе с защелками отличается от обычного, использованного в фильтрах и ранее описанных указателях. Каждый из них представляет собой либо логический символ (при этом служебный символ "теквхожд" обозначает текущий идентифицируемый в приеме подтерм), либо терм в теоремных переменных, либо определяющий вхождение терм "вхождение(C)", где C - операторное выражение.

   Проверка отсутствия комментария и его ввод происходят немедленно после идентификации всех переменных термов $A_2, \dots, A_n$. Если этот момент нужно отложить до того, как дополнительно будут идентифицированы переменные $X_1, \dots, X_m$, то к списку $A_2, \dots, A_n$ добавляется элемент "фикс($X_1 \dots X_m$)" (он никак не изменяет самого комментария). Использование в $A_2, \dots, A_n$ указателей вхождения в теорему не предусмотрено. Еще один способ отложить момент проверки отсутствия комментария и его ввода - размещение в конце списка $A_2, \dots, A_n$ терма "посылки($i_1 \dots i_k$)", перечисляющего номера тех антецедентов, истинность которых должна быть установлена до указанного момента.

2. Следующий тип "защелки" - проверка отсутствия комментария и ввод его при откате, если прием оказался не реализованным. Указатель здесь имеет вид "попытка($A_1 \dots A_n$)". Как и в предыдущем случае, проверка отсутствия комментария происходит сразу же после идентификации всех переменных, входящих в термы $A_2, \dots, A_n$. Чтобы отложить этот момент, можно применять термы "фикс($X_1 \dots X_m$)" и "посылки($i_1 \dots i_k$)".

3. Если проверка отсутствия комментария и его ввод должны происходить непосредственно перед применением преобразования приема, то используется указатель "биключ($A_1 \dots A_n$)". Записи "фикс(...)" и "посылки(...)" здесь излишни.

4. Предшествующие указатели вводили общий комментарий задачи (в случае задач на исследование - комментарий к посылкам задачи) либо пакетного оператора. Чтобы проверить отсутствие комментария к текущему терму задачи (внутри которого находится точка привязки приема) и сразу после этого ввести данный комментарий, применяется указатель "первыйключ($A_1 \dots A_n$)". Проверка происходит сразу же после идентификации переменных термов $A_2, \dots, A_n$; чтобы отложить этот момент, можно применять записи "фикс($X_1 \dots X_n$)" и "посылки($i_1 \dots i_k$)".

5. Чтобы ввести комментарий к текущему терму задачи не сразу после проверки отсутствия этого комментария, а лишь при откате, если прием не был реализован, применяется указатель "Попытка($A_1 \dots A_n$)". Он совершенно аналогичен приведенному выше указателю "попытка($A_1 \dots A_n$)".

Коррекция ранее введенного комментария

Чтобы изменить ранее введенный комментарий, применяется указатель "изменение($A\ B_1 \dots B_n\ C_1 \dots C_k$)". Здесь A - идентифицирующий терм "комментарий(...)" либо "комментарийпосылки(...)" либо "примечание(...)"; $B_1, \dots, B_n$ - фильтры, определяющие дополнительные условия на введенные в терме A разряды преобразуемого

комментария (могут отсутствовать); $C_1, \ldots, C_k$ - термы вида "замена(i P)", где i - номер изменяемой позиции комментария (номер 0 имеет заголовок комментария); P - операторное выражение, значением которого является заменяющий объект.

Часто встречается ситуация, в которой комментарий используется как накопитель объектов некоторого типа. При первом появлении такого объекта комментарий должен быть введен, а при каждом последующем появлении - список объектов в уже имеющемся комментарии должен быть пополнен. В случае комментариев к посылкам задачи данный режим вводится указателем "Примечание(A B)". A - логический символ, являющийся заголовком комментария. Комментарий имеет вид (A N), где набор N - накопитель объектов. Операторное выражение B должно определять логический символ либо переменную.

Удаление комментария

При выполнении приема замены, реализуемого в сканировании задачи, можно удалить заданный комментарий. Для этого используется указатель "удалениезамечания(A $B_1 \ldots B_n$)". Здесь A - идентифицирующий терм "комментарий(...)" либо "комментарийпосылки(...)" либо "примечание(...)"; $B_1, \ldots, B_n$ - фильтры, определяющие дополнительные условия на введенные в терме A разряды удаляемого комментария (могут отсутствовать). В идентифицирующем терме A допускается явное указание комментария, без каких-либо идентифицируемых переменных. Если же такие переменные есть, то удаляются все комментарии выделенного типа.

Аналогичным образом, для удаления комментария к нормализатору используется указатель "удалениезамечания(B $A_1 \ldots A_n$)". Здесь B - фильтр, определяющий условие, при котором происходит удаление комментария; $A_1, \ldots, A_n$ - набор, определяющий удаляемый комментарий - в стандартном формате, как для указателя "замечание(...)".

Ввод комментариев при обращении к проверочному оператору, синтезатору либо задаче на доказательство, обрабатывающим заданный антецедент

1. Если для проверки истинности заданного антецедента либо его реализации происходит обращение к вспомогательной задаче, проверочному оператору либо синтезатору, то при обращении им может быть передан заданный комментарий. Для этого используется указатель "комментарий(i условие(B) $A_1 \ldots A_n$)". Здесь i - номер антецедента; B - фильтр, определяющий условие ввода комментария; ($A_1 \ldots A_n$) - набор, определяющий комментарий. Формат задания комментария здесь стандартный (такой же, как в указателе "замечание(...)").

2. Если происходит обращение к задаче на доказательство антецедента, имеющего вид кванторной импликации $\forall_{x_1 \ldots x_n}(P_1 \ \& \ \ldots \ \& \ P_m \rightarrow P_0)$, то $P_1, \ldots, P_m$ становятся посылками данной задачи. Утверждение P_i как посылку при этом можно снабдить заданным комментарием. Для этого используется указатель "комментарий(K условие(B)$A_1 \ldots A_n$)", аналогичный приведенному выше. Однако, здесь уже K является не номером антецедента, а указателем вхождения в теорему приема утверждения P_i

3. При обращении из пакетного оператора к проверочному оператору либо синтезатору, обрабатывающему некоторый антецедент, по умолчанию происходит

передача комментариев текущего оператора к подоператору. Чтобы передавать не все такие комментарии, а только имеющие заголовки $A_1, \ldots, A_n$, используется указатель "комментарии(i $A_1 \ldots A_n$)", где i - номер антецедента. Если нужно, наоборот, удалить все комментарии с данными заголовками, то используется указатель "удалениепримечания(i B $A_1 \ldots A_n$)", где B - фильтр, определяющий условия, при которых происходит удаление (может отсутствовать).

4. Наконец, предусмотрено удаление конкретных комментариев при обращении из пакетного оператора к проверочному оператору либо синтезатору. Здесь используется указатель "исключение(i условие(B)$A_1 \ldots A_n$)", где i - номер антецедента; B - фильтр, определяющий условия, при которых происходит удаление комментария; $A_1, \ldots, A_n$ - набор, задающий комментарий в стандартном формате (как для указателя "замечание(...)").

Ввод комментария к вспомогательной задаче, возникающей при попытке замены

Прием замены, имеющий указатель "попытказамены", не предпринимает, как уже говорилось выше, непосредственного изменения текущей задачи, а вводит вспомогательную задачу - копию текущей, в которой уже и реализуется данная замена (при неудаче решения последней происходит откат к прерванному решению текущей задачи). Возможна передача комментария этой вспомогательной задаче. Здесь применяется указатель "Замечание($A_1 \ldots A_n$)", где набор $A_1, \ldots, A_n$ определяет комментарий в том же формате, как и для указателя "замечание(...)".

Специальные комментарии, используемые для учета взаимной выводимости утверждений

Во многих случаях приемы вывода следствий в действительности определяют эквивалентное преобразование одной из использованных посылок. Например, при получении линейной комбинации двух уравнений с ненулевыми коэффициентами возможен обратный логический переход от нее к одному из исходных уравнений, в предположении истинности второго уравнения. Учет такого рода обратных переходов может существенно сократить трудоемкость решения задачи, устраняя проверку исходных уравнений. Для реализации его применяется указатель "прообраз(i посылки($A_1 \ldots A_n$) вывод($B_1 \ldots B_m$))". Здесь $A_1, \ldots, A_n$ - дополнительно проверяемые проверочными операторами утверждения. Если удалось установить их истинность, то прием дополнительно выводит следствия $B_1, \ldots, B_m$ и регистрирует в комментарии "прообраз ..." тот факт, что i - й антецедент является следствием консеквента, прочих использованных при выводе утверждений из контекста идентификации, а также утверждений $B_1, \ldots, B_m$. Списки $A_1, \ldots, A_n$ и $B_1, \ldots, B_m$ могут быть пустыми. Заметим, что фактически указатель "прообраз(...)" применялся только в приемах анализаторов.

Использование фильтра, проверяющего существование объектов некоторого типа, для сохранения в комментариях списка всех таких объектов

Если прием нормализатора должен не только проверить при оценке целесообразности своего срабатывания существование объектов, удовлетворяющих заданному

условию, но также передать внешним приемам список всех таких объектов либо некоторых характеристик этих объектов, то можно разместить внутри фильтра "контекст(...)" (например, в качестве последнего операнда) указатель "буфер(A R)". В этом случае данный фильтр не ограничится отысканием единственного набора значений своих внутренних парметров, при котором его условия истинны, а станет перечислять все такие значения. Для каждого из них будет определяться значение операторного выражения R, и это значение будет регистрироваться в накопителе N комментария (A N); повторяющиеся элементы в накопитель не заносятся. При обращении к нормализатору должен быть введен его комментарий с заголовком A. Фактически данный режим был использован в нормализаторе "повторчисло", обеспечивающем выделение в явном виде повторяющихся вхождений подвыражений с неизвестными. После того, как некоторый прием этого нормализатора изменяет одно из уравнений таким образом, чтобы в нем возникло выражение с неизвестными, встречающееся в другом уравнении, ссылка на такое внешнее уравнение заносится в специальный комментарий, который по окончании работы с текущим уравнением инициирует повторное рассмотрение зарегистрированных в нем уравнений.

Ввод комментариев к внешнему нормализатору

Иногда бывает нужно передать комментарий одной из внешних процедур - согласно цепочке обращений, в конце которой находится текущая реализуемая процедура. Например, это может быть связано с передачей сообщения о целесообразности разбора случаев, которая выяснилась лишь в текущей подпроцедуре. Если внешняя процедура представляет собой нормализатор, то для такой передачи комментария предусмотрен указатель "нормализация(A B $C_1 \ldots C_n$)". Здесь A - заголовок внешнего нормализатора; B - фильтр, определяющий условие передачи комментария; $C_1, \ldots, C_n$ - набор, определяющий комментарий в стандартном формате. В фильтре B можно ссылаться на текущий обрабатываемый внешним нормализатором A терм посредством служебного логического символа "внешкорень". Если имеется несколько внешних нормализаторов с заголовком A, для которых выполняется условие B, то комментарий передается каждому из них. Фильтр B может отсутствовать.

Предусмотрен вариант, в котором комментарий передается не всем внешним нормализаторам A (если их несколько), а лишь тому из них, который ближе всего по цепочке обращений расположен к текущей процедуре. Здесь используется указатель "учетнормализации(A B $C_1 \ldots C_n$)".

11.4.9 Указатели, определяющие переключение внимания

После того, как был применен прием, может оказаться вероятным срабатывание некоторых других приемов. Чтобы такое срабатывание стало возможным, необходимо уменьшить веса тех посылок либо условий, при рассмотрении которых эти другие приемы активизируются - "переключить внимание". Наиболее вероятные точки, для которых целесообразно осуществить переключение внимания, выявляются в процессе обучения решателя на потоке задач (например, просто сохраняются указания на переключение внимания, оказавшиеся необходимыми для решения отдельных задач обучающей выборки). Ниже приводятся типы указателей, обеспечивающих переключение внимания.

1. Наиболее часто используемый указатель - "новаяпосылка(X A N)". В этом указателе A представляет собой фильтр, определяющий условие на посылку X текущей задачи, вес которой должен быть понижен до N (если он уже не был меньшим или равным N). N - явно указываемое символьное число; X - новая переменная, используемая в контексте данного указателя - она пробегает все посылки текущей задачи. Если перед X размещается запись "условие(P)", то понижение весов будет предприниматься только при истинности фильтра P.

2. Аналогичным образом используется указатель "новоеусловие(X A N)", у которого X пробегает все условия текущей задачи на описание. В приеме явно должно быть указано, что он применяется к задаче на описание.

3. Если требуется понизить до величины N вес посылки, с которой был идентифицирован i - й антецедент теоремы, то применяется указатель "актив(i N)".

4. По умолчанию, при компиляции приема сканирования задачи, в начале его программы вставляется оператор "новый". Этот оператор блокирует применение приема в циклах повторного сканирования терма T задачи, попавшего на некоторый период в "теневую зону" сканирования, предпринимаемых перед выходом его из "теневой зоны" (т.е. когда текущий уровень задачи становится на 1 меньше веса данного терма). Однако, если срабатывание приема существенно зависит от возможности появления за период пребывания в "теневой зоне" терма T новых посылок либо условий, такой оператор нежелателен. Устранение его обеспечивается применением указателя "новый". Заметим, что в предметных областях, характеризующихся большими длинами списка посылок (например, в геометрии) неосторожное применение указателя "новый" может существенно замедлить работу решателя. Компенсировать его отсутствие приходится явными указателями на переключение внимания.

5. Чтобы понизить до 0 веса всех условий и посылок задачи, содержащих некоторый терм T (записанный в теоремных переменных), используется указатель "внимание(T)". Если вместо T использовать служебный символ "условие", то понижается до 0 вес условия задачи. Эту возможность следует применять лишь тогда, когда в описании приема явно указывается, что он применяется для задачи на преобразование либо на доказательство, т.е. ссылка на условие задачи корректна.

6. Если прием изменяет посылку либо условие, либо вводит новую посылку или условие, то измененный либо новый терм задачи получает вес 0, цикл сканирования прерывается и возобновляется с нулевого текущего уровня. Если же прием изменяет только комментарии задачи, то такого прерывания цикла сканирования не происходит. Однако, иногда после ввода нового комментария необходимо искусственно понизить до 0 вес текущего терма задачи и возобновить сканирование с нулевого текущего уровня - чтобы сработали те приемы, которым адресован комментарий. В этом случае прием с заголовком "замечание" (изменяющий только комментарии задачи) снабжается указателем "пересмотр".

11.4.10 Ограничитель трудоемкости

Для предотвращения неоправданно длительных попыток применения приема можно использовать несколько специальных указателей, вводящих слежение за трудоемкостью попытки реализации приема и обрывающих эту попытку в случае превышения установленного лимита. В связи с обработкой заданного антецедента теоремы уже был описан ограничитель трудоемкости "Обрыв(...)". Здесь приведен еще один указатель такого типа - "лимит(N $A_1 \ldots A_n$)". У него N - число шагов работы интерпретатора ЛОСа, по превышении которого попытка применения приема обрывается. $A_1, \ldots, A_n$ - термы, используемые для уточнения той позиции в программе приема, после которой устанавливается лимит в N шагов. Именно, устанавливающий лимит оператор ЛОСа вводится сразу же, как только оказываются идентифицированы все переменные данных термов. Термы $A_1, \ldots, A_n$ могут отсутствовать.

11.4.11 Отложенная фильтрация

При применении приема не всегда удается сразу хорошо оценить действительные последствия его действий. Однако, часто можно сформулировать достаточно легко проверяемое условие P на желательные последствия. Тогда используется "отложенная" фильтрация для приема: проверка данного условия предпринимается не сразу, а лишь через заданное число шагов N. Эта фильтрация задается с помощью указателя "контроль(P лимит(N) K)". Если через N шагов не будет выполнено условие P, то произойдет откат к тому состоянию задачи, которое имелось до применения приема. При этом задача будет снабжена комментариями, определяемыми термами "замечание(A)", перечисляемыми в списке K (этот список может быть пустым). Для организации режима отложенной фильтрации вводится комментарий (контроль ...) к списку посылок текущей задачи. Он анализируется процедурой "Контроль(...)", обращения к которой происходят при каждом выходе на программу логического символа - типа задачи при сканировании.

11.4.12 Специальные указатели пакетных операторов

Нормализаторы

1. Уровень срабатывания приема нормализатора устанавливается не фильтром, как в случае приемов остальных типов, а указателем "уровень(A)". Здесь предусматривается единственное значение уровня.

2. Если в описании формата нормализатора содержится логический символ "корень", то все его приемы будут применяться только к корневому вхождению в преобразуемый терм. Для прочих нормализаторов такое ограничение на срабатывание приема можно установить, введя указатель "корень". В частности, этот указатель может оказаться необходимым, если заменяемый терм представляет собой функциональную переменную.

3. Достаточно часто входная информация логического уровня предметной области передается нормализатору через комментарии. Например, нормализатор разложения в ряд Тейлора получает указание на переменную x, по которой ведется разложение, и на точку a, в окрестности которой оно выполняется, через

комментарий (рядтейлора x a). Для дополнительной идентификации теоремных переменных через такой комментарий используется указатель "вход(A $x_1 \dots x_n$)". Здесь A - заголовок комментария, который должен иметь вид (A b_1 $\dots$ b_n), причем переменные $x_1, \dots, x_n$ идентифицируются с $b_1, \dots, b_n$. Компилятор предполагает, что $b_1, \dots, b_n$ суть термы.

4. Если требуется, чтобы сразу после выполнения текущего преобразования нормализатор выдал полученный терм в качестве окончательного результата, то прием снабжается указателем "выход". В качестве примера использования такого указателя приведем приемы группировки тригонометрических выражений при разложении на множители (см. нормализатор "видумножение"). Прием, выполняющий некоторую группировку, предпринимает попытку разложить полученное выражение на множители, и если это удалось (а тогда автоматически и каждый из множителей будет подвергнут попыткам доразложения), то продолжение попыток разложения на множители результирующего выражения бесполезно.

5. Как уже отмечалось, при выполнении преобразований приемами сканирования задачи часто бывает необходимо обеспечивать специальные действия по поддержанию сопровождения по о.д.з. - иначе выполнение последующих приемов будет заблокировано из-за невозможности убедиться в выполнении самых элементарных требований на о.д.з. для рассматриваемых выражений. Аналогичные действия необходимо выполнять и в случае применения нормализатора. Для этого служит указатель "вывод(условие(P)A эквивалентно($i_1 \dots i_n$))". Этот указатель дополняет преобразование выводом следствия A, необходимого для поддержания сопровождения по о.д.з. A регистрируется в комментарии "коррекцияпосылок B" и заносится в список посылок нормализатора (фактически заносятся конъюнктивные члены A). P - фильтр, указывающий условие, при котором предпринимается вывод следствия. $i_1, \dots, i_n$ - номера антецедентов теоремы приема, истинность которых является следствием утверждения A и остальных антецедентов теоремы. Фильтр P и список $i_1, \dots, i_n$ могут отсутствовать. Заметим, что комментарий "коррекцияпосылок B" по завершении применения нормализатора используется для создания во внешней задаче, обратившейся к нормализатору, необходимых дополнительных посылок и комментариев, обеспечивающих сопровождение по о.д.з.

6. Если нормализатор обращается к преобразованиям вспомогательного выражения, содержащего новую переменную x, введенную в качестве обозначения для некоторого терма t, а затем возвращается к исходным обозначениям, заменяя в результате преобразований x на t, то может понадобиться коррекция сохраненной в комментарии "коррекцияпосылок A" информации о выводимости утверждений, содержащих x. Для этого используется указатель "подствпосылки(x t)", обеспечивающий подстановку t вместо x в таких утверждениях.

7. Иногда возникает необходимость таким образом подобрать значения входящих в выражение A параметров, чтобы это выражение оказалось возможным преобразовать к некоторому специальному виду. Один из примеров этой ситуации - подбор показателя степени, при котором степенная подстановка делает правую часть дифференциального уравнения зависящей только от отношения y к x. В таких случаях целесообразно воспользоваться нормализаторами, так как

подбору параметра обычно предшествует некоторая цепочка преобразований. В тот момент, когда становится "видимой" подсказка относительно возможного подбора параметра, применяется прием нормализатора, основанный на фиктивном тождестве. Левая (заменяемая) часть этого тождества идентифицируется с содержащим "подсказку" о подборе параметров фрагментом преобразуемого терма; правая часть представляет собой конъюнкцию равенств, определяющих подбор параметров. Собственно проверка допустимости данного подбора переносится в обработку антецедентов псевдотеоремы приема (например, здесь может быть предпринята явная подстановка параметров в терм и попытка преобразования его к требуемому виду, выполняемая другим нормализатором либо вспомогательной задачей). Прием оформляется как обычный прием тождественной замены. Однако, в то время как заменяемая часть его идентифицируется с подтермом преобразуемого терма, заменяющая часть вводится вместо всего преобразуемого терма. Этот режим обеспечивается указателем "нормвывод". Заметим, что обычно такие приемы являются завершающими, и в них используется также указатель "выход".

Проверочные операторы

1. Как правило, прием проверочного оператора имеет альтернативный характер: при неудаче его применения будут предприниматься попытки выполнить проверку другими средствами. Однако, иногда он сводит проверку истинности рассматриваемого утверждения к проверке истинности других утверждений (конъюнкция последних должна быть эквивалентна истинности первого), и при неудаче одной из этих проверок сразу выдается отказ. Под выдачей отказа здесь и далее понимаем, что значение проверочного оператора, как оператора ЛОСа, становится равным "ложь"; никакого логического символа "отказ" при этом не используется. Например, проверка отличия от 0 произведения нескольких выражений сводится указанным образом к проверке отличия от 0 каждого из сомножителей. Чтобы ввести такой режим функционирования приема, служит указатель "спуск". После проверки истинности "несущественных" антецедентов теоремы, указывающих типы значений переменных и другие необходимые элементы сопровождения по о.д.з., включается установка на выдачу отказа после неудачной проверки истинности хотя бы одного из оставшихся антецедентов.

2. Последний режим может применятсья также в тех случаях, когда эквивалентность проверяемого утверждения конъюнкции подмножества антецедентов теоремы усматривается лишь после проверки истинности части других ее антецедентов. Например, если при проверке нечетности суммы двух целых чисел установлено, что первое слагаемое четное, то проверка сводится к установлению нечетности второго слагаемого. В этой ситуации применяется указатель "спуск $(i_1 \dots i_k)$", перечисляющий номера $i_1, \dots, i_k$ тех антецедентов теоремы, после установления истинности которых, а также установления истинности "несущественных" антецедентов, включается установка на выдачу отказа после неудачной проверки истинности хотя бы одного из остальных антецедентов.

3. Наконец, прием проверочного оператора может сводить выполняемую проверку к рассмотрению конкретного антецедента, и если его истинность установить не удалось, то сразу будет выдан отказ. Для этого применяется указатель "внешобрыв(i)", где i - номер данного антецедента.

4. Иногда бывает достаточно найти хоть какую-нибудь идентификацию части антецедентов теоремы приема проверочного оператора, определив при этом значения $A_1, \ldots, A_n$ некоторых теоремных переменных $x_1, \ldots, x_n$, и проводить оставшиеся действия внутри данного приема лишь для указанных значений. Если эти действия довести до конца не удается, то альтернативные возможности идентификации для $x_1, \ldots, x_n$ не рассматриваются, и выполнение приема обрывается. Однако, в отличие от предыдущих случаев, здесь не происходит выдачи отказа - просто проверку в таких случаях целесообразно проводить другими средствами. Чтобы ввести данный режим, используется указатель "откат(i)", где i - номер того антецедента, после идентификации которого осуществляется фиксация определенных к текущему моменту теоремных переменных.

5. В определенных ситуациях возможно быстрое усмотрение заведомой ложности проверяемого утверждения. Тогда, для ускорения выхода из бесперспективной проверки, используются специальные приемы проверочного оператора, имеющие указатель "не". Консеквент теоремы такого приема имеет вид отрицания утверждения, проверяемого рассматриваемым проверочным оператором. Если удается установить истинность антецедентов теоремы и идентифицировать проверяемое утверждение с отрицанием консеквента, то сразу выдается отказ.

6. Можно обеспечить выход из проверочного оператора с выдачей отказа, если не удалось установить (также с помощью проверочных операторов) истинность антецедентов, номера которых перечислены в указателе "сброс($i_1 \ldots i_m$)".

Анализаторы

Работа анализатора заключается в сканировании утверждений, занесенных в его буфер, а также посылок текущей задачи на исследование, и занесении в буфер следствий указанных посылок и утверждений буфера. Возможно преобразование специальными приемами ранее занесенных в буфер утверждений. В процессе вывода некоторые заносимые в буфер утверждения могут снабжаться комментарием "внешвывод" (вводимым с помощью стандартных указателей); по окончании работы анализатора такие утверждения будут перенесены в список посылок текущей задачи на исследование - собственно они и составляют результат применения анализатора.

Если прием анализатора выводит настолько ценное следствие, что целесообразно безотлагательное перенесение его во внешнюю задачу и возвращение к сканированию этой задачи, то используется указатель "обрыв(A)", где A - фильтр, уточняющий условие обрыва работы анализатора. В случае обрыва сформированное приемом утверждение автоматически снабжается комментарием "внешвывод" и таким образом переносится во внешнюю задачу.

Если необходимо преобразовать ранее занесенное в буфер утверждение, то используются приемы, теорема которых представляет собой эквивалентность. Левая часть этой эквивалентности идентифицируется с преобразуемым утверждением, а правая - определяет новый вид утверждения. Такие приемы снабжаются указателем "внутрпреобр".

11.4.13 Указатели, определяющие отбор и сохранение ссылок на задачи, рассмотрение которых представляет интерес для развития приема

Для усиления и оптимизации приема может оказаться полезным накопление списков ссылок на задачи, при решении которых он сработал либо, наоборот, был заблокирован тем или иным из своих фильтров. Имеется общий механизм, обеспечивающий сохранение в архиве всех ссылок на задачи, при решении которых происходило срабатывание приема. Такие данные автоматически обновляются при каждом запуске серийного решения всех задач выбранного раздела задачника, и использование их не предполагает введения в приемы каких-либо специальных указателей. Предусмотрена, однако, возможность сохранения ссылок на избранные задачи, в которых сработал прием, а также ссылок на задачи, в которых он не сработал из-за специально указанных причин.

Если требуется начать накопление закрепленного за приемом списка ссылок на задачи, в которых он сработал, причем в момент срабатывания выполнялось дополнительное условие, определяемое фильтром A, то вводится указатель "задачи(i A)". Здесь i - произвольно определяемый номер установки на отбор. Таких установок в приеме может вводиться несколько, и для каждой будет создаваться свой список ссылок.

Если требуется создать список задач, в которых прием не сработал из-за того, что срабатывание было заблокировано фильтром A, то этот фильтр помещается в описании приема внутри записи "задачи(i A)", где i - снова номер установки на отбор задач.

Для просмотра списка отобранных задач нужно выделить многоцветной указкой вхождение рассматриваемого указателя либо фильтра "задачи(...)" и нажать правую клавишу мыши либо клавишу "з".

11.4.14 Указатели, определяющие формирование информации для трассировки

Ввод новых элементов чертежа при дополнительных построениях

При решении геометрических задач, для которых изначально был создан чертеж, возможно пополнение этого чертежа в процессе дополнительных построений. Для такого пополнения используются перечисляемые ниже указатели.

1. Чтобы ввести на чертеже точку A - основание перпендикуляра, опущенного из точки B на прямую, проходящую через точки C и D, служит указатель "перпендикулярно(A B C D)". Заметим, что сам перпендикуляр при этом не прорисовывается - для его прорисовки необходимо добавить приводимый ниже указатель "отрезок(A B)".

2. Чтобы провести отрезок, соединяющий точки A и B, используется указатель "отрезок(A B)".

3. Чтобы ввести на чертеже точку A пересечения прямых, проходящих через точки B, C и D, E соответственно, используется указатель "точкапрямой(A B C D E)".

4. Чтобы выбрать произвольным образом точку A, лежащую на отрезке с концами B и C, используется указатель "точкаотрезка(A B C)".

5. Чтобы ввести точку A внутреннего касания окружности с центром B, проходящей через точку C, и окружности с центром D, проходящей через точку E, используется указатель "внутркасаются(A B C D E)". Аналогичным образом, в случае внешнего касания указанных окружностей применяется указатель "внешкасаются(A B C D E)".

6. Чтобы ввести точку A пересечения прямой, проходящей через точки B, C, с прямой, проведенной из точки D параллельно прямой, проходящей через точки E, F, используется указатель "параллельны(A B C D E F)".

7. Чтобы нарисовать окружность с центром в точке A, проходящую через точку B, используется указатель "окружность(A B)".

8. Чтобы ввести на чертеже центр A окружности, описанной около треугольника с вершинами в точках B, C, D, и прорисовать саму окружность, используется указатель "описана(A B C D)".

9. Чтобы отложить на отрезке с концами B, C точку A, отстоящую от B на расстояние, равное расстоянию между точками D, E, используется указатель "расстояние(A B C D E)".

10. Чтобы ввести на чертеже точку A пересечения перпендикуляра к прямой, проходящей через точки B и C, восставленного в точке D, с прямой, проходящей через точки E и F, используется указатель "перпендикулярны(A B C D E F)".

11. Чтобы ввести точку A на луче BC, расстояние которой до точки B, выраженное в пикселях, равно символьному числу r, используется указатель "луч(A B C r)". Явное указание расстояния означает, что выбор на луче точки A в некотором смысле произволен (например, вводится конец координатного вектора), а константа r подбирается из соображений достаточной различимости точек на чертеже.

12. Тесно примыкает к предыдущему указатель "прямкоорд(A B C D r)", вводящий точку A на перпендикуляре к прямой BC, восставленном в точке B, лежащую от прямой BC по ту же сторону, что и точка D; расстояние от A до B, выраженное в пикселях, равно символьному числу r.

Ввод контрольного прерывания перед применением приема

Чтобы при отладке проще было отслеживать моменты срабатывания приема, можно использовать указатель "стоп". Его наличие обеспечивает выход в отладчик ЛОСа перед моментом выполнения преобразований. Разумеется, после отладки этот указатель следует удалить.

Шаблоны и указатели текстформульного сопровождения приема

Если в описании приема не предусмотреть специальных средств для вывода на экран пояснений при его срабатывании, то по умолчанию будет выводиться текст того концевого пункта оглавления базы приемов, в котором размещен этот прием. Во многих

случаях это, в сочетании с возможностью просмотра описания приема на ГЕНОЛОГе, достаточно наглядно объясняет смысл выполняемых действий. Чтобы получить более качественное сопровождение пошаговых действий решателя, предусмотрена возможность создания в описании приема специальных текстформульных шаблонов выводимых на экран пояснений.

Собственно текстовая часть шаблона выводится при просмотре описания приема на экран при нажатии клавиши "6". Эта часть распадается на несколько независимых фрагментов, каждый из которых начинается с символа ¬, вслед за которым идет номер фрагмента (последовательность цифр, завершающаяся пробелом). Нумерация начинается с 1; если фрагмент всего один, то запись ¬1 отбрасывается. Для указания мест, на которые должны вставляться формулы, используются пары расположенных подряд скобок (). Если фрагмент единственный и запись ¬1 отброшена, то при вводе текста следует отступить от левого края на одну позицию, иначе первая буква пропадет. Напомним, что для ввода символа ¬ текстовым редактором служит клавиша "Ctr-n".

Чтобы указать, какие именно формулы следует вставлять, и какие именно фрагменты в каких случаях использовать, служат указатели "титр(P)", размещаемые вместе с прочими указателями приема. Здесь P - выражение, построенное при помощи условных выражений "вариант(...)" из атомарных указателей текстформульного сопровождения. Каждый такой атомарный указатель имеет вид "набор($A\,B_1 \ldots B_n$)", где A - символьный номер используемого фрагмента текстового шаблона, связанного с приемом; $B_1, \ldots, B_n$ - информация о заполнении текстового шаблона. Если в некотором подслучае нужно отменить отображение срабатывания приема, то используется указатель "стоп". Условная конструкция "вариант(...)" может отсутствовать, и тогда во всех случаях пояснение будет строиться по одному и тому же шаблону; в противном случае для разных ситуаций будут извлекаться различные шаблоны.

Информационные элементы B_i, уточняющие использование текстового шаблона для создания текстформульного сопровождения срабатывания приема, бывают следующих двух типов:

1. терм($C_1 \ldots C_n$) - $C_1, \ldots, C_n$ суть операторные выражения, определяющие термы, подставляемые последовательно в текстовый шаблон вместо пар скобок (). Для указания на теоремный терм T, не подлежащий обработке нормализаторами, внутри C_i можно использовать запись "фикс(T)".

2. параметры($X_1 \ldots X_m\ D$) - $X_1, \ldots, X_m$ суть теоремные переменные, для которых при подстановке в термы $C_1, \ldots, C_n$ информационного элемента "терм($C_1 \ldots C_n$)" вводятся новые переменные, обозначающие объекты типа D (при отсутствии D берутся первые неиспользуемые большие буквы).

Чтобы сопровождать пояснениями обращения к нормализаторам и другим пакетным операторам, эти обращения сопровождаются комментариями "титр(P)", аналогичными приведенному выше указателю "титр(P)" (например, при обращении к нормализатору используется запись "замечание(титр(P))".

Для боле гибкого учета различных ситуаций можно использовать в одном и том же описании приема несколько указателей "титр(...)", вводящих располагаемые подряд независимо определяемые текстформульные фрагменты.

11.5 Нормализаторы приема

При применении приема формируются различные термы - как вспомогательные (например, необходимые для принятия решения или представляющие собой промежуточные результаты вычислений), так и основные (входящие в новую посылку или в новое условие, либо в заменяющий терм при тождественной или эквивалентной замене, и т.п.). Любой из таких термов может быть преобразован цепочкой обращений к пакетным нормализаторам либо вспомогательным задачам. Эта цепочка обращений определяется специальной группой служебных термов ГЕНОЛОГа, называемых нормализаторами приема. Каждая группа нормализаторов связана с обработкой определенного терма A, который выделяется либо непосредственно в теореме приема, либо в некотором фильтре приема (в том числе в фильтре, встречающемся внутри указателя). Заметим, что обработке заданной группой нормализаторов подлежат все термы, равные A, независимо от их вхождения в теорему или в фильтры - нет необходимости связывать ее с каждым из таких вхождений A (для выделения вхождений терма A, не подлежащих нормализации, используется, как отмечалось выше, указатель "копия"). Собственно для выделения A служит специальный интерфейс: сначала в теореме или в фильтре многоцветной указкой выбирается какое-либо вхождение A; нажимается Enter, и в возникающем при этом окне текстового редактора набирается группа нормализаторов, относящихся к A.

Каждый нормализатор приема представляет собой либо обращение к пакетному нормализатору, либо обращение к задаче, либо указатель коррекции посылок, относящийся сразу ко всей цепочке преобразований (можно вводить либо удалять заданные посылки, в предположении истинности которых будут проводиться преобразования). Преобразования выполняются в соответствии с указанным порядком, слева направо. Разделители между нормализаторами - пробелы.

11.5.1 Обращения к пакетным нормализаторам

Каждое обращение к пакетному нормализатору либо состоит из названия A этого нормализатора, либо имеет вид $A(B_1 \dots B_n)$. $B_1, \dots, B_n$ - специальные термы, называемые указателями обращения к нормализатору. Возможные типы таких термов приводятся ниже:

1. "замечание$(A_1 \dots A_n)$" - вводится, если его не было, комментарий $(A_1 \dots A_n)$. Перед A_1 возможно размещение терма "условие(B)", уточняющего условие, при котором вводится комментарий. В этом случае после A_n можно разместить терм "альтернатива$(C_1 \dots C_m)$", указывающий комментарий $(C_1 \dots C_m)$, вводимый при невыполнении условия B. Формат, в котором здесь задаются комментарии - тот же, что и для указателя приема "замечание(...)". Кроме того, можно явно задавать комментарий посредством выражения "префикс(...)"; в этом случае указатель обращения имеет вид "замечание(префикс(...))". Заметим, что по умолчанию все комментарии текущего пакетного оператора передаются нормализаторам, используемым в приеме этого пакетного оператора.

2. "примечание$(A_1 \dots A_n)$" - вводится (даже если такой комментарий уже был) новый экземпляр комментария $(A_1 \dots A_n)$. В остальном аналогично указателю "замечание(...)".

3. "удалениезамечания($A_1 \dots A_n$)" - удаляется комментарий ($A_1 \dots A_n$). Перед A_1 возможно размещение терма "условие(B)", уточняющего условие, при котором происходит удаление.

4. "исключкоммент($A_1 \dots A_n$)" - при обращении к нормализатору отбрасываются все комментарии с заголовками $A_1, \dots A_n$

5. "условие(P)" - P есть фильтр, накладывающий дополнительные ограничения на применение данного пакетного нормализатора A.

6. "вариант(A_1 A_2)" - при выполнении условия A_1 берется альтернативный пакетный нормализатор, имеющий заголовок A_2.

7. "комментарии($A_1 \dots A_n$)" - при обращении нормализатору передаются все комментарии текущей задачи, имеющие заголовки $A_1, \dots, A_n$.

8. "лимит(N)" - при превышении N шагов обращение к нормализатору обрывается (предпринимается откат к установленному перед обращением оператору "лимит(...)"). Если обращение к нормализатору не превысило N шагов, то после него установленный лимит сбрасывается. Этот указатель не отменяет слишком долгого процесса нормализации, передавая далее ненормализованный терм, а просто блокирует в этом случае дальнейшее применение приема.

9. "расширениепосылок" - к списку посылок нормализатора присоединяются следствия данного списка, введенные ранее использованными в приеме нормализаторами. Такое пополнение списка посылок позволяет нормализатору использовать утверждения, введенные предшествующими нормализаторами для сопровождения по о.д.з. измененных выражений (эти утверждения извлекаются из сквозного накопителя "коррекцияпосылок ...", заполняемого нормализаторами приема).

11.5.2 Обращения к вспомогательным задачам

Нормализатор приема, обеспечивающий обращение к вспомогательной задаче для преобразования нормализуемого терма, имеет вид "задача(A $B_1 \dots B_n$)", где A - уровень обращения к вспомогательной задаче; $B_1, \dots, B_n$ - указатели на формирование задачи. Те B_i, которые суть логические символы (кроме указываемых далее служебных слов), определяют цели новой задачи. Прочие типы указателей перечисляются ниже.

1. По умолчанию, вводится задача на преобразование. Чтобы ввести задачу на описание, используется указатель "тип(описать)".

2. Чтобы перенести в новую задачу все цели текущей задачи, имеющие заголовки $A_1, \dots, A_n$, используется указатель "цель(повторение($A_1 \dots A_n$))".

3. Если логический символ A_2 должен быть введен в качестве цели новой задачи лишь при выполнении фильтра A_1, то используется указатель "цель(условие(A_1 A_2))".

4. Чтобы ввести цель (неизвестные $x_1 \dots x_n$), в которой перечисляются все переменные $x_1, \dots, x_n$, идентифицированные с переменными $X_1, \dots, X_m$ либо встречающиеся в идентифицированных с ними списках переменных, служит указатель "цель(неизвестная($X_1 \dots X_m$))". Другую возможность ввести цель (неизвестные $x_1 \dots x_n$) дает указатель "цель(неизвестные($X_1 \dots X_m$))". Здесь $x_1, \dots, x_n$ суть все неизвестные текущей задачи на описание, являющиеся параметрами подтермов, идентифицированных с переменными $X_1, \dots X_m$.

5. Аналогичным образом, чтобы ввести цель (параметры $x_1 \dots x_n$), в которой перечисляются все переменные $x_1, \dots, x_n$, идентифицированные с переменными $X_1, \dots, X_m$ либо встречающиеся в идентифицированных с ними списках переменных, используется указатель "цель(параметры($X_1 \dots X_m$))".

6. Чтобы была введена цель ($A_1 \dots A_n$), используется указатель "цель(набор($A_1 \dots A_n$))". Формат описания цели здесь такой же, как при описании комментариев.

7. Для передачи вспомогательной задаче списка комментариев A текущего пакетного нормализатора, к которому относится реализуемый прием, используется указатель "цель(нормализация)". Он вводит в формируемую задачу цель (нормализация A).

8. Иногда бывает нужно разрешить нормализуемое утверждение относительно заданного списка выражений $t_1, \dots, t_n$, которые были идентифицированы с теоремными переменными $x_1, \dots x_n$. Для этого используется указатель цели вспомогательной задачи на описание "цель(вспомописание($x_1 \dots x_n$))". Условие этой задачи получается из нормализуемого утверждения заменой в нем выражений $t_1, \dots, t_n$ на вспомогательные неизвестные $z_1, \dots, z_n$, относительно которых и решается задача. В найденный ответ вместо переменных $z_1, \dots, z_n$ подствляются выражения $t_1, \dots, t_n$, и таким образом получается результат нормализации.

9. Для передачи формируемой задаче комментария ($A_1 \dots A_n$) используется указатель "комментарий(условие(P)$A_1 \dots A_n$)". Здесь P - фильтр, определяющий условие ввода комментария; подтерм "условие(P)" может отсутствовать. Комментарий задается в стандартном формате - как в указателе приема "замечание($A_1 \dots A_n$)".

10. Указатель "лимит(N)" определяет число шагов N, по превышении которого выдается отказ на задачу. Как и в случае пакетного нормализатора, такой отказ означает блокировку дальнейшего выполнения приема и откат.

11. Если происходит обращение к вспомогательной задаче на преобразование из приема пакетного нормализатора, то для регистрации в списке посылок нормализатора всех новых посылок, введенных при решении этой задачи, служит указатель "учетодз".

12. Если нужно заблокировать ввод при обращении к вспомогательной задаче комментария "коррекцияпосылок A" - накопителя информации, используемой для коррекции сопровождения по о.д.з. после ее решения, то используется указатель "коррекцияпосылок".

11.5.3 Указатели коррекции посылок

Нормализатор приема "посылки($A_1 \ldots A_n$)" используется для указания на то, что все обращения к пакетным операторам и вспомогательным задачам, определяемые данной цепочкой нормализаторов приема, выполняются при дополнительных посылках $A_1, \ldots, A_n$. Здесь $A_1, \ldots, A_n$ - утверждения в теоремных переменных.

Для исключения заданных утверждений из списков посылок всех пакетных операторов и вспомогательных задач, определяемых рассматриваемой цепочкой нормализаторов приема, используется нормализатор приема "удалениепосылок(x A)". Здесь A - фильтр, определяющий условие на исключаемое утверждение, обозначенное в нем переменной x.

Глава 12

Редактор приемов ГЕНОЛО́ГА

12.1 Просмотр описаний приемов

12.1.1 Вход в редактор приемов

Вход в просмотр и редактирование приемов, реализованных на ГЕНОЛОГе, осуществляется через оглавление приемов. Войти в это оглавление можно из главного меню, нажав клавишу "г" или выбрав левой кнопкой мыши окно "Оглавление приемов". Приемы распределены в оглавлении, в целом, по тематическому принципу. Это распределение нужно лишь для удобства поиска приема вручную и самим решателем никак не используется.

В корневом меню оглавления приемов перечисляются основные разделы, в которых происходило обучение. Начинается перечисление с раздела "Логические приемы", в котором собраны реализованные на ГЕНОЛОГе общелогические приемы (подраздел "Общие приемы"), а также приемы вывода теорем и проч. Одним из последних размещается раздел "Словарь", в котором собраны приемы текстового анализатора. Он организован по принципу обычного словаря: подразделы обозначены буквами от "А" до "Я". В зависимости от того, были ли сохранены или сброшены данные о последних изменениях в базе приемов, корневое меню может завершаться разделом "Буфер". В трех его подразделах "Новые приемы", "Измененные приемы" и "Удаленные приемы" указываются имевшие место изменения. Степень проработки различных разделов, представленных в базе приемов, весьма различна. В некоторых случаях проработка лишь начата на простейших единичных примерах, в других - доведена до более-менее устойчивого решения стандартных задач.

После выбора нужного концевого пункта оглавления приемов нажимается клавиша "курсор вправо" или Enter, и на экране появляется запись приема, относящегося к выбранному пункту.

Для выхода из просмотра приема обратно в оглавление базы приемов нажимается клавиша "курсор влево" либо Esc.

К одному и тому же концевому пункту оглавления могут относиться несколько приемов; смена этих приемов осуществляется клавишами "курсор вверх" - "курсор вниз".

12.1.2 Отображение приема на экране

Описание приема размещается в четырех окнах, отделенных друг от друга горизонтальными линиями; слева расположены номера этих окон. В первом окне находится теорема приема; во втором - заголовок приема; в третьем - список фильтров приема, и в четвертом - список указателей приема. Обращения к нормализаторам выводятся на экран с помощью описываемого ниже специального интерфейса и при общем просмотре не видны. Если одно из окон пусто (таким может быть либо третье, либо четвертое окно), то номер его не прорисовывается, а верхняя и нижняя горизонтальные линии прорисовываются на расстоянии двух-трех пикселей друг от друга. В исключительных случаях могут оказаться пустыми оба (третье и четвертое) окна - таких случаев при создании новых приемов следует избегать, так как здесь описание приема (та его часть, которая остается за вычетом теоремы приема и его заголовка) может оказаться пустым, и прием не будет сохранен в файлах.

В верхней части первого окна может быть размещен чертеж. Это делается лишь для большей наглядности теоремы приема; компилятором ГЕНОЛОГа чертеж не используется.

Если описание приема слишком велико, то содержимое третьего и (или) четвертого окон разрезается на страницы. При просмотре будет выдаваться лишь одна из таких страниц, причем интерфейс позволяет перелистывать страницы двух окон независимо друг от друга. Признаком наличия предыдущей страницы служит запятая в начале текста окна; признаком наличия следующей страницы - запятая в конце окна.

Кроме того, для работы с большими описаниями приемов можно временно убирать с экрана одно или несколько окон. i - е окно убирается и восстанавливается на экране при нажатии клавиши Ctr-Fi.

Иногда текстовый (в скобочной записи термов) вид теоремы приема занимает меньше места, чем стандартный, формульный. Тогда для уменьшения размеров описания приема можно перейти к текстовому режиму отображения теоремы. Это делается путем нажатия клавиши "т". Для восстановления формульного режима нажимается "ф".

Если для каких-либо логических символов или сочетаний таких символов не предусмотрен режим отображения формульным редактором, то теорема приема прорисовывается текстовым редакторов в скобочной записи.

12.1.3 Просмотр компонент описания приема

Применение цветовой указки для выделения элементов описания приема

Обычно все элементы описания приема размещены на экране одновременно; это упрощает восприятие приема "в целом", однако при чтении сколь-нибудь громоздких фрагментов описания бывает удобно применять многоцветную указку, аналогичную той, которая применялась при чтении программ ЛОСа. Эта же указка позволяет получить справочную информацию относительно тех или иных терминов, использованных в описании приема.

Чтобы войти в режим применения многоцветной указки, следует прежде всего выделить то окно, в котором предполагается ее использовать. Для этого существует два

способа - либо нажать клавишу "курсор вправо", и тогда будет выбрано окно номер 1 (номер выбранного окна перекрашивается в малиновый цвет), либо сразу нажать клавишу i для номера требуемого окна; i - 1,2,3 либо 4. Смену номера выбранного окна можно выполнять клавишами "курсор вверх - курсор вниз". После того, как нужный номер выделен, нажатие клавиши "курсор вправо" переводит собственно в просмотр элементов окна с применением цветовой указки. Для третьего и четвертого окон такой просмотр совершенно аналогичен применявшемуся в редакторе программ ЛОСа: смена терма выполняется клавишами "курсор вправо - курсор влево"; вход в первый операнд текущей операции - "курсор вниз"; выход обратно - "курсор вверх". По достижении "корневого" уровня выход из просмотра текущего окна и возвращение к режиму выбора номера окна - тоже клавишей "курсор вверх". Во втором окне многоцветная указка включается лишь тогда, когда заголовок приема более чем односимвольный.

Просмотр теоремы приема, записанной в текстовом (скобочном) виде, происходит так же, как просмотр термов третьего и четвертого окон. Если же теорема изображена в формульном виде, то для выделения ее подтермов используется одноцветный режим: выделенная часть перекрашивается в малиновый цвет. В остальном принципы использования клавиатуры - те же, что для многоцветной указки: вход в первый операнд операции - "курсор вниз"; смена операнда - "курсор влево - курсор вправо"; выход в надоперанд - "курсор вверх".

Для быстрого выхода из режима цветовой указки используется клавиша "пробел" - нажатие ее в любом контексте использования цветовой указки сразу возвращает в режим общего просмотра приема.

Для быстрого выделения конкретного элемента описания приема, расположенного в третьем либо четвертом окнах, можно также использовать мышь: если ее курсор разместить на представляющем интерес элементе и нажать левую клавишу, то он будет выделен многоцветной указкой. Этот же режим можно применять и для первого окна. Если теорема записана в формульном виде, то мышью бывает трудно выделить нужный подтерм для инфиксных операций - вместо него выделяется какой-либо подфрагмент, например, переменная. В таких случаях можно сначала выделить мышью что-то близкое к нужному подтерму, а затем скорректировать выбор подфрагмента клавишами курсора.

Нажатие левой клавиши мыши вне элементов описания приема, как и нажатие пробела, возвращает к режиму общего просмотра приема.

Если третье либо четвертое окно разрезано на несколько страниц, то для перехода к следующей или предыдущей его странице нужно войти в режим цветового выделения элементов этого окна и использовать клавиши "PageDown", "PageUp". Можно также переместить текущий указатель к запятой, расположенной в конце либо в начале окна, и нажать клавишу "курсор вправо" либо, соответственно, "курсор влево". Смену страницы можно выполнить, не входя в просмотр элементов окна - для этого нужно подвести курсор мыши к соответствующей запятой и нажать левую кнопку мыши.

Смена формульного и текстового режимов просмотра теоремы

Для просмотра теоремы приема предусмотрены два режима - обычный формульный и технический текстовый, при котором отображается используемая программами

ЛОСа скобочная запись. Чтобы ввести формульный режим, следует нажать клавишу "ф"; чтобы ввести текстовый режим - нажать клавишу "т". Выбранный режим сохраняется на все время пребывания в оглавлении приемов и сбрасывается при выходе из него; по умолчанию при входе в оглавление устанавливается формульный режим.

Просмотр нормализаторов приема

Для просмотра нормализаторов приема нажимается клавиша "5". После этого под горизонтальной линией, завершающей четвертое оконо приема, возникает набор нормализаторов приема, обрабатывающих некоторый его терм A. Одновременно терм A выделяется цветовой указкой в теореме приема либо, если он встречался не в теореме, а в фильтрах либо указателях, в соответствующих текстах третьего либо четвертого окон. Чтобы перейти к просмотру цепочки нормализаторов, связанных со следующим термом A, нажимается клавиша "PageDown"; чтобы вернуться обратно - клавиша "PageUp". Выйти из данного режима просмотра цепочек нормализаторов, связанных с различными термами A, можно только одним способом - нажатием клавиши "End".

Если требуется внести изменения в текущую цепочку нормализаторов, нажимается клавиша "н" - тогда в начале записи нормализаторов возникает курсор текстового редактора. После завершения редактирования (нажатие Enter либо, в случае отказа от изменения записи, - Esc) - восстанавливается указанный выше режим просмотра. Вносимые здесь и в других случаях редактирования фрагментов описания приема изменения не затрагивают хранящейся в файлах записи приема; они лишь изменяют копию этой записи, используемую для текущего отображения на экране. Для фактического изменения приема нужно после возвращения в общий режим просмотра нажать F3 либо F4. Другая возможность изменять нормализаторы приема (по существу, основная) будет описана ниже - в разделах, относящихся к редактированию приема.

Если при просмотре нормализаторов приема нажать клавишу Ctr-Del, то произойдет удаление текущей цепочки нормализаторов - как и выше, без изменения записи приема в файлах до тех пор, пока не будет нажата F3 либо F4.

Просмотр текстформульного шаблона, используемого для формирования поясняющего срабатывание приема текста

Для входа в просмотр и редактирование текстформульного шаблона, используемого при отображении срабатывания приема в протоколе решения, нажимается клавиша "6". После этого в области, расположенной под нижней горизонтальной линий четвертого окна, отображается текст ранее введенного текстформульного шаблона (если он имелся) и включается обычный режим текстформульного редактора.

12.1.4 Получение справочной информации при просмотре приема

Чтобы получить справочную информацию о том или ином логическом символе, встречающемся в третьем либо четвертом окне описания приема, достаточно выделить этот символ цветовой указкой (с помощью клавиатуры либо поместив на

него курсор мыши и нажав левую клавишу) и далее нажать правую клавишу мыши. Вместо последнего действия можно нажать: в случае третьего окна - Ctr-y (здесь y - кириллица),а в случае четвертого окна - Ctr-x (x - кир.). Справочный текст появится под нижней линией четвертого окна; если для него не хватило места, то следует убрать одно или несколько неиспользуемых окон с помощью Ctr-Fi; i = 1,2,3,4 - как указано выше.

Иногда для одного и того же логического символа имеется несколько различных справочных текстов, соответствующих различным случаям его использования. Эти тексты могут быть двух типов - пояснения к фильтрам и пояснения к указателям приема. Переходить от одного такого текста к другому внутри текстов одного типа, после выдачи на экран некоторого из них, можно с помощью клавиш "курсор вверх - курсор вниз". Переход к текстам следующего типа - нажатием любой другой клавиши (например, "пробел") либо кнопкой мыши. При использовании клавиш Ctr-y, Ctr-x такого рода неоднозначность уменьшается, так как в первом случае выдаются только тексты, связанные с фильтрами приема, а во втором случае - только тексты, связанные с указателями.

Справочный текст убирается нажатием любой не используемой при его просмотре клавиши (например, клавиши "пробел") либо нажатием клавиши мыши вне окон приема.

Чтобы получить во время создания нового приема необходимую информацию о конструкциях, используемых в ГЕНОЛОГе, нужно выйти в исходную ситуацию просмотра приема либо нажать клавишу F1 (тогда появится оглавление общего описания системы), либо нажать одну из клавиш Ctr-з (оглавление заголовков приемов), Ctr-y (оглавление фильтров приемов), Ctr-x (оглавление указателей приемов).

Чтобы при просмотре приема получить доступ к информационным текстам, связанным с заданным логическим символом, нужно нажать клавишу "c". Тогда появляется курсор текстового редактора, с помощью которого набирается название данного символа. После набора символа нажимается Enter, и на экране появляется первая страница указанных информационных текстов. Эти тексты - те же самые, что и просматриваемые из редактора программ ЛОСа. При нажатии любой не используемой в интерфейсе просмотра информационных текстов клавиши - возвращение в просмотр приема.

Начато создание сводного оглавления логического языка решателя. С его помощью можно будет получать информацию о логических символах, встречающихся в теоремах приемов. Вызов оглавления из просмотра приема осуществляется нажатием клавиши "Ctr-я". Если выделить подфрагмент теоремы приема в режиме формульного просмотра и нажать правую клавишу мыши, то внизу появится справочная информация относительно заголовка подфрагмента, при условии, что она уже имеется в данном оглавлении.

С приемом можно связывать набираемые текстовым редактором примечания, облегчающие регулировку системы приемов "вручную". Для входа в просмотр таких примечаний следует нажать клавишу "э". Если в примечания нужно внести изменения, то после этого нажимается Enter (возникает курсор текстового редактора), Нажатие Enter по окончании редактирования приводит к немедленному сохранению изменений в примечаниях.

12.1.5 Указатели на степень готовности приема

Если прием только что введен либо изменен, но изменения не сохранены в файлах, хранящих описания приемов, то нижняя линия четвертого окна имеет голубой цвет. В этой ситуации нельзя выходить куда-либо из просмотра приема - все изменения (а в случае нового приема и сам прием) будут утеряны.

Если прием (новый либо после изменения старого) сохранен в файлах, хранящих описания приемов, но не откомпилирован в программу ЛОСа, то нижняя линия четвертого окна имеет красный цвет. В этом случае можно выходить из редактора приемов в другие разделы и выключать систему, не теряя описания приема.

Наконец, если описание приема сохранено в файлах и прием оттранслирован в программу ЛОСа, то нижняя линия четвертого окна имеет черный цвет. Заметим, что если изменения в описание были внесены для уже откомпилированного приема, то при сохранении изменений в файлах без перекомпиляции приема (это достигается нажатием клавиши F4) нижняя линия четвертого окна останется черной, хотя фактически программа приема на ЛОСе не будет соответствовать новой версии описания приема. Чтобы не создавать такого рода рассогласований, нужно либо использовать при изменениях приема клавишу F3 сохранения с перекомпиляцией, либо изменять отключенный прием с удаленной ЛОСовской программой (она удаляется нажатием клавиши F7).

12.1.6 Переход от просмотра описания приема к просмотру других разделов системы

От описания откомпилированного приема можно перейти к просмотру его программы на ЛОСе, нажав клавишу Home. Далее можно произвольным образом перемещаться по фрагментам ЛОСовской программы (не выходя в главное меню системы) - из любой текущей позиции для возвращения в просмотр приема достаточно нажать клавишу End.

Для быстрого возвращения в главное меню системы из просмотра описания приема предусмотрена клавиша End. При этом в оглавлении базы приемов будет сохранен путь, ведущий к просматриваемому приему, и при возвращении в это оглавление будет выделен синим цветом содержащий данный прием концевой пункт. Если в концевом пункте было несколько приемов, то при входе в просмотр данного пункта будет выдан прием, из которого в последний раз был предпринят выход по End.

Если несколько приемов созданы на основе одного и того же представления теоремы в файлах (такое представление называется узлом теоремы в базе приемов), то для перехода от одного из них к другим служат клавиши PageDown, PageUp. Впрочем, в решателе крайне редко один и тот же узел теоремы обслуживает несколько различных приемов; даже если теоремы двух приемов совпадают, они, как правило, имеют различные узлы.

Возможен прямой переход от просмотра приема к оглавлению программ либо к оглавлению задачника. В первом случае нажимается Shift-1; во втором случае Shift-3.

Для возвращения в данный прием из просмотра другого приема предусмотрен буфер перехода. Чтобы занести в него ссылку на прием, достаточно при просмотре приема

нажать клавишу "минус". Для возвращения к приему из просмотра другого приема нажимается клавиша "равно". Одновременно последний прием сам заносится в буфер перехода, так что еще одно нажатие "равно" восстановит ситуацию.

Предусмотрена возможность связывать прием с той теоремой из базы теорем, на основе которой он создан. Это можно делать не для всех приемов - некоторые из них основаны на псевдотеоремах, не имеющих ничего общего с теоремами предметной области и представляющих собой чисто технические записи. Однако, большинство приемов действительно имеют своим источником конкретную "обычную" теорему. В этих случаях теорема приема возникает из соответствующей ей теоремы предметной области несложными преобразованиями, в настоящей версии системы автоматизированными лишь частично. Имеется интерфейс для установления соответствия между теоремой приема и ее прототипом в базе теорем вручную. Установление такого соответствия является одним из первых шагов в расшифровке процедур автоматического синтеза приемов, дающим исходный материал для анализа. Чтобы перейти от просмотра приема к просмотру той теоремы в базе теорем, которая служит источником приема, служит клавиша F9. Если источник еще не был выбран, то после нажатия F9 возникает оглавление базы теорем, в котором можно выбрать нужный источник - войти в просмотр теоремы и нажать "И".

12.2 Редактирование приемов

12.2.1 Ввод нового приема

Инициализация приема и ввод его теоремы

Для ввода нового приема следует прежде всего найти в оглавлении базы приемов тот концевой пункт, в котором предполагается зарегистрировать прием, либо создать такой концевой пункт. Затем следует войти в просмотр данного концевого пункта (клавиша "курсор вправо"). Если в пункте уже имелись какие-либо приемы, то на экране будет отображен один из них, и для создания нового приема нужно нажать клавишу Ctr-п. В случае пустого концевого пункта клавиша Ctr-п не нажимается. Далее в обоих случаях начинается заполнение окон нового приема.

Если прием должен сопровождаться чертежем, то прежде всего нажимается клавиша Ctr-ч, которая вводит в геометрический редактор. По окончании набора чертежа нажимается Enter, если теорема будет набираться формульным редактором, и Ctr-Enter, если ее нужно набрать текстовым редактором. Вслед за этим возникнет курсор формульного либо текстового редактора для набора теоремы приема (чертеж на экране сохраняется).

Если чертеж не создается, то для начала ввода теоремы приема следует либо нажать клавишу Ctr-ф (тогда теорема будет вводиться формульным редактором), либо нажать клавишу Ctr-т (тогда теорема будет вводиться в скобочной записи текстовым редактором; здесь "т" - кириллица). По окончании ввода теоремы нажимается Enter - под теоремой прорисовывается горизонтальная линия. Если теорема вводилась текстовым редактором, но возможна ее прорисовка в формульном виде, то эта прорисовка осуществляется вместо скобочной записи. В левом верхнем углу теоремы появляется номер окна - цифра 1.

Если теорема выдана в формульном виде, то непосредственно под ней размещается выделенная голубым цветом строка обозначений переменных - пар вида (x - xn),

где x - обозначение переменной в формульной записи, а xn - обозначение этой же переменной во внутреннем скобочном представлении. Эти обозначения необходимы для ввода фильтров и указателей приема, так как в них используется только скобочное представление термов.

После набора теоремы под ней проводится нижняя горизонтальная линия. Далее возможен ручной ввод описания приема либо автоматическое создание его исходной версии. В первом случае следует нажать Enter - после этого появляются цифра "2" - номер второго окна приема, а также курсор текстового редактора для ввода заголовка приема. Во втором случае, для входа в оглавление типов приемов, нажимается "a" (кир.). Заметим, что пока автоматическое создание приема используется только для приемов стандартных справочников, хотя существующие процедуры могут предлагать какие-то исходные версии приема и в других случаях.

Интерфейс ручного ввода нового приема устроен так, что по окончании ввода каждого из окон 1 - 3 появляется приглашение для ввода следующего окна (в левом углу прорисовывается его номер, и возникает курсор текстового редактора для набора содержимого окна). Этот режим, однако, можно в любой момент прервать, нажав клавишу Esc. Уже набранные к этому моменту окна приема будут сохранены на экране (но пока не в файлах, так что выход из просмотра приема обратно в оглавление либо смена приема клавишами "курсор вверх - куроср вниз" ПРИВЕДУТ К ПОТЕРЕ набранной его части). Для возобновления набора приема далее нужно поступать так же, как при изменении ранее введенного приема - нажать клавишу Ctr-i, где i - номер того окна, в котором предполагается продолжать набор приема; $i = 2, 3, 4$.

Ввод заголовка приема

Заголовок приема набирается текстовым редактором обычным образом. Если для выбора заголовка здесь понадобится справочная информация, то ее можно получить, нажав клавишу Ctr-з. Тогда появляется оглавление типов заголовков приемов. Выбрав нужный концевой пункт в этом оглавлении, следует нажать клавишу "курсор вправо" - текст шаблона заголовка приема будет прорисован во втором окне автоматически. В нижней части экрана появится текстовое пояснение к этому шаблону. После преобразования шаблона заголовка в заголовок нажимается Enter. Автоматически появляется приглашение к набору 3-го окна.

Ввод фильтров и указателей приема

Третье и четвертое окна (соответственно, фильтры и указатели) набираются текстовым редактором. Если при наборе третьего окна нажать клавишу Ctr-у (кир.), то появляется оглавление фильтров; если при наборе четвертого окна нажать клавишу Ctr-х, то появляется оглавление указателей. При выборе нужного концевого пункта одного из этих оглавлений и нажатии на нем клавиши "курсор вправо" произойдет прорисовка шаблона соответствующего фильтра либо указателя начиная с той позиции окна, на которой находился курсор при обращении к оглавлению (содержимое окна при выходе из оглавления восстанавливается). По окончании набора окна нажимается Enter.

Если при наборе третьего либо четвертого окна появляется необходимость ввести указатель вхождения в теорему, то нажимается Shift-1. Вслед за этим в первом окне

включается режим выбора подтерма теоремы цветовой указкой. Выбранный подтерм прорисовывается малиновым цветом в случае формульной прорисовки теоремы либо, при скобочной прорисовке, выделяется группой цветов, как в редакторе программ ЛОСа. После выбора подтерма, расположенного на нужном вхождении, нажимается "ф". Тогда, начиная с той позиции, на которой во время нажатия Shift-1 находился курсор, прорисовывается указатель вхождения "фикс(...)". Если при просмотре подтермов стало ясно, что указатель вхождения не нужен, то нажимается Esc. Следует заметить, что в особых случаях формульный редактор не позволяет непосредственно выходить на цветовое выделение части подтермов теоремы; тогда для автоматического набора указателя вхождения рекомендуется нажатием Enter выйти из редактирования окна, поменять формульный режим отображения теоремы на текстовый, и снова войти в редактирование окна.

По окончании набора предварительной версии приема нажимается клавиша F4. Это приводит к сохранению в файлах набранной части приема. Далее можно начинать цикл коррекций исходной заготовки приема, присоединяя к ней необходимые фильтры, указатели, нормализаторы и т.п.

Ввод нормализаторов приемов

Для ввода нормализаторов приема необходимо выделить цветовой указкой в теореме либо в некотором терме третьего или четвертого окон нормализуемый терм. Если после этого нажать Enter, то под нижней линией четвертого окна появляется курсор текстового редактора. Если для выделенного подтерма уже были введены нормализаторы, то они будут прорисованы начиная с позиции курсора. Далее предпринимается ввод либо изменение нормализаторов и нажимается Enter (в случае отказа от редактирования - Esc). После этого восстанавливается режим цветовой указки, и можно переходить к выбору нового нормализуемого подтерма.

Как и в случае автоматического набора указателя вхождения, следует учитывать, что не во всех случаях возможно корректное выделение подтерма теоремы в режиме формульного редактора. В тех случаях, когда оно не обеспечено, следует переходить к текстовому режиму отображения теоремы.

Группа нормализаторов приема, связанная с обработкой некоторого подтерма, вводится лишь однократно - по произвольному вхождению этого подтерма в теорему, фильтр либо указатель.

Иногда бывает нужно связывать группу нормализаторов приема с термом A, встречающимся в других нормализаторах приема (например, в дополнительных посылках либо в фильтрах, встречающихся в нормализаторах приема). Если такие термы не встречаются в 1,3,4 окнах, то в 4-м окне вводится фиктивный указатель "фикс(A)", и после этого для A указывается цепочка нормализаторов. Заметим, что данный фиктивный указатель при сохранении приема в файлах отбрасывается, и при последующих просмотрах приема (если его не ввести снова) появляться не будет.

Как и для любых изменений приема, после коррекции нормализаторов нужно нажать клавишу F4 для сохранения изменений в файлах либо клавишу F3 для сохранения изменений и перекомпиляции.

12.2.2 Сохранение описания приема и компиляция

По окончании ввода приема следует по крайней мере сохранить его описание в файлах системы (оно сохраняется в 8-м информационном блоке). Для этого нажимается клавиша F4. Если введенная версия приема не требует какой-либо доработки, то вместо F4 можно нажать F3 - тогда кроме сохранения описания приема сразу же будет выполнена компиляция его в программу на ЛОСе.

В целях отладки иногда бывает нужно временно удалять программу приема, сохраняя его описание в файлах. Для этого используется клавиша F7. Чтобы снова откомпилировать программу такого приема, нажимается F5.

Для отладки компилятора ГЕНОЛОГа предусмотрен вход в компиляцию с установкой прерывания через оглавление программ ЛОСа. Здесь используется клавиша Ctr-F5, после нажатия которой возникает некоторый пункт оглавления программ. В этом оглавлении нужно войти в подраздел "Компилятор ГЕНОЛОГа"; выбрать нужный концевой пункт и нажать на нем "курсор вправо". Как только при компиляции будет достигнут соответствующий данному концевому пункту оператор "прием(
N)", будет включен пошаговый режим отладчика ЛОСа.

12.2.3 Изменение приема

Чтобы изменить теорему приема, обычно применяется режим текстового редактора, так как в нем можно скорректировать произвольный подфрагмент теоремы, не переписывая заново всей теоремы. Для входа в такой режим достаточно нажать клавишу Ctr-т (кир.). Разумеется, для работы в текстовом редакторе следует заблаговременно уточнить обозначения теоремных переменных во внутреннем "скобочном" представлении, используя располагаемый под "формульной" записью теоремы выделенный голубым цветом список таких обозначений. По окончании редактирования нажимается Enter, после чего (если только не был специально установлен текстовый режим просмотра) измененная теорема будет перерисована в формульном виде. Как и во всех случаях изменения приема, для сохранения изменений здесь следует нажать F4 либо (для одновременной перекомпиляции) F3.

Для изменения теоремы приема формульным редактором имеется несколько возможностей. Первая из них - войти в набор теоремы "с конца" (нажатие клавиши "Ф" из общего просмотра приема), отбросить с помощью Backspace старую версию концевой части, и затем перенабрать эту часть в требуемом виде. Другая возможность - выделить курсорами либо мышью подтерм теоремы, который нужно изменить, нажать "Ф", ввести формульным редактором новую версию, и нажать "Enter". Если нужно добавить новый антецедент теоремы, то выделяется тот антецедент, перед которым выполняется вставка, нажимается "в", и формульным редактором вводится добавляемый антецедент. Если выделить антецедент и нажать "В", то вставка нового антецедента будет осуществлена непосредственно после выделенного. Если выделить антецедент и нажать "Ctr-Del", то он будет удален. Перечисленные операции можно выполнять не только с антецедентами, но также с операндами любой коммутативно-ассоциативной операции, встречающейся в теореме. Наконец, упомянем совсем радикальный способ изменения теоремы приема - полный перенабор ее формульным редактором. Для этого нажимается клавиша Ctr-ф, вводящая в режим формульного редактора. Для удобства во время набора новой версии старая версия теоремы

сохраняется в верхней части экрана - кроме случаев, когда места для обеих версий сразу недостаточно.

Для коррекции чертежа - как и для ввода нового чертежа - нажимается клавиша Ctr-ч, вводящая в геометрический редактор.

Если нужно изменить содержимое окна с номером 2,3 либо 4, то нажимается, соответственно, клавиша Ctr-2, Ctr-3 либо Ctr-4. По завершении редактирования нажимается Enter.

Для изменения нормализаторов, связанных с некоторым подтермом теоремы, фильтра либо указателя приема, следует поступать так же, как для ввода новых нормализаторов - выделить этот подтерм цветовой указкой и нажать Enter. Для удаления ранее введенных нормализаторов следует войти в просмотр всего их списка (Ctr-5, и далее PageDown - PageUp), выделить ненужную группу нормализаторов и нажать Ctr-Del. Из просмотра списка нормализаторов, инициируемого нажатием клавиши Ctr-5, можно также изменять нормализаторы. Для этого на текущей просматриваемой группе нормализаторов нажимается "н". Наконец, для удаления вообще всех нормализаторов данного приема нажимается Ctr-End. Последняя операция бывает необходима, если была существенно изменена теорема или удалены некоторые фильтры либо указатели, с подтермами которых были связаны нормализаторы (в этих случаях попытка просмотра нормализаторов с помощью Ctr-5 может приводить к сообщениям о некорректном обращении к операторам ЛОСа). Еще раз напоминаем о необходимости использовать F4 либо F3 для сохранения изменений.

12.2.4 Буфер базы приемов

Чтобы можно было контролировать вводимые в приемы изменения, используется буфер базы приемов. Этот буфер создается автоматически; в нем регистрируются новые, измененные и удаленные приемы. Каждый раз при его создании в корневом меню оглавления базы приемов добавляется последним пунктом подменю "Буфер". В этом подменю вводятся далее подменю "Новые приемы", "Измененные приемы" и "Удаленные приемы" (в зависимости от наличия таковых). При добавлении, изменении либо удалении приема в соответствующем (одном из указанных трех) подменю вводится концевой пункт, подзаголовок которого копируется с подзаголовка соответствующего (содержащего либо содержавшего прием) концевого пункта основной части оглавления. Через этот концевой пункт происходит выход в просмотр старой (в случае нового приема - основной) версии приема.

Для перехода от просмотра в буфере старой версии к просмотру новой версии используется клавиша "о"; для возвращения от просмотра основной версии к просмотру старой версии - клавиша "б".

При просмотре версии приема, находящейся в буфере, не рекомендуется вносить какие-либо изменения в описание приема (!) - это может привести к необходимости "ремонта" оглавления базы приемов. Однако, при просмотре версии в буфере возможна ее компиляция либо уничтожение ее программы - как и обычно, с помощью клавиш F5 и F7. Указанный выше ремонт оглавления состоит в том, чтобы через технический просмотр 8-го информационного блока (см. подраздел "Ресурсы и установки"главного меню системы) выйти в корень оглавления базы приемов (метки "оглавление", "прием") и изменить содержимое логического терминала "позиция" на "число(0 0)". После выхода из технического просмотра - вернуться в просмотр

оглавления приемов и сразу же нажать в нем клавишу "О" (кир.) для сброса непригодного далее буфера.

Если необходимо для целей отладки временно откомпилировать старую версию приема, удалив программу новой версии, то из просмотра новой версии приема нажимается клавиша "И". Обратно, для удаления программы старой версии и компиляции программы новой (снова из просмотра новой версии) нажимается клавиша "Т". Наконец, если новая версия не оправдала надежд и нужно вернуться к старой версии, нажимается "Б".

Время от времени буфер следует расчищать: если в нем накопилось слишком много приемов, то будут заблокированы изменение приемов, ввод новых приемов и удаление старых. Расчистка буфера происходит при нажатии в любом месте оглавления базы приемов клавиши "О" (если это происходило в момент просмотра пунктов буфера, то произойдет выход в главное меню, иначе - сохранится просмотр оглавления), Если после расчистки буфера и возвращения из главного меню в оглавление базы приемов экран оказался пустым, то это означает, что пункты оглавления оказались "прокручены вверх". Для их возвращения используется PageUp.

Для удобства работы с буфером предусмотрен автоматический переход в него из любого места оглавления базы приемов - по нажатию клавиши "ф".

12.2.5 Перенесение приема в другой концевой пункт оглавления

Чтобы перенести прием в другой концевой пункт оглавления базы приемов, следует войти в его просмотр, нажать Insert, выйти обратно в оглавление, перейти к нужному концевому пункту и (не входя в него) снова нажать Insert. Перед вторым нажатием Insert следует убедиться, что необходимый концевой пункт действительно выделен в качестве текущего (т.е. имеет голубой цвет).

12.2.6 Удаление приема

Для удаления ненужного приема нажимается клавиша Ctr-Del. Заметим, что в этом случае прием сохраняется в буфере базы приемов и при необходимости может быть оттуда извлечен обычной процедурой перенесения приема из одного пункта оглавления в другой пункт: на переносимом приеме нажимается Insert, далее по оглавлению базы приемов находится нужный концевой пункт и (без входа в этот пункт, но после выделения его голубым цветом) снова нажимается Insert. После расчистки буфера (нажатием клавиши "О") происходит окончательное удаление ранее исключенных приемов.

Особой осторожности требует удаление приемов пакетных нормализаторов. Если удаляется или даже просто перекомпилируется единственный оставшийся прием такого оператора, то происходит нарушение архитектуры его программы, для восстановления которой нужно обязательно (!) удалить клавишей F7 и затем снова откомпилировать прием "окончание" этого нормализатора, играющий роль переключателя уровней срабатывания. Лишь затем можно подключать к программе этого оператора другие приемы - вводя новые или компилируя временно отключенные старые. Если нормализатор имеет более одного приема, то удаление приема и перекомпиляция возможны без ограничений.

12.2.7 Автоматическое пополнение описания приема

Процедуры, созданные для автоматического синтеза приемов на ГЕНОЛОГе, имеют пока предварительный характер, однако даже и в таком виде они могут существенно ускорить процесс ввода элементов описания приема и подсказать полезные добавления к этому описанию.

Если после ввода теоремы нажать "а", то возникает оглавление типов приемов. Фактически это ветвь оглавления программ для подраздела "Справочник ЗАГОЛОВОКПРИЕМА" раздела "Генератор приемов". Переходя в этом оглавлении к концевому пункту (точка после номера пункта) для приемов требуемого типа и заходя в этот пункт, инициируем работу генератора приемов. Если был создан хотя бы один прием, то на экране появится описание первого из таких приемов, ограниченное снизу голубой линией. Это описание, после необходимой коррекции, можно сохранить в базе приемов (F4) или даже сразу откомпилировать (F3). Если было создано несколько приемов, то каждый следующий прием из серии вызывается на экран нажатием клавиши "ш"; его также можно сохранить либо откомпилировать. По исчерпании серии, нажатия клавиши "ш" каких-либо изменений на экране вызывать не будут. Заметим, что данный режим создания приемов пока применяется исключительно для приемов справочников - согласно первом пункту "Справочники и сопровождающие их простейшие приемы" оглавления типов приемов.

Значительно более употребительным (практически постоянно) является автоматическое пополнение списков указателей и нормализаторов приема. Для такого пополнения нажимается клавиша "н". Если процедура предлагает ввести новый элемент либо удалить старый, то он прорисовывается под нижней линией четвертого окна. В случае, если этот элемент является нормализатором приема, одновременно выделяется цветовой указкой нормализуемый им подтерм. Нажатием клавиши Insert можно ввести предлагаемый элемент в описание приема, нажатием клавиши Delete - отказаться от его ввода. Если процедура предлагает добавить новый элемент, то в левой части прорисовываемого вверху экрана меню написано "Новый указатель"; если она предлагает удалить старый элемент, то в левой части прорисовываемого вверху экрана меню написано "Старый указатель". В последнем случае нажатие Delete приведет к удалению элемента, а нажатие Insert - к его сохранению. Предложения выдаются последовательно, по мере нажатия клавиш Insert - Delete. Этот процесс можно в любой момент оборвать, нажав Esc. Введенные до этого изменения сохранятся (но не в файлах, а только в текущем отображении приема).

Заметим, что, как правило, предложения по удалению старых указателей вызваны не какими-то специальными мотивировками, а просто тем, что процедура не генерирует именно эти типы указателей - они находятся вне ее компетенции. Автоматическое создание обращений к нормализаторам общей стандартизации существенно облегчает ручной ввод приемов - ведь такими указателями обычно снабжаются практически все отличные от переменных и констант подвыражения формируемых приемом термов.

12.3 Дополнительные возможности редактора приемов

12.3.1 Разрезание окна на несколько частей либо склейка частей окна

Если третье либо четвертое окно становится слишком большим и затрудняет работу с приемом (например, не оставляет места для просмотра нормализаторов), то его можно разрезать на несколько частей. Для этого нужно войти в просмотр термов окна цветовой указкой, выбрать тот терм, который будет последним в первой половине разрезаемого окна, и нажать клавишу "р" (кир.). Другой способ разрезать окно - войти в режим его редактирования и поместить запятую в точке разрезания.

Чтобы склеить текущую часть окна с непосредственно следующей за ней, нужно войти в просмотр термов окна цветовой указкой и нажать F6. Другой способ - войти в редактирование окна и убрать запятую в его конце.

12.3.2 Копирование приема либо его фрагментов

При программировании на ГЕНОЛОГе иногда возникают серии похожих приемов, для ускоренного ввода которых можно применять копирование первого приема серии с последующими исправлениями.

Для полного копирования приема имеются две возможности - с регистрацией в файлах и компиляцией для вводимой копии, либо без регистрации и компиляции. Во втором случае нажимается клавиша Ctr-д, после чего возникает заготовка для редактирования альтернативной версии приема (старая версия приема сохраняется; под новой версией проведена голубая линия). Эту версию приема после завершения редактирования можно сохранить и откомпилировать обычным образом (F4 либо F3). В первом случае нажимается клавиша "д", и созданная копия приема сразу же сохраняется и компилируется. Эта возможность, в отличие от предыдущей, практически не применяется.

Иногда бывает нужно многократно использовать в различных приемах одни и те же фильтры. Чтобы ускорить их ввод, можно выделить такой фильтр цветовой указкой и нажать клавишу "у". Если затем войти в просмотр другого приема и, не входя в режим цветовой указки, нажать "у" снова, то данный фильтр (а если указанным образом ранее было отобрано несколько фильтров, то все они сразу) будет занесен в третье окно приема. Далее нужно сохранить введенное таким образом изменение, нажав F4 либо F3. Чтобы сбросить накопитель фильтров, вводимых в приемы по нажатии клавиши "у", используется клавиша "У".

Вообще, для перенесения каких-либо фрагментов из приема в прием можно использовать обычный буфер текстового редактора: сначала нужно войти в режим редактирования текстовым редактором того окна, которое содержит копируемый фрагмент (в том числе первого окна); затем занести этот фрагмент в буфер нажатием PageDown в начале и конце фрагмента; перейти в другой прием либо в другое окно того же самого приема, и извлечь фрагмент из буфера нажатием PageUp.

Можно создавать различные приемы, используя одну и ту же теорему. Для этого сначала нужно войти в прием, имеющий данную теорему, и для занесения ее в специальный буфер нажать Ctr-б. Далее, при создании нового приема можно начинать

цикл его ввода с нажатия Ctr-и либо Ctr-к (кир.). В обоих случаях будет прорисована отобранная в буфер теорема. В первом случае прием будет сохранен в файлах на основе уже имеющейся в них записи теоремы - "узла теоремы"; во втором - будет введен новый узел той же теоремы. Вторая возможность используется, если с первым узлом уже был связан прием, имеющий тот же самый заголовок: выход из узла теоремы на структуру данных приема по заголовку этого приема должен происходить однозначным образом.

12.3.3 Изменение логического символа, за которым закреплен прием

Иногда бывает удобно изменить логический символ, за которым первоначально был закреплен прием (например, прием был создан на основе копии аналогичного приема, относившегося к другому понятию). В этом случае нажимается клавиша Ctr-й; вводится текстовым редактором необходимый символ, и нажимается Enter. Произойдет закрепление приема за новым символом, перенесение его описания в файлах на новое место и перекомпиляция.

12.3.4 Поиск приемов заданного вида в базе приемов

Для быстрого просмотра всех приемов базы приемов ГЕНОЛОГа, удовлетворяющих заданному условию, предусмотрен специальный режим, использующий оператор ЛОСа "текприем(...)". Задание на отбор приемов формулируется в виде ЛОСовской программы этого оператора. Все, что нужно для этого сделать - разместить после начального оператора "метка(икс(десять))" этой программы и до последнего ее оператора "ответ(набор(См))" систему условий на отбираемые для просмотра приемы. В описании оператора "текприем(...)" уточняется, какую информацию о текущем приеме несут различные входные переменные оператора "текприем(...)". Основные из них: x3 - теорема приема; x4 - заголовок приема; x5 - описание приема. Описание приема представляет собой набор указателей приема, к которому добавлен терм "условие(и($A_1 \dots A_n$))", где $A_1, \dots, A_n$ - все фильтры приемы, а также добавлены термы "быстрпреобр($B_1 \dots B_m$)", определяющие нормализаторы приема (B_1 определяет нормализуемый подтерм - прямо либо через указатель вхождения "фикс(...)"; $B_2, \dots, B_m$ суть нормализаторы приема, связанные с этим подтермом).

После того, как установлена необходимая программа оператора "текприем(...)", следует выйти из редактора программ ЛОСа и перейти в произвольный раздел оглавления базы приемов. Далее нажатие клавиши F8 запускает цикл просмотра всех (вне зависимости от раздела, где была нажата F8) приемов системы, удовлетворяющих заданному ограничению. Чтобы перейти от текущего приема к просмотру следующего, нажимается клавиша "курсор влево". Чтобы оборвать просмотр, нажимается Esc, переводящее в главное меню. После этого можно вернуться в оглавление базы приемов - текущими окажутся те конечный пункт оглавления и прием этого пункта, на котором был оборван просмотр. Если в процессе просмотра приемы исчерпаны либо их вообще не было, то система выходит в отладчик ЛОСа на оператор "трассировка(стоп 0)" - далее нажатие Esc возвратит ее в главное меню. Заметим, что оператор "текприем(...)" можно запрограммировать так, чтобы он не только отбирал нужные приемы для просмотра, но и вносил в них те или иные изменения, с перекомпиляцией или без нее. Таким образом, появляется возможность с разумными затратами сил

осуществлять при необходимости различные глобальные преобразования всей базы приемов. Отметим также, что изменять просматриваемые в указанном выше цикле приемы можно и обычным образом, вручную. Единственное ограничение здесь - невозможность проконтролировать результаты перекомпиляции путем перехода к редактору программ ЛОСа (клавиша Home). Чтобы изменить найденный прием с возможностью такого контроля, следует оборвать на нем просмотр с помощью Esc, а затем снова зайти в базу приемов и выполнить редактирование приема.

При регулировке решателя те или иные приемы иногда отключаются с помощью клавиши F7. Чтобы выявить все такие отключенные (не откомпилированные) приемы базы приемов, следует нажать клавишу Ctr-F8 и далее действовать, как в указанном выше цикле просмотра (переход к следующему отключенному приему - "курсор влево"; обрыв цикла просмотра - Esc).

12.3.5 Перенесение приемов из версии системы, находящейся в директории ALT

Чтобы переносить приемы ГЕНОЛОГа из одной версии системы в другую, следует разместить ту версию, приемы которой будут копироваться, в поддиректории ALT рабочей директории той версии, в которую нужно скопировать приемы. Если при входе в оглавление ГЕНОЛОГа основной версии нажать Ctr-Tab, то произойдет переход в просмотр оглавления ГЕНОЛОГа для версии, хранящейся в поддиректории ALT. В правом верхнем углу экрана появится выделенный красным индикатор "ALT". Режим работы с альтернативной версией сохранится даже при выходе в главное меню и переходе из него в редактор ЛОСа либо в другое оглавление. Для возвращения в обычный режим нужно в любой из указанных ситуаций повторно нажать Ctr-Tab и убедиться в отстутствии индикатора "ALT".

Находясь в режиме просмотра альтернативной версии, нужно зайти в просмотр того приема ГЕНОЛОГа, который требуется скопировать в основную версию. На этом приеме нажать "Insert", затем выйти в оглавление, нажать Ctr-Tab, выбрать тот концевой пункт оглавления ГЕНОЛОГа, в котором требуется зарегистрировать копию, и нажать "Insert". Прием будет скопирован, но не откомпилирован. Компиляцию приема, после необходимой его коррекции, нужно будет выполнять вручную - нажатием F3.

12.4 Анализ применений приема на обучающем материале

Почти любой новый прием, заносимый в решатель, требует уточнений, выявляемых в процессе прогонки решателя по тем или иным разделам задачника. Для такой прогонки после ввода приема предпринимается выбор необходимого раздела в оглавлении задачника и нажатие Ctr-з в корневом меню данного раздела. По окончании цикла решения задач данного раздела нажимается клавиша "з" и анализируется статистика процессов решения - так, как об этом говорилось выше в разделе "Запуск решения задачи и его пошаговый просмотр" 4-й главы. В первую очередь, здесь выявляются задачи, ответ которых изменился, либо решение которых существенно

замедлилось, либо которые вообще перестали решаться после ввода приема или группы новых приемов. Каждая такая задача далее анализируется отдельно. Если была введена либо изменена группа приемов, то с помощью буфера базы приемов приемы этой группы последовательно отключаются либо восстанавливаются в своем исходном виде. Это позволяет выявить конкретного виновника замедления, ошибки либо отказа. Далее устанавливается прерывание при входе в найденный таким образом прием, и на уровне отладчика ЛОСа анализируется его работа в процессе решения задачи. Для более надежного выявления моментов срабатывания приема рекомендуется перекомпилировать его, введя в четвертое окно указатель "стоп" - тогда выход в отладчик ЛОСа будет происходить непосредственно перед реализацией преобразований приема.

На основе рассмотрения выявленных указанным способом особых случаев срабатывания приемов в группе задач происходит выработка новых его эвристических решающих правил. Чтобы впоследствии можно было быстро находить все те задачи задачника, в которых прием сработал хотя бы однажды (это может понадобиться при проверке того, что усиление фильтров приема не нарушило процессов решения таких задач), информация о данных задачах автоматически сохраняется при запуске очередной прогонки в специальном информационном блоке - архиве задачника. Впоследствии можно получить список таких задач, расположенных в заданном разделе задачника, используя следующую процедуру:

1. Выделить в некотором меню задачника тот пункт, который вводит в требуемый раздел;
2. Нажать клавишу Shift-2 для перехода в оглавление базы приемов;
3. Войти в просмотр нужного приема и нажать клавишу "А" (кир.);
4. Нажать клавишу Shift-3 для перехода в оглавление задачника;
5. Нажать клавишу "ф" для перехода в буфер задачника.

В буфере задачника будет создано новое подменю, заголовок которого копирует текст концевого пункта базы приемов, к которому относился прием. В этом подменю размещаются дубликаты указанных выше задач (регистрируются не более 50 первых задач - в случае большего числа задач нужно по отдельности анализировать подразделы рассматриваемого раздела). Из просмотра дубликата задачи можно перейти к просмотру оригинала нажатием клавиши "б". Однако, все тестовые запуски (в том числе групповой запуск для данного подменю буфера) можно выполнять непосредственно на дубликатах. В действительности здесь дублируются не сами задачи, а только ссылки на них из оглавления. Поэтому данные об измененных результатах решения задачи будут просматриваться одновременно из буфера и из того раздела оглавления, в котором расположена основная версия задачи.

При вводе нового приема можно сокращать прогонку по задачнику, отбирая лишь те задачи, при решении которых возникают логические символы, необходимые для срабатывания этого приема. Для организации такой сокращенной прогонки следует применять следующую процедуру:

1. Войти в просмотр приема и нажать клавишу "С";

2. Нажать Shift-F3 для перехода в оглавление задачника (заметим, что в этот момент в буфере задачника будет находиться пустой подраздел, заголовок которого копирует текст концевого пункта базы приемов, содержащего анализируемый прием; в этом подразделе при прогонке будут накапливаться дубликаты задач, в которых прием сработал хотя бы однажды;

3. Выбрать нужный раздел задачника и нажать "З".

Начнется сокращенный цикл решения задач раздела; по окончании его выполняется обычный анализ статистики.

Для оптимизации решателя предусмотрена возможность определения "холостого хода" приема на заданном разделе задачника - средней трудоемкости попыток применения приема на одно его срабатывание. Чтобы получить такую характеристику, следует запустить цикл решения задач из выбранного раздела задачника не клавишей Ctr-з, а клавишей "а" (кир.). Процесс решения чуть-чуть замедлится, так как для приемов, в попытки применения которых система будет входить при решении задач, начнется накопление пар (суммарная трудоемкость попыток применения приема - число применений приема). Моментом входа в попытку применить прием здесь считается прохождение через оператор "контрольприема(...)", размещаемый компилятором в начале программы приема. По окончании прогонки следует войти в просмотр статистики, где будут указаны случаи наибольшей средней трудоемкости; нажатием клавиши "х" (кир.) можно перейти в режим просмотра соответствующих приемов (переход к каждому следующему приему - нажатием "ш"), упорядоченных по убыванию этой трудоемкости. Чтобы на экран была выдана пара (суммарная трудоемкость - число применений), при просмотре приема следует нажать клавишу "Х"(кир.).

Предусмотрена некоторая автоматизация анализа избыточности применений приема. Чтобы выявить те задачи раздела, где отказ от приема приводит к ускорению решения, следует прежде всего создать в буфере задачника список дубликатов задач этого раздела, в которых прием срабатывает (выбрать раздел задачника; нажать Ctr-2; выбрать прием; нажать "А" (кир.); нажать Ctr-3 для возвращения в задачник). После этого нажатие клавиши "й" инициирует цикл попыток решить отобранные в буфере задачи без данного приема. Трудоемкости решения с приемом и без него сравниваются; если задача быстрее решается без приема, то в оглавлении буфера после ее номера ставится знак вопроса.

12.5 Расширение списка символов, прорисовываемых формульным редактором

Чтобы формульный редактор мог прорисовывать тексты задач в произвольных предметных областях, приходится постоянно пополнять его словарь. Для упрощения понимания обозначений, названия новых логических символов переносятся в формульный редактор без каких-либо изменений. Однако, это происходит не автоматически - сначала нужно сопроводить новое понятие A несколькими приемами справочников формульного редактора. Все эти приемы создаются по одной и той же теореме - "формредактор(A B_1 ... B_n)". Здесь $B_1, \ldots, B_n$ - указатели, уточняющие способ прорисовки. Обязательными являются указатели "префикс" и "логсимвол".

Они означают, что будет использоваться запись $A(t_1 \dots t_m)$, причем заголовок представляет собой текст названия логического символа A. Если операция (отношение) A многоместное, причем операнды отделяются запятыми, то вводится указатель ",". Если A - многоместное, но операнды суть переменные, не разделяемые запятыми, то вместо этого берется указатель "кортеж". Если A есть символ константы, то используется указатель "конст". Наконец, если нужно обеспечить возможность набора символа двмуя нажатиями клавиш, то вводится указатель "ключ(K_1 K_2)". Здесь K_1, K_2 - логические символы, кодирующие выбранные клавиши, причем в латинском регистре. Чтобы при наборе теоремы приема текстовым редактором ввести такие символы, нажимается клавиша F12, переводящая в латинский регистр, затем нажимается F8 (режим ввода названий кодов клавиш), и далее нажимается требуемая клавиша. При переходе к очередному K_i повторное нажатие F12 не требуется, но повторное нажатие F8 необходимо. Для возвращения из латинского регистра в кириллицу, по окончании ввода символов K_i нажимается F11.

Если указатель "ключ(...)" отсутствует, то название A придется вводить формульным редактором через вспомогательное обращение к текстовому редактору (Ctr-Enter). Это не так удобно, как ввод названия двумя касаниями, однако приходится учитывать, что пар клавиш не так уж много, и вводить их лишь для сравнительно часто используемых понятий.

Как только теорема приема введена и нажата клавиша Enter, завершающая работу текстового редактора, нажимается клавиша "а" (кир.). Она переводит в оглавление типов приемов. Здесь следует выбрать первый пункт - "Справочники и сопровождающие их простейшие приемы". Далее нажимается клавиша "курсор вправо", после чего на экране появляется автоматически сгенерированный первый из необходимых приемов. Для сохранения и компиляции его нажимаем F3. Для извлечения очередного приема справочника формульного редактора нажимаем "ш". Затем - снова F3, и так до тех пор, пока генератор приемов не исчерпает новые приемы.

Если указатель "ключ(...)" не использовался, то на этом присоединение понятия A к словарю формульного редактора завершается. В противном случае нужно проверить, не была ли уже использована предложенная комбинация K_1, K_2. Для этого в серии новых приемов находим прием с заголовком "префикс(2)" и переходим к просмотру его программы (клавиша "Home"). Здесь нужно рассмотреть ту ветвь программы справочника "префикс", к которой относится текущий фрагмент программы, чтобы убедиться, что в ней уже не было другого фрагмента, содержащего оператор "равно(конец(x1) K_2)" или эквивалентный ему оператор, связывающий x1 с K_2. Иногда новый и старый фрагменты оказываются занесены в общую ветвь, идущую после указанного оператора. Если такой фрагмент был, то сочетание клавиш уже использовано. Тогда нужно вернуться в редактирование теоремы приема (только данного), выбрать другое сочетание клавиш, перекомпилировать прием, и снова предпринять указанную проверку.

12.6 Инициализация пакетных операторов

Перед тем, как создавать приемы нового пакетного оператора, необходимо провести определенную подготовительную работу - выбрать название оператора и создать приемы справочников, описывающих формат этого оператора. Предусмотрены два

способа такой инициализации оператора - вручную либо с применением специального интерфейса.

12.6.1 Инициализация оператора вручную

Проверочный оператор

Для ввода нового проверочного оператора необходимо прежде всего выбрать (либо ввести новый) логический символ - название этого оператора. Затем вводится прием справочника "легковидеть", определяющий формат оператора (см. выше подраздел "Приемы справочников" раздела "Типы заголовков приемов"). После ввода псевдотеоремы справочника "легковидеть" рекомендуется нажать "а", выбрать первый пункт оглавления типов приемов (курсор вправо), и вводить далее серию приемов, нажимая поочередно F3 и "ш" - пока такие приемы будут появляться. Кроме справочника "легковидеть", будут также введены прием справочника "очевидно" и два простейших приема самого проверочного оператора - прием "быстрпроверка", обеспечивающий непосредственное обнаружение в контексте проверяемого утверждения, а также ряд других простейших способов проверки, и прием "контрольбуфера", предпринимающий попытку извлечь готовый результат из буфера предыдущих обращений.

Синтезатор

Для ввода нового синтезатора определяется его название, и вводятся приемы справочников "синтезатор" и "значения", определяющие формат синтезатора. После ввода псевдотеоремы справочника "синтезатор" рекомендуется нажать "а" и выбрать первый пункт оглавления типов приемов - это приведет к автоматической генерации приема данного справочника, справочника "значение" и первого приема синтезатора, осуществляющего попытку извлечь готовый результат из буфера результатов обращения к данному синтезатору. Как и в случае проверочных операторов, здесь последовательно нажимаются F3 и "ш".

Нормализатор

Для ввода нового нормализатора выбирается его название и вводится прием справочника "быстрпреобр", определяющий формат оператора. В случае нормализатора общей стандартизации термов с заданным заголовком вводятся также приемы справочников "нормализатор" и "ключоператора". Как и в предыдущих случаях, после ввода псевдотеоремы справочника "быстрпреобр" рекомендуется нажать "а" и использовать оглавление типов приемов - тогда, кроме приема данного справочника, будет также введен первый фиктивный прием нормализатора, обеспечивающий смену текущего уровня его работы (его заголовок - "окончание"). В случае нормализатора общей стандартизации нужно не забыть ввести после этого также приемы справочников "нормализатор", "ключоператора".

Анализатор

Для ввода нового анализатора выбирается его название и вводится прием справочника "анализатор", определяющий формат оператора. Это делается с помощью ог-

лавления типов приемов, причем одновременно создается прием для переключения текущего уровня анализатора (его заголовок - "внешвывод").

Оператор фильтра

Для ввода нового оператора фильтра выбирается его название и вводятся приемы справочников "блок", "блокпроверок", определяющие формат оператора.

12.6.2 Интерфейс ускоренной инициализации оператора

Для упрощения процесса создания нового пакетного оператора создан специальный интерфейс. Чтобы создать с его помощью оператор, выбирается тот раздел оглавления базы приемов, из которого будет сделана ссылка на ветвь данного оператора. Затем нажимается клавиша Ctr-п.

После нажатия на экране возникает диалоговый блок для ввода названия оператора и выбора его типа. Чтобы ввести название, предпринимается нажатие левой клавиши мыши внутри верхнего окна ("Название оператора"). Тогда появляется курсор текстового редактора, с помощью которого и вводится название. Это название может быть либо новым, либо старым, но не использованным ранее в качестве названия оператора ЛОСа. В случае нового названия оно сразу же регистрируется в файлах словаря. Если вводится нормализатор общей стандартизации выражений с заданным заголовком, то его название можно вообще не вводить, а сразу нажать левую кнопку мыши в окне "Нормализатор". Иначе - нажатие левой кнопки мыши в одном из окон "Нормализатор", "Проверочный оператор", "Синтезатор", "Анализатор", "Оператор фильтра" предпринимается после ввода названия. После этого нажатия появляется новый диалоговый блок. В зависимости от типа оператора, далее выполняются следующие действия:

1. Нормализатор. Если вводится нормализатор общей стандартизации выражений с заданным заголовком, то этот заголовок набирается в левом верехнем окне (предварительно нажимается клавиша "с" либо на этом окне нажимается левая кнопка мыши). Если набрано название "...", то оператор будет называться "усм...". Автоматически устанавливается ряд значений в других окнах, так что в данном случае сразу можно нажать Enter. Если же вводимый нормализатор не является нормализатором общей стандартизации выражений с заданным заголовком, то левое верхнее окно не заполняется, но уточняются прочие параметры оператора (число уровней срабатывания, наличие списка посылок, использование буфера и т.д. - все эти пункты соответствуют стандартным элементам из описания формата оператора). Далее также нажимается Enter.

2. Проверочный оператор. Здесь в верхнем окне вводится текстовым редактором в скобочной записи вид проверяемого утверждения. В окне, расположенном ниже, перечисляются формульным редактором входные переменные из шаблона проверяемого утверждения, расположенные в том порядке, в котором они будут располагаться при обращении к оператору. Далее указывается число уровней срабатывания (обычно больше 1), и нажимается Enter.

3. Синтезатор. В верхнем окне набирается одно или несколько утверждений, представляющих собой те условия, которым должны удовлетворять подбираемые

синтезатором выходные значения. Затем в окнах, расположенных ниже, перечисляются входные и выходные переменные, а также указывается число уровней срабатывания. Нажатия левой клавиши мыши на окнах, указывающих возможные уточнения типа синтезатора, приводят к появлению либо удалению знаков "плюс" справа от этих окон. По завершении ввода нажимается Enter.

4. Анализатор. В верхней части экрана расположены друг под другом 5 пар окон. В левом окне пары указывается уровень сканирования задачи на исследование, по достижении которого будет происходить обращение к анализатору; в правом окне - максимальный уровень соответствующего обращения к анализатору, по превышении которого работа анализатора будет прервана. Обычно указывается не более одной - двух таких пар (например, одна - для предварительного обращения, имеющего целью быстро усмотреть нужные следствия, и другая - для более длительного анализа). По завершении ввода нажимается Enter.

5. Оператор фильтра. В верхнем окне текстовым редактором набирается краткое описание смысла входных и выходных переменных оператора. Оно будет использоваться в качестве текста того пункта оглавления базы приемов, в котором расположатся задающие формат оператора справочники. Оно же будет выводиться вместо фиктивной теоремы приема "блок" при просмотре приемов оператора. Далее перечисляются (как термы "вхождение(...)", "тип(...)" и т.п.) типы значений входных и выходных переменных, уточняется тип обращения к оператору, и нажимается Enter.

По окончании указанных действий будут созданы все необходимые приемы, инициализирующие работу с оператором, причем эти приемы автоматически регистрируются в новых подпунктах оглавления приемов, созданных внутри ветви оператора.

Глава 13

Примеры записи приемов на ГЕНОЛОГе и упражнения

Сравнительно подробное описание приемов ГЕНОЛОГа, созданных для различных предметных областей, содержится в 2-6 томах монографии "Компьютерное моделирование логических процессов". Объем этих томов - примерно 5000 страниц. К сожалению, аппарат решения задач в том или ином разделе обычно "рассыпается" на множество сравнительно мелких приемов, что существенно затрудняет восприятие целостной картины. Лишь при работе с примерами удается понять, какие приемы избыточны, а каких не хватает. Трудно выделить какие-то "самые важные" приемы и рассмотреть их в данном учебнике. При таком выделении все равно будет пропущено множество крайне существенных моментов. Поэтому для серьезного понимания того, как работает решатель и как применяется ГЕНОЛОГ, придется ознакомиться с указанными томами монографии. Повторять их фрагменты здеь не имеет смысла. Однако, чтобы возникло хотя бы предварительное представление о программировании на ГЕНОЛОГе, разберем несколько десятков записанных на этом языке приемов, и проиллюстируем на них наиболее часто встречающиеся элементы описания приема.

13.1 Примеры записи приемов

Приводимые в данном разделе примеры имеются в реальной базе приемов решателя. Рекомендуется при ознакомлении с очередным приводимым примером приема войти в просмотр его оригинала в базе приемов, и находить в этом оригинале те элементы, о которых далее пойдет речь.

13.1.1 Приемы тождественной замены

Начнем с приема, осуществляющего переход от логарифма произведения к сумме логарифмов (путь к нему в оглавлении базы приемов - "Элементарная алгебра", "Логарифмы", "Общая стандартизация выражений", "Логарифм произведения"). Сразу ясно, что здесь возможны различные варианты приема - в зависимости от того, усматриваются ли из контекста знаки сомножителей: случай, когда усматривается неотрицательность одного сомножителя и произведения остальных; случай, когда усматривается неположительность одного сомножителя и произведения остальных; случай, когда при преобразовании вводятся модули, и т.п. Остановимся лишь на первом из них. Теорема приема здесь может быть представлена в следующем виде:

$$\forall_{abc}(0 \le a \ \& \ 0 \le b \to \log_c(ab) = \log_c a + \log_c b).$$

Сразу заметим, что в ней опущены условия на область допустимых значений: $0 < c, \neg(c = 1)$, число(a), число(b), число(c). Это - простейшее проявление принципа контекстной корректности для теорем приемов: отбрасываются проверки тех условий, которые, при соблюдении некоторых необременительных требований на "правильную" постановку задачи и на фактически допускаемые решающими правилами приемов преобразования, автоматически будут выполняться в контексте применения приема.

Заголовок приема определяет направление замены слева направо, т.е. представляет собой логический символ "второйтерм" (в приемах замены заголовок указывает на заменяющий терм равенства либо эквивалентности).

Первый из фильтров приема - "уровень(1)" - определяет уровень срабатывания приема при сканировании задачи. Ясно, что в большинстве стандартных ситуаций целесообразно бывает почти сразу переходить к логарифмам сомножителей, чтобы после приведения подобных членов добиться упрощения выражения. После такой стандартизации, на этапе редактирования ответа, можно будет предпринять обратную свертку в логарифм произведения. Разумеется, прием можно было бы сделать более избирательным, блокируя данное преобразование, если оно заведомо ничего не дает. Впрочем, такие дополнительные фильтры при обучении вводятся лишь по мере надобности, и на рассматривавшихся задачах для данного приема не возникли.

Следующий фильтр приема имеет вид "или(не(тип(преобразовать)) посылка коммент(длина))". Он блокирует применение приема для изменения условия задачи на преобразование, если эта задача уже имеет комментарий "длина". Такой комментарий вводится после завершающей обработки преобразуемого терма процедурой "свертка", обращающейся к пакетным нормализаторам сокращенной записи - очевидно, на этом этапе развертка логарифма произведения в сумму логарифмов нецелесообразна.

Наконец, последний фильтр имеет вид "не(контекст(подтерм(равно(теквхожд x4)) тип(описать) условие) не(известно(теквхожд)) известно(x4))". Он блокирует преобразование логарифма в сумму, если этот логарифм содержит неизвестные и является левой частью уравнения, правая часть которого известна. В такой ситуации целесообразнее сразу провести потенцирование.

Оба антецедента теоремы приема обрабатываются с помощью проверочных операторов (именно, "усмменьшеилиравно"). Для задания такого способа их обработки служит указатель "блокпроверок(1 2)".

Чтобы предусмотреть случай, когда перед произведением стоит минус, в прием введен указатель идентификации "заменазнака(минус x1)". При наличии минуса он будет передан сомножителю x1; проверка неотрицательности выполняется для модифицированного таким образом x1.

Наконец, указатель "заменавхождений" определяет режим одновременной замены всех вхождений в задачу данного логарифма (с контролем возможности усмотрения в контексте замены неотрицательности сомножителей); указатель "нормализатор" - выделяет прием как выполняющий преобразование общей стандартизации. Он активизирует попытку общей стандартизации надтермов преобразуемого выражения после выполнения замены.

Подтермы заменяющего терма приема снабжены указателями нормализации, обеспечивающими их общую стандартизацию. Сам заменяющий терм $\log_c a + \log_c b$ снабжен нормализатором "нормплюс"; выражения $\log_c a$,$\log_c b$ - нормализаторами "нормлогарифм". Например, если основание логарифма c совпадало с множителем b, то нормализатор "нормлогарифм" заменит выражение $\log_c c$ на 1, и в качестве заменяющего терма будет использовано выражение $\log_c a + 1$.

Разобранный только что пример весьма прост; для сравнения приведем другой, более сложный (хотя далеко не самый сложный из имеющихся в решателе) прием тождественной замены. Он осуществляет стандартизацию тригонометрических выражений с помощью тождества $(\sin x)^2 + (\cos x)^2 = 1$. Мы специально используем здесь обозначение $(\sin x)^2$ вместо $\sin^2 x$, так как в формульном редакторе решателя принята именно эта запись.

Прежде всего, предпримем некоторое обобщение данного тождества, чтобы им можно было пользоваться даже в неявных ситуациях. Первый шаг обобщения - введение коэффициента a перед слагаемыми левой части: $a(\sin x)^2 + a(\cos x)^2 = a$. Следующий шаг - учет возможных множителей в a, представляющих степени синуса либо косинуса x. Если такие множители есть, то фактически в задаче будут усматриваться не вторые, а какие-то другие степени синуса и косинуса x; именно, здесь понадобится теорема вида $m - p = 2 \ \& \ q - n = 2 \rightarrow a(\cos x)^m(\sin x)^n + a(\cos x)^p(\sin x)^q = a(\cos x)^p(\sin x)^n$. Она получена из предыдущей выделением в a сомножителей $(\cos x)^p$, $(sinx)^n$; для остатка множителей снова использована буква a. Чтобы такая обобщенная теорема давала необходимую идентификацию и в простейшем случае, когда никаких дополнительных множителей указанного вида в общем коэффициенте не было, придется использовать указатели приема, объясняющие компилятору, что значения p и n могут равняться 0, а соответствующие множители $(\cos x)^p$, $(\sin x)^n$ - отсутствовать. Следующий шаг обобщения теоремы - учет случая, когда коэффициенты при слагаемых различны и применяется частичное исключение квадратов синуса и косинуса - например, при переходе от $5(\sin x)^2 + 3(\cos x)^2$ к $2(\sin x)^2 + 3$. Если выделять константу на основе того численного коэффициента, который меньше по модулю, причем делать это только в случае совпадения знаков слагаемых, то окончательно возникает теорема вида:

$m - p = 2 \ \& \ q - n = 2 \ \& \ (0 < r \ \& \ 0 < s \ \& \ t = \min(r, s) \vee r < 0 \ \& \ s < 0 \ \& \ t = \max(r, s)) \ \& \ g = r - t \ \& \ h = s - t \rightarrow ra(\cos x)^m(\sin x)^n + sa(\cos x)^p(\sin x)^q = ga(\cos x)^m(\sin x)^n + ha(\cos x)^p(\sin x)^q + ta(\cos x)^p(\sin x)^n$.

Здесь r, s - числовые коэффициенты; a - произведение прочих общих у рассматриваемых слагаемых множителей (если они есть); t - определяемый в третьем антецеденте числовой коэффициент с наименьшим модулем; g, h - остатки числовых коэффициентов слагаемых после выделения из них t. Заметим, что хотя бы один из этих остатков всегда равен 0, но для упрощения записи теоремы оба они явно фигурируют в заменяющем выражении. Применение к данному выражению нормализаторов общей стандартизации сразу же отбросит нулевые слагаемые, и в итоге заменяющий терм будет выглядеть вполне неизбыточным образом. Замедление, проистекающее из такого, на первый взгляд не очень экономного способа вычислений, в действительности пренебрежимо мало по сравнению с паузами сканирования между применениями приемов. Поэтому исключение разбора случаев за счет применения заменяющих термов избыточно общего вида, в сочетании с обработкой их нормализаторами, устраняющими в конкретных случаях всю возможную избыточность - это

стандарт программирования на ГЕНОЛОГе.

Заметим, что в приведенной выше записи для краткости опущен квантор общности по всем переменным, встречающимся в тождестве. При фактическом вводе теоремы приема наличие такого квантора является обязательным. В большинстве случаев пропуск в кванторной приставке нескольких переменных особой роли играть не будет, однако иногда это может привести к неправильной компиляции приема (в особенности там, где перед компиляцией выполняется преобразование теоремы приема - например, в геометрии).

Прием, имеющий теорему указанного выше вида, можно найти в разделе ("Элементарная алгебра","Тригонометрия", "Синус", "Общая стандартизация выражений", "Стандартизация тригонометрических многочленов с применением тождества ..."). В концевом пункте этого раздела расположено несколько приемов (кроме рассматриваемого здесь применения тождества для суммы квадратов синуса и косинуса, обобщаются также замены вида $1-(\cos x)^2=(\sin x)^2$ и $1-(\sin x)^2=(\cos x)^2$, а кроме того, рассматривается стандартизация с участием четвертых степеней синуса и косинуса). К нужному нам приему можно перейти, нажимая клавишу "курсор вниз" до последней точки цепи приемов. Переменные в его теореме обозначены иначе, чем в приведенной выше, однако в остальном эти теоремы идентичны.

Заголовок приема - "второйтерм"; фильтры - "уровень(1)" (тождество применяется почти сразу же, как только предоставляется такая возможность); "натуральное(m)", "натуральное(n)", "натуральное(p)", "натуральное(q)"; "десчисло(r)", "десчисло(s)". Последние фильтры ограничивают применение приема лишь теми случаями, когда показатели степеней при синусе и косинусе суть десятичные записи натуральных чисел, а коэффициенты r,s также суть десятичные записи числовых констант.

Указатель "программа(1 2 3 4 5)" означает, что все антецеденты являются программно реализуемыми; это возможно из-за того, что встречающиеся в них переменные идентифицированы, согласно фильтрам, с числовыми константами, и соответствующие операции вычитания, определения минимума и максимума, а также проверка неравенств реализуются непосредственно.

Указатели "перечень(r десчисло(r))" и "перечень(s десчисло(s))" заставляют компилятор сразу идентифицировать r,s с произведениями всех числовых множителей рассматриваемых слагаемых. В действительности, благодаря предшествующей стандартизации, заключающейся в перемножении числовых констант, таких множителей может быть не более одного; при их отсутствии переменная идентифицируется с единицей рассматриваемой ассоциативно-коммутативной операции, т.е. в данном случае с обычной единицей. В принципе, данные указатели здесь не обязательны - они лишь препятствуют отнесению числовых коэффициентов к произведению a остальных общих множителей.

Указатели "единица(1 m n p q r s a)" и "заменазнака(минус r s)" определяют возможность вырожденных, единичных значений степеней и коэффициентов, а также возможность наличия знаков "минус" перед слагаемыми, которые будут отнесены к числовым коэффициентам.

Указатели "подстановка(фикс(0 1 1 4) n 0)" и "подстановка(фикс(0 1 2 3) p 0)" определяют возможность отсутствия множителей $(\sin x)^n$ и $(\cos x)^p$, причем в этом случае показатели степени n,p идентифицируются с нулем.

Указатель "пустослово(a)" означает, что список оставшихся общих множителей может быть пустым - он нужен, чтобы компилятор не вставил проверку непустоты такого списка. Указатель "пересечениесписков(фикс(0 1 1)фикс(0 1 2))" означает, что заголовки слагаемых (даже после отбрасывания возможных знаков "минус") не обязательно должны быть символами умножения. При отсутствии данного указателя компилятор вставил бы проверку наличия заголовка "умножение" несмотря на наличие указателя "единица(...)", так как произведения слишком длинны и автоматическое распознавание возможности вырожденного случая здесь пока не предусмотрено.

Указатель "титр(набор(1 терм(фикс(0 1 1)фикс(0 1 2))))" определяет поясняющий срабатывание приема текст. Шаблон этого текста можно просмотреть и при необходимости отредактировать, нажав из просмотра приема клавишу "6". В данном случае появится следующий текст: "Преобразуем слагаемые () и (), используя формулу для суммы квадратов синуса и косинуса". Вместо скобок в него будут вставлены выражения, определяемые указателем, т.е. в точности рассматриваемые слагаемые - их указатели вхождения в теорему суть "фикс(0 1 1)" и "фикс(0 1 2)".

Нормализаторы приема просматриваем, нажав сначала клавишу "5" и используя для смены нормализатора клавиши PageDown, PageUp (выход из просмотра - по End). Все подвыражения заменяющей части с заголовками "степень", "умножение", "плюс", снабжены нормализаторами общей стандартизации "нормстепень", "нормумножение", "нормплюс". Кроме того, сами преобразуемые слагаемые также снабжены нормализаторами "нормумножение". Это сделано для того, чтобы в сопровождающий текст подставлялись выражения, у которых устранены умножения на коэффициенты 1, возникшие при идентификации.

13.1.2 Приемы эквивалентной замены

Приемы эквивалентной замены обычно используются либо для общей стандартизации утверждений, либо для разрешения относительно неизвестных условий задачи на описание (уравнения, неравенства и т.п.).

Рассмотрим прием, предназначенный для решения квадратных уравнений. В оглавлении базы приемов к нему ведет путь "Элементарная алгебра", "Степени", "Решение уравнений", "Решение квадратного уравнения"; далее нажатиями клавиши "курсор вниз" переходим к последнему приему группы приемов данного концевого пункта.

При формулировке теоремы приема следует учитывать, что приемы предварительной стандартизации уравнений переводят все известные слагаемые в правую часть уравнения, оставляя в левой части только неизвестные слагаемые. Соответственно, квадратное уравнение будет иметь вид $ax^2 + bx = c$, а не традиционный вид $ax^2 + bx + c = 0$. Теорема приема записывается в следующем виде (для удобства ссылок в ней взяты обозначения переменных из базы приемов - неизвестная вместо x обозначена d):

$\forall_{abcde}(e = b^2 + 4ac \to ad^2 + bd = c \leftrightarrow (\neg(a = 0)\ \&\ 0 \le e\ \&\ (d = \frac{\sqrt{e}-b}{2a} \vee d = -\frac{\sqrt{e}+b}{2a})$
$\vee bd = c\ \&\ a = 0))$.

Единственный антецедент теоремы служит для того, чтобы можно было ввести вспомогательное обозначение e для дискриминанта трехчлена. Фактически, как будет

видно из дальнейшего, он не только вводит обозначение для дискриминанта, но и вычисляет этот дискриминант, обращаясь для преобразований его к вспомогательным пакетным операторам.

Заголовок приема - "второйтерм". Фильтры "уровень(1 2)", "условие", "тип(описать)", "известно(a)", "известно(b)", "известно(c)", "не(известно(d))", "корень" указывают, что прием применяется при сканировании задачи на описание для известных a, b, c и не известном d, причем уровень сканирования должен быть равен 1 либо 2, а преобразуемое равенство должно иметь корневое вхождение, т.е. собственно являться уравнением.

Фильтр "альтернатива(неизвестные(2)уровень(1)уровень(2))" уточняет, что в задачах, имеющих более одной неизвестной, уровень срабатывания приема равен 1, а в задачах с одной неизвестной он равен 2. Такого рода фильтры возникают, если в тех или иных примерах желательно обеспечить опережающее срабатывание одних приемов и отложить срабатывание других. Они имеют сугубо эвристический характер и могут при последующем обучении многократно изменяться или отбрасываться - если локальные "конфликты" между приемами, из-за которых эти фильтры возникли, были устранены другими методами.

Последние два фильтра - "или(равно(числонеизвестных(корень)1)не(контекст(условие(х6)заголовок(х6 целое натуральное)первыйсимвол(х6 х7)неизвестная(х7)входит(х7 d))))" и "не(контекст(ключ(цели связка х6)пересекаются(х6 d)))" - отсекают случаи уравнений, решаемых средствами других разделов. В первом из них блокируется применение формулы корней квадратного уравнения для уравнения в целых числах, имеющего более одной неизвестной; во втором - применение этой формулы при рассмотрении дифференциального уравнения (для решения дифференциальных уравнений, не разрешенных явно относительно производной, используется специальный прием, который обозначает производную новой переменной - неизвестной, относительно которой и выполняется разрешение в рамках некоторой вспомогательной задачи). Такие фильтры возникают по мере того, как обучение решателя распространяется на новые разделы. Часто при этом бывает нужна некоторая расчистка для блокировки срабатываний ненужных приемов из ранее рассматривавшихся разделов; обычно для нее бывает достаточно рассмотрения нескольких первых примеров.

Указатель "идентификатор(1)" определяет, что первый антецедент теоремы используется для ввода вспомогательного обозначения e. Указатели "единица(1 a b)" и "заменазнака(минус a b)" определяют, что коэффициенты a, b могут принимать вырожденное значение 1 и что минус перед соответствующими членами передается данным коэффициентам.

Наконец, указатель "примечание(условие(меньше(1 число(условие(х6)заголовок(х6 равно)))разборслучаев))" вводит комментарий "разборслучаев" к преобразованному уравнению, учитывая свозможность, что оно после преобразования может оказаться дизъюнкцией, если число уравнений в задаче больше одного. Такой комментарий форсирует разбор случаев по данной дизъюнкции - для предотвращения возникновения новых дизъюнкций, которые могли бы появиться при независимом преобразовании остальных уравнений.

Перечислим нормализаторы приема. Для вычисления дискриминанта используется цепочка обращений к нормализаторам "нормплюс", "стандплюс", "видумножение". Первый предпринимает общую стандартизацию сумм; второй выполняет раскрывание скобок; последний - пытается разложить дискриминант на множители,

чтобы впоследствии упростить извлечение из него квадратного корня. Утверждение $a = 0$ из заменяющей части обрабатываемого нормализатором "нормусм" (этот нормализатор предпринимает попытку усмотрения истинности либо ложности обрабатываемого им утверждения путем применения проверочного оператора, и в случае удачи заменяет данное утверждение на логическую константу "истина" либо "ложь"), а также нормализатором "нормчисло" (он служит для простейшей стандартизации числовых равенств). Если удается усмотреть истинность либо ложность утверждения $a = 0$, то в заменяющей части вместо него сразу же вводится соответствующая логическая константа.

Аналогично, утверждение $0 \leq e$ из заменяющей части преобразуется нормализаторами "нормусм" и "нормменьшеилиравно" (простейшая стандартизация нестрогих неравенств). Если его истинность либо ложность очевидна, то вместо него сразу же будет подставлена логическая константа. При обращении к указанным нормализаторам вводится дополнительная посылка $\neg(a = 0)$.

При преобразовании квадратного корня из дискриминанта используются нормализаторы "нормстепень" и "сбросмодуля". Последний нормализатор отбрасывает модули у сомножителей преобразуемого им выражения - в данной ситуации модуль не нужен, так как все равно при определении корней берутся два значения корня из дискриминанта, отличающиеся знаком. При обращении к нормализаторам вводится дополнительная посылка $0 \leq e$.

Выделим, далее, цепочки "нормплюс", "видумножение", "посылки($0 \leq e$)", используемые для разложения на множители числителей $\sqrt{e} - b$, $\sqrt{e} + b$, а также нормализатор "видумножение", обеспечивающий разложение на множители коэффициента a. Наконец, заметим, что ко всему заменяющему терму применяется нормализатор "нормлог", обеспечивающий общелогическую чистку утверждения, в частности, устраняющий содержащиеся в нем логические константы.

Следующий пример приема эквивалентной замены - интегрирование дифференциального уравнения с разделяющимися переменными. Этот прием может быть найден в разделе ("Дифференциальные уравнения", "Уравнения первого порядка", "Уравнение с разделяющимися переменными", "Интегрирование уравнения с разделяющимися переменными"). Берется последний прием из группы приемов данного концевого пункта - переход к нему осуществляется путем нажатий клавиши "курсор вниз". Напомним, что дифференциальное уравнение с одной неизвестной записывается как равенство с производными и формально представляет собой условие не на саму неизвестную функцию y, а лишь на значения ее и производных в некоторой "текущей" точке x. Более аккуратная в логическом отношении запись потребовала бы ввода явного обозначения для функций, определяемых левой и правой частями равенства, и оказалась бы более громоздкой и непривычной. Поэтому в логическом языке решателя (как и в обычной практике) здесь применяется контекстная семантика: равенство для значений в точке x в действительности понимается по умолчанию, как равенство для функций, причем такое понимание определяется специальной целевой установкой задачи - наличием цели (связка $x_1 \ldots x_n$), перечисляющей все аргументы неизвестных функций, встречающихся в условиях задачи. Эта цель является как бы вынесенной за скобки логической конструкцией, связывающей встречающиеся в задаче переменные $x_1, \ldots x_n$. Учитывая такое соглашение о записи дифференциальных уравнений в условиях задачи, для решения дифференциального уравнения с разделяющимися переменными, приведенного предварительной

стандартизацией к виду

$$a(x)b(y(x))\frac{dy(x)}{dx} + c(x)e(y(x)) = 0,$$

будем иметь следующую теорему приема:

$$\forall_{abcexyhv}(\int \frac{b(z)}{e(z)}dz = \lambda_z(h(z), \text{число}(z)) \ \& \ \int \frac{c(x)}{a(x)}dx = \lambda_x(v(x), \text{число}(x)) \rightarrow$$
$$a(x)b(y(x))\frac{dy(x)}{dx} + c(x)e(y(x)) = 0 \leftrightarrow \exists_p(\text{число}(p) \ \& \ h(y(x)) = -v(x) + p)$$
$$\& \ \neg(e(y(x)) = 0) \vee e(y(x)) = 0 \ \& \ a(x)b(y(x))\frac{dy(x)}{dx} = 0 \ \& \ \text{число}(\frac{dy(x)}{dx})).$$

Первый и второй антецеденты теоремы будут определять вычисление соответствующих интегралов; возникающие после интегрирования функции $h(z)$ и $v(x)$ (где вспомогательная переменная z введена на период интегрирования вместо $y(x)$) используются для описания параметрического семейства решений с произвольной постоянной p. Особо учтен подслучай обращения в 0 второго слагаемого, причем утверждение "число$(dy(x)/dx)$"из этого подслучая введено как эквивалент условия дифференцируемости функции $y(x)$ в тех точках, где она рассматривается.

Заголовок приема - "второйтерм". Фильтры "условие(2)", "условие", "тип(описать)", "корень", "неизвестная(y)" означают, что прием срабатывает на втором уровне сканирования (сразу после простейшей стандартизации уравнения), причем преобразуется условие задачи на описание, у которого y - неизвестная функция. Фильтры "известно(фикс(0 1 1 1 1))", "известно(фикс(0 1 1 2 1))" означают, что выражения $a(x)$, $c(x)$ не содержат неизвестных.

Указатель "идентификатор(1 2)" определяет рассмотрение первого и второго антецедентов как равенств, определяющих значения вспомогательных выражений. В каждом из них изначально идентифицированы все переменные левой части; эта часть будет сформирована и обработана сопровождающими ее нормализаторами (они и вычислят интегралы). Затем полученное выражение будет идентифицироваться с правой частью равенства, что и позволит получить $h(z)$,$v(x)$ для дальнейшего их использования.

Указатель "отображение(a b c e h v)" означает, что для перечисленных в нем переменных Y выражения $Y(X)$ - в скобочной записи они выглядят как "значение(Y X)" - будут идентифицироваться с произвольными выражениями. При отсутствии данного указателя $Y(X)$ идентифицировалось бы только с выражением вида "значение(...)".

Указатель "примечание(серия)" сопровождает преобразованное уравнение комментарием "серия", означающим, что возникший в нем квантор существования дает параметрическое описание ответа.

Указатели "перечень(b не(известно(b)))", "перечень(e не(известно(e)))" определяют идентификацию $b(y(x))$, $e(y(x))$ со всеми содержащими неизвестные множителями соответствующих слагаемых. Указатели, определяющие выделение указанных в теореме двух слагаемых путем группировки всех членов суммы, имеющих производную, и всех членов, имеющих множителем неизвестную функцию, отсутствуют, так как эти группировки заблаговременно осуществляются другими приемами.

Указатели "единица(1 a b c)" и "заменазнака(минус a c)" означают возможность вырожденных значений 1 для $a(x)$, $b(y(x))$, $c(x)$, а также передачу знаков "минус" перед слагаемыми выражениям $a(x)$,$c(x)$.

Указатели "новаргумент(b x фикс)", "новаргумент(e x фикс)" определяют способ идентификации $e(y(x))$, $b(y(x))$. Перед идентификацией этих выражений переменные y, x должны быть уже идентифицированы. Компилятор создает обращение к процедуре "новаргумент", проверяющей, что вхождения переменной x внутри идентифицируемых с $e(y(x))$, $b(y(x))$ выражений расположены только внутри $y(x)$. Все такие вхождения выделяются, и при формировании $b(z), e(z)$ вместо них будет подставлена новая переменная z. То, что эта переменная - новая, следует пояснить компилятору отдельно, что и делает указатель "новаяпеременная(z)". Логический символ "фикс" в приведенных выше указателях введен, чтобы показать, что идентифицируемые выражения не подвергаются предварительной обработке нормализаторами, ориентированными на приведение всех подвыражений с x к виду $y(x)$ - очевидно, здесь это излишне.

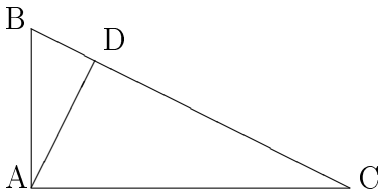
Указатель "конец(1 2)" отодвигает на конец трудоемкую обработку первого и второго антецедентов (т.е. фактически интегрирование уравнения) до того, как будут проверены все остальные условия срабатывания приема.

Наконец, указатель "удалениеусловия(число(производная(отображение(x4 число(x4)значение(y x4))x)))" удаляет из списка условий утверждение о существовании производной $dy(x)/dx$, которое после интегрирования уравнения не нужно.

Оба интеграла из антецедентов снабжены нормализатором "нормИнтеграл". Это - чрезвычайно большой пакетный нормализатор, осуществляющий в решателе формальное интегрирование. Перед обращением к нему подынтегральные выражения обрабатываются нормализатором "упрощинтеграл", выполняющим некоторые несложные предварительные преобразования. Этому нормализатору передается комментарий (переменная X), указывающий переменную интегрирования X. Для предварительного упрощения параметрического описания, возникающего после интегрирования уравнения, применен нормализатор "уравндифф". Впрочем, и нормализатор "упрощинтеграл", и нормализатор "уравндифф" - очень маленькие, почти вырожденные пакеты, которые при последующем развитии системы, возможно, будут отброшены. Прочие используемые в приеме нормализаторы осуществляют общую стандартизацию подтермов заменяющего терма.

13.1.3 Приемы вывода в посылках задачи

Рассмотрим один из простейших приемов вывода, связанный с прямоугольным треугольником, у которого проведена высота к гипотенузе (см. в оглавлении базы приемов путь "Элементарная геометрия", "Фигуры", "Треугольник", "Прямоугольный треугольник", "Высота прямоугольного треугольника, проведенная из прямого угла", "Соотношение для длин катета, гипотенузы и отрезка гипотенузы между основанием высоты и вершиной треугольника"). В этом случае прием уже снабжается чертежом:



Теорема приема имеет вид:

$\forall_{ABCD}$(прямая(AB)$\perp$прямая(AC) & $D \in$ прямая(BC) & прямая(AD)$\perp$прямая(BC) $\rightarrow l(BD) \cdot l(BC) = (l(AB))^2$).

Прием имеет заголовок "вывод"; он позволяет в тех случаях, когда длины отрезков BD,BC,AB уже введены в рассмотрение либо их ввод в рассмотрение представляется целесообразным, присоединить к посылкам задачи связывающее эти длины соотношение, указанное в теореме.

Фильтр "уровень(5 7)" определяет два возможных уровня срабатывания приема - пятый и седьмой. Хотя один из последующих фильтров уточняет, в каких случаях срабатывание на пятом уровне нецелесообразно, но он сравнительно слабый, и остается достаточно много ситуаций, при которых срабатывание возможно как на пятом, так и на седьмом уровнях. Такая избыточность иногда создается в приемах вывода, чтобы срабатывание было возможным, даже если необходимые для него посылки появляются в задаче не сразу, а лишь на достаточно высоких уровнях сканирования. Напомним, что другая возможность обеспечить срабатывание приема после появления недоставивших ранее посылок - уменьшить вес его "опорной" посылки тем приемом, который ввел представляющую интерес для ее пересмотра информацию.

Фильтры "посылка" и "или(тип(доказать)тип(исследовать))" встречаются практически во всех геометрических приемах, так как обычно геометрические задачи на вычисление решаются путем вывода следствий в посылках их блока анализа. Иногда здесь добавляется возможность срабатывания в посылках задач на преобразование, что бывает нужно для задач на нахождение геометрических мест точек.

Фильтры "не(равно(D B))" и "не(равно(D C))" представляют собой ускорители отсечения: они позволяют в процессе идентификации сразу отбрасывать положения точки D, совпадающие с B и C, не выполняя в этих случаях дальнейших проверок. Опыт показывает, что отсутствие в геометрическом приеме фильтров, отсекающих вырожденные положения точек, может существенно замедлить работу решателя, так как прием начнет при сканировании задачи предпринимать повсюду множество бессмысленных проверок. Поэтому они обычно вводятся еще до оптимизации приема на примерах.

Фильтры "усм(актив(расстояние(A B)))", "усм(актив(расстояние(B C)))", "усм(актив(расстояние(D x5)))" означают, что в посылках задачи должны встречаться расстояния $l(BC)$,$l(AB)$, а также расстояние от точки D до какой-либо другой точки. Эти фильтры обрабатываются идентифицирующими операторами - ввод фильтра "усм(F)" эквивалентен присоединению F к списку антецедентов теоремы приема и указанию на обработку этого антецедента идентифицирующим оператором. Фильтры вида "усм(...)", кроме ввода определенных ограничений на срабатывание приема, играют роль ускорителей идентификации, так как позволяют переходить от одного объекта к другому с использованием буферов идентифицирующих операторов - фактически это эквивалентно применению сетевого представления. Так, фильтр "усм(актив(расстояние(A B)))", если бы точка A была уже идентифицирована, а B - еще нет, позволил бы искать последнюю лишь среди тех точек, для которых явно введено в рассмотрение их расстояние до точки A.

Ввод в рассмотрение расстояний $l(AB)$,$l(BC)$ должен быть обеспечен до попытки применения данного приема; впрочем, можно создавать несколько различных версий одного и того же приема, имеющих различную фильтрацию и срабатывающих на различных уровнях. Одна из такиз версий данного приема (с уровнем срабаты-

вания 13) приведена в том же концевом пункте оглавления базы приемов, что и описываемый прием.

Фильтры "конец(или(известно(терм(расстояние(AB))) Неизв(терм(расстояние(AB)))))" и "конец(или(известно(терм(расстояние(BC))) Неизв(терм(расстояние(BC)))))" означают, что каждое из расстояний $l(AB)$,$l(BC)$, после обработки нормализатором "нормрасстояние", должно быть либо выражено термом без неизвестных, либо представлять интерес для определения неизвестных внешней задачи на описание. Последнее условие выражается при помощи фильтра "Неизв(...)" (как в данном случае), либо при помощи фильтра "неизв(...)", накладывающего несколько более сильные требования и применяемого чаще, чем "Неизв". Каждый из этих фильтров прослеживает цепочку равенств в посылках задачи, начинающуюся равенством, содержащим заданное выражение, и заканчивающуюся равенством, содержащим неизвестные внешней задачи на описание; связи между равенствами в такой цепочке - по вхождению общего выражения вида "расстояние(...)" либо "угол(...)". Подробнее - см. описания операторов "Неизв(...)", "неизв(...)". Указатели "конец(...)" в приведенных выше фильтрах отодвигают их проверку на конец программы приема - чтобы сравнительно трудоемкие процедуры определения нормализованных расстояний применялись лишь после установления необходимой конфигурации чертежа.

Фильтр "конец(не(равно(терм(расстояние(AB)) терм(расстояние(AC)))))" указывает, что треугольник ABC не должен быть равнобедренным - иначе вводимое приемом соотношение дублирует соотношения, вводимые другими, более простыми приемами.

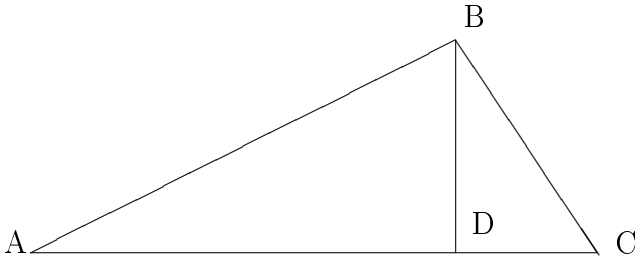
Следующие два фильтра - "конец(или(известно(терм(расстояние(AB))) известно(терм(расстояние(CD))) Неизв(терм(расстояние(CD))) комментпосылок(треугольник степень треугольник(ABC))))" и "или(уровень(7) комментпосылок(треугольник степень треугольник(ABC)) внешнеизв(терм(расстояние(AB))) известно(терм(расстояние(AB)))меньше(число(разряд(результат расстояние хб))2))" - возникли при попытке несколько ограничить применение данного приема, если для рассматриваемого прямоугольного треугольника уже выписывалось соотношение теоремы Пифагора. Ввод этого соотношения сопровождается добавлением комментария "треугольник степень треугольник(ABC)" к посылкам задачи. Оба фильтра чисто эвристические; формулировки их извлечены из сравнения двух классов задач, в одном из которых срабатывание приема было необходимым, а в другом - избыточным.

Наконец, последний фильтр "не(известно(результат))" - блокирует выписывание соотношения, если все его переменные суть известные параметры задачи.

Указатели приема суть "усм(1 2 3)" и "теквхожд(1)"; они определяют применение идентифицирующих операторов для обработки антецедентов теоремы. Активизация приема начинается с усмотрения в посылках задачи некоторого утверждения о перпендикулярности, отправляясь от которого программа приема и идентифицирует первый антецедент.

В качестве следующего примера вывода в посылках рассмотрим геометрический прием, выполняющий простейшее дополнительное построение - проведение высоты в треугольнике. Собственно говоря, имеется много различных приемов, выполняющих именно это действие - каждый применяется в некоторой своей специфической ситуации. Мы выберем одну из наименее громоздких версий - лежащую в разделе ("Элементарная геометрия", "Фигуры", "Треугольник", "Отрезок, соединяющий вершину

треугольника с точкой на противоположной стороне или на ее продолжении", "Высоты треугольника", "Проведение высоты в треугольнике"). В данном разделе находится два приема; берем последний, переходя к нему при необходимости нажатием клавиши "курсор вниз". Чертеж приема несложен:



Теорема этого приема - один из возможных примеров используемых в ГЕНОЛОГе псевдотеорем:

$\forall_{ABCD}$(актив($l(AB)$) & актив($l(BC)$) & актив($l(AC)$) $\rightarrow$ точка(D) & $D \in$ прямая(AC) & прямая(BD)$\perp$прямая(AC) & актив($l(BD)$)).

Она не является формулировкой какого - либо свойства геометрических конфигураций, а лишь определяет ту ситуацию, в которой нужно выполнить дополнительное построение (это делают антецеденты теоремы приема) и перечисляет те утверждения (в консеквенте), занесение которых в список посылок означает выполнение построения. Как и обычно, фильтры приема накладывают ряд дополнительных условий на ситуацию, в которой это построение будет делаться.

Антецеденты теоремы означают, что к моменту применения приема расстояния $l(AB)$, $l(BC)$, $l(AC)$ уже должны в явном виде упоминаться в посылках задачи. В геометрии такое упоминание расстояний, углов и площадей приводит к немедленному срабатыванию приемов, регистрирующих их в посылках вида "актив(X)", где X - данное расстояние, угол, площадь, и т.п.

Консеквент теоремы представляет собой конъюнкцию утверждений о новой точке D - основании высоты, проводимой приемом. Здесь указывается, что новый объект D представляет собой точку, лежащую на прямой AC, что прямые BD и AC перпендикулярны, и что расстояние $l(BD)$ - длина высоты - далее рассматривается как представляющее интерес.

Заголовок приема - "вывод". Фильтры "уровень(8)", "посылка", "или(тип(доказать) тип(исследовать))" - стандартные для планиметрии.

Следующий фильтр - "контекст(усм(принадлежит(х5 отрезок(AB)) принадлежит (х6 отрезок(AC)) перпендикулярно(прямая(х5 х6)прямая(AC)) актив(расстояние(х5 х6))) не(равно(х5 A)) не(равно(х5 х6)) не(равно(х5 C)))" - раскрывает подлинную мотивировку срабатывания приема: на одной из сторон AB,BC уже имеется точка, из которой опущен перпендикуляр на сторону AC, причем длина этого перпендикуляра явно упоминается в посылках. Разумеется, это обстоятельство делает данное дополнительное построение гораздо более объяснимым.

Фильтр "не(контекст(усм(перпендикулярно(прямая(х7 B)прямая(AC)))))" означает отсутствие ранее проведенного перпендикуляра к прямой AC, проходящего через точку B - он необходим, чтобы прием не начал бесконечный процесс ввода все новых и новых обозначений для основания такого перпендикуляра.

Фильтр "конец(или(контекст(усм(равно(расстояние(AB) расстояние(BC)))) не(контекст(список(х5 терм(расстояние(AB)) терм(расстояние(BC))) не(известно(х5))

))))" требует, чтобы либо треугольник ABC был равнобедренным, с вершиной B, либо чтобы расстояния $l(AB), l(BC)$ уже были вычислены. В этих случаях попытки связать с помощью теоремы Пифагора длины сторон треугольника и длину высоты $l(BD)$, ради которых и введен прием, становятся достаточно осмысленными.

Указатель "усм(1 2)" определяет обработку первого и второго антецедентов с помощью идентифицирующих операторов; точка инициализации приема при сканировании задачи - третий антецедент. Он дает точки A и C, а точка B далее извлекается из пересечения множеств точек, для которых выделены расстояния их до A и C.

Указатель "новыйсимвол(D фикс(0 1)фикс(0 2)фикс(0 3))" означает, что для D выбирается новая переменная, а в посылки внешней задачи на описание передаются утверждения "точка(D)", "$D \in$ прямая(AC)", "прямая(BD)$\perp$прямая(AC)".

Указатели "новаяпосылка(х5 контекст(вид(х5 принадлежит(х6 отрезок(х7 B)))) 2)", "новаяпосылка(х5 контекст(вид(х5 актив(расстояние(х6 х7)))усм(принадлежит (х6 прямая(A C))))3)" определяют переключение внимания после проведения высоты: веса посылок, указывающих принадлежность точки отрезку с концом в B, уменьшаются до 2, а веса посылок, выделяющих расстояния до точек прямой AC, уменьшаются до 3. Такие указатели вводятся по мере надобности в процессе прогонки решателя по задачнику - там, где указанное переключение внимания было совершенно необходимо для решения задачи либо позволяло существенно его ускорить. По существу, пока они представляют собой лишь обучающий материал для анализа подходов к автоматическому созданию в приемах средств переключения внимания. Однако, даже при накоплении их по указанному выше принципу (в общем, укладывающемуся в рамки концепции обучения методом "поощрений и наказаний"), решатель достаточно часто начинает замечать новые возможности, возникающие после срабатываний приемов.

Указатели "перпендикулярно(D B A C)" и "отрезок(B D)" определяют пополнение чертежа; первый из них вводит на чертеже точку D как основание перпендикуляра, опущенного из B на AC, второй - соединяет B с D отрезком.

Следующий пример - тоже геометрический. Он связан с ситуациями, в которых вводится численный параметр, обозначающий некоторую характеристику чертежа - расстояние, угол, и т.п. Такой параметр нужен лишь для промежуточных вычислений и в ответ включаться не будет. Однако, если временно рассматривать его как известную величину, то будет инициировано срабатывание многих полезных приемов, и шансы найти ответ задачи, в котором вспомогательные параметры, "сократившись" так или иначе, не встретятся, существенно возрастут.

Рассматриваемый прием вводит обозначение для длины катета прямоугольного треугольника, если нужно определить угол такого треугольника. Он расположен в оглавлении базы приемов по адресу ("Элементарная геометрия", "Перпендикулярно", "Ввод вспомогательного параметра - длины перпендикуляра, опущенного из точки на прямую"); как и ранее, берется последний прием из концевого списка, к которому переходим, нажимая клавишу "Курсор вниз" до прекращения изменений на экране.

Чертежа этот прием не имеет - ввиду простоты ситуации его применения. Теорема приема имеет вид:

$\forall_{ABCab}(\angle(ABC) = a \ \& \ \text{прямая}(AB) \perp \text{прямая}(AC) \rightarrow b = l(AB))$.

Заголовок приема - "вывод". Фильтры "уровень(6)", "тип(исследовать)", "цель(известно)" определяют срабатывание его на сравнительно большом (шестом) уровне в задаче на исследование.

Фильтр "внешнеизв(a)" означает, что выражение a не известно, и все входящие в него неизвестные являются теми неизвестными внешней задачи на описание, которые следует вычислить. В частности, в a не могут входить обозначения точек чертежа - обычно все эти обозначения суть неизвестные задачи на исследование и одновременно рассматриваются как известные во внешней задаче на описание, ибо входят в ее посылки. Данный фильтр указывает, что вычисление угла ABC в текущем контексте представляет интерес. Ясно, что для определения угла достаточно знать лишь отношение длин катетов, и поэтому временное рассмотрение длины $l(AB)$ одного из них, как известной, в рассматриваемой задаче перспективно.

Фильтр "контекст(посылка(х3)заголовок(х3 равно)вхождениетерма(х3 терм(расстояние(AB))х4)не(контекст(операнд(х5 х4)символ(х5 равно))))" означает, что расстояние $l(AB)$, для которого вводится вспомогательное обозначение, до этого уже было введено в рассмотрение в некотором равенстве, связывающем между собой параметры чертежа, причем это равенство не было явно разрешено относительно $l(AB)$.

Фильтр "не(контекст(равно(х3 терм(расстояние(AB))) или(известно(х3)внешнеизв(х3))))" отбрасывает действие приема, как заведомо бессмысленное, если расстояние $l(AB)$ либо уже вычислено, либо выражено только через те неизвестные, которые суть вычисляемые во внешней задаче параметры. В этих случаях необходимая инициализация приемов будет выполняться и без ввода специального обозначения для данного расстояния.

Фильтр "не(контекст(посылка(х3) вид(х3 равно(х4 расстояние(AB))) переменная(х4)))" блокирует ввод обозначения, если рассматриваемое расстояние уже равно некоторой переменной. Фильтр "не(Входит(вспомпараметр комментариипосылок))" блокирует применение приема, если до этого уже был введен какой-либо вспомогательный числовой параметр. Возможно, это и чересчур жесткое ограничение - вполне можно представить себе задачу, в которой нужно ввести несколько вспомогательных числовых параметров. В общем-то, некоторые из приемов решателя, вводящих такие параметры, лишены этого ограничителя, и при дальнейшем обучении может понадобиться несколько ослабить его и в данном приеме, однако пока такой надобности не возникало.

Фильтры "усм(актив(прямая(AB)))" и "усм(актив(прямая(AC)))" указывают, что прямые AB,AC уже были явно выделены в задаче.

Указатель "усм(2)" определяет применение идентифицирующего оператора при проверке условия перпендикулярности. Указатель "вспомпараметр(b фикс(0))" определяет выбор новой переменной b в качестве обозначения для $l(AB)$ и регистрацию ее в качестве временно известного параметра. Указатель вхождения "фикс(0)", ссылающийся на заносимое в посылки задачи на исследование утверждение $b = l(AB)$, также определяет перенесение его в список посылок внешней задачи на описание.

Чтобы в указанных фильтрах выражение "терм(расстояние(AB))" для уже вычисленного расстояния заменялось на его величину, в приеме введен нормализатор "нормрасстояние" для $l(AB)$. Одновременно он может затронуть и результирующее утверждение теоремы приема, однако в тех случаях, когда фильтры истинны, это несущественно.

Приведем теперь примеры приемов вывода в посылках, не относящиеся к планиметрии.

В режиме логического вывода решаются задачи на качественное исследование графика функции вещественной переменной. Исследуемая функция и связываемые с ней вспомогательные функции обозначаются переменными, и в посылках задачи на исследование происходит накопление различной информации об этих функциях. Например, расмотрим прием вывода, вычисляющий множество корней производной исследуемой функции (см. раздел "Математический анализ", "Общие свойства числовых функций", "Исследование функций", "Качественное описание графика функции", "Определение корней производной"). Его теорема имеет вид:

$$\forall_{afgb}(a = \lambda_x(g(x), f(x)) \rightarrow \mathrm{roots}(a, \mathrm{Dom}(a)) = set_x(f(x)\ \&\ g(x) = 0)).$$

Здесь a - обозначение для производной исследуемой функции (что будет видно из фильтров приема). Правая часть равенства в консеквенте обрабатывается вспомогательной задачей, в которой решается уравнение $g(x) = 0$; на основе результата решения создается новая посылка, дающая явное выражение для множествак корней производной на ее области определения. Конечно, данная теорема приема - это еще один пример псевдотеоремы, или почти - псевдотеоремы.

Фильтры "уровень(2)", "посылка", "тип(исследовать)", "цель(исследовать)" стандартные; новым является лишь последний из них, указывающий на то, что решается задача на качественное описание графика функции. Фильтр "контекст(посылка(x3) вид(x3 Производная(x4 a)))" означает наличие посылки, характеризующей функцию a как производную некоторой функции x4. Фильтр "комментпосылки(1 корни)" выражает отсутствие комментария к задающему функцию a равенству, означающего, что корни этой функции уже ранее вычислялись. Указатель "примечпосылки(1 корни)" создает такой комментарий в случае применения данного приема.

Указатель "примечание(ориентацияравенства)" снабжает вводимое равенство для множества корней производной пометкой, блокирующей перестановку его левой и правой частей. Уточнение ориентации равенства позволяет управлять последующими заменами, которые оно порождает. В данном случае выражение "roots(a,Dom(a))" будет заменяться на явное описание множества корней, а не наоборот. Указатель "примечание(равно)" снабжает новую посылку пометкой, блокирующей замену его подтермов с помощью других равенств. Указатель "примечание(упростить)" вводит пометку, блокирующую последующие попытки упрощать новое равенство.

Указатель "отображение(fg)" определяет идентификацию $f(x), g(x)$ с произвольными выражениями; Указатель "точкапривязки(отображение)" уточняет, что инициализация применения приема при сканировании будет начинаться с усмотрения символа "отображение" в равенстве для a. Наконец, указатель "лимит(20000000)" ограничивает трудоемкость попыток нахождения корней производной.

Нормализаторов приема всего два. Прежде всего, это обработка пакетным нормализатором "нормкласс" всей правой части нового равенства. Такой нормализатор предпринимает попытку избавиться от описателя "класс", перейдя к более простым средствам задания множеств. Однако, еще до обращения к нему, условия принадлежности классу $f(x)\ \&\ g(x) = 0$ преобразуются вспомогательной задачей на описание, в которой, собственно, и осуществляется основная работа данного приема. Нормализатор приема, организующий обращение к этой задаче, имеет вид "задача(6 тип(описать) цель(неизвестная(x)) полный явное прямойответ корни одз упростить)". Он сопровождается дополнением "удалениепосылок(x3 входит(x x3))", отбрасывающим из

контекста применения приема все утверждения с x. Применение его происходит путем решения вспомогательной задачи на описание, условиями которой служат условия принадлежности классу, посылками - урезанный указанным образом контекст, а неизвестной - x, причем эта задача решается до максимального уровня 6. Ответ задачи и берется в качестве результата применения нормализатора приема.

В качестве последнего примера приемов вывода рассмотрим прием из аналитической геометрии, извлекающий из условия параллельности прямой и плоскости соотношение для коэффициентов уравнений, задающих эти прямую и плоскость. Прием размещен в разделе ("Аналитическая геометрия", "Уравнение прямой в пространстве", "Условие параллельности прямой и плоскости").

Теорема приема имеет вид:

$\forall_{ABCDEKabcdefpqrs}$(прямая$(AB)\|$плоскость$(CDE)$ & коорд(прямая$(AB), K) =$ set_{xyz}(пропорцнаборы$((x+a, y+b, z+c), (d, e, f))$ & число(x) & число(y) & число(z)) & коорд(плоскость$(CDE), K) = \mathrm{set}_{uvw}(pu + qv + rw + s = 0$ & число(u) & число(v) & число$(w)) \to pd + qe + rf = 0$).

Используемый в теореме предикат "пропорцнаборы$((x + a, y + b, z + c), (d, e, f))$"означает линейную зависимость указанных в нем двух трехмерных векторов и определяет, таким образом, прямую, проходящую через точку $(-a, -b, -c)$ и имеющую направляющий вектор (d, e, f). Значением переменной K служит та аффинная система координат, в которой выписываются уравнения.

Заголовок и фильтры приема - обычные; указатель "усм(1)" определяет проверку параллельности прямой и плоскости с помощью идентифицирующих операторов - так, как они использовались в элементарной геометрии. Указатели "единица(0 a b c s)" и "единица(1 p q r)" допускают вырожденные нулевые и единичнеы значения слагаемых и коэффициентов. Указатель "заменазнака(минус p q r)" учитывет возможность знака "минус" перед коэффициентами. Указатели "подстановка(фикс(3 2 4 1 1 1)p 0)", "подстановка(фикс(3 2 4 1 1 2)q 0)" и "подстановка(фикс(3 2 4 1 1 3)r 0)" определяют возможности пропуска в уравнении плоскости слагаемых с каждой из переменных u, v, w.

Уравнение плоскости идентифицируется непосредственно с некоторым равенством в посылках, в то время как для второго антецедента, определяющего уравнение прямой , введен указатель "идентификатор(2)". Это означает, что сначала выражение коорд(прямая$(AB), K)$ обрабатывается нормализатором, определяющим некоторое уравнение прямой, а затем уже найденное уравнение идентифицируется с правой частью второго антецедента. Указатель "точкапривязки(коорд)" выбирает для инициализации срабатывания приема при сканировании символ "коорд" из уравнения плоскости.

Кроме уже упомянутого выше нормализатора "нормкоорд", в приеме используются обычные нормализаторы общей стандартизации, а также нормализатор "стандравно", упрощающий новое равенство для блокировки вывода дублирующих соотношений - в таких случаях равенство заменяется на константу "истина" и вывод не происходит.

13.1.4 Приемы вывода в условиях задачи на описание

Приемы вывода следствий в условиях задачи на описание применяются очень редко, так как они лишь увеличивают объем записей в списке условий, а этот список должен быть постепенно преобразован к виду ответа. При получении в списке условий явных выражений для неизвестных будет выполнена подстановка этих выражений в оставшиеся условия, и далее начнутся преобразования по проверке их истинности. Всякое добавление новых утверждений к списку условий будет, как правило, лишь усложнять данную проверку. Поэтому основными средствами решения задач на описание являются, во-первых, эквивалентные преобразования условий, постепенно приводящие их к виду ответа, и, во-вторых, использование логического вывода в блоке анализа задачи на описание, с сохранением указаний на взаимную выводимость получаемых утверждений. Эти указания, в случае нахождения в блоке анализа явных выражений для неизвестных, позволяют хотя бы частично исключить из списка условий задачи на описание те условия, которые являются следствиями найденных выражений для неизвестных.

В ряде случаев, однако, прием вывода настолько явно приближает задачу к ответу, что его целесообразно применить непосредственно для пополнения списка условий, не откладывая до рассмотрения посылок блока анализа. Мы приведем здесь два примера такого рода. Первый из них связан с возведением в куб уравнения с тремя кубическими радикалами. Он может быть найден в разделе ("Элементарная алгебра", "Степени", "Решение уравнений", "Уравнение с радикалами", "Возведение в куб уравнения с тремя кубическими радикалами").

Теорема приема имеет вид:

$\forall_{abcdef}(a\sqrt[3]{d}+b\sqrt[3]{e}+c\sqrt[3]{f}=0 \rightarrow a^3d+b^3e+c^3f-3abc\sqrt[3]{d}\sqrt[3]{e}\sqrt[3]{f}=0)$.

При ее получении последнее слагаемое левой части переносится в правую часть; далее обе части возводятся в куб; после группировки в левой части выделяется выражение $3ab\sqrt[3]{d}\sqrt[3]{e}(a\sqrt[3]{d}+b\sqrt[3]{e})$, в котором сумма в скобках заменяется, с учетом уравнения, на $-c\sqrt[3]{f}$.

Формально такой переход не является эквивалентным (результирующее уравнение будет, в частности, выполнено при равенстве трех слагаемых исходного уравнения, а исходное при этом может быть не выполнено). Поэтому он реализован как прием вывода следствия в условиях задачи. Заголовок приема - "выводусловия". Прием уменьшает число слагаемых с кубическими радикалами с трех до одного, что создает благоприятные предпосылки для быстрого решения нового уравнения. После его решения остается старое уравнение, в которое подставляется найденный корень, и далее полученное равенство "проверяется" - упрощается эквивалентными преобразованиями до получения логической константы "истина" либо "ложь", либо сводится к некоторому переносимому в ответ условию на параметры задачи.

Фильтры приема - простейшие: "уровень(5)" определяет срабатывание на пятом уровне; "условие", "тип(описать)" и "корень" - указывают, что рассматривается уравнение в условиях задачи на описание; "не(известно(d))", "не(известно(e))" и "не(известно(f))" - указывают, что выражения d, e, f под радикалами содержат неизвестные.

Так как прием не исключает "старого" уравнения, то для блокирования его многократных срабатываний нужно принимать специальные меры. Это делается с помощью указателя "первыйключ(3 фикс($abcdef$))". После идентификации переменных

a, b, c, d, e, f этот указатель инициирует проверку наличия комментария "3" к возводимому в куб уравнению (т.е. к условию текущей задачи на описание). Если такой комментарий имеется, то применение приема прерывается, иначе данный комментарий вводится, и последующие попытки применить прием оказываются заблокированы.

Указатели идентификации "единица(1 a b c)" и "заменазнака(минус a b c)" разрешают коэффициентам a, b, c принимать вырожденное единичное значение, а также передают этим коэффициентам знак минус перед соответствующим слагаемым, если он имеется.

Для обработки подтермов нового уравнения используются нормализаторы общей стандартизации. Кроме того, для обработки левой части нового уравнения применяется нормализатор "стандплюс", обеспечивающий раскрывание скобок и приведение подобных членов после возведения в куб.

Прием имеет текстформульный шаблон, используемый для пояснения преобразований: "Переносим слагаемое () уравнения () в правую часть и возводим обе части уравнения в куб. После этого в сумме утроенных произведений из левой части выделяем сомножитель - левую часть возводимого в куб уравнения, и заменяем его на правую часть данного уравнения. Далее все ненулевые слагаемые группируем в левой части". Указатель "титр(набор(1 терм(фикс(фикс(1 1 3))фикс(1))))" определяет подстановку в данный шаблон выражения $c\sqrt[3]{f}$ и исходного уравнения вместо скобок.

Многие приемы вывода в условиях задачи на описание служат для инициализации разбора случаев, вводя некоторую дизъюнкцию, определяющую такой разбор случаев. В качестве примера рассмотрим прием решения целочисленных уравнений либо неравенств, который при появлении неравенства $x \leq n$ для натурального выражения x с неизвестными выводит дизъюнкцию $x = 1 \vee \ldots \vee x = n$, перечисляющую все его возможные значения. Этот прием находится в разделе ("Элементарная алгебра", "Неравенства", "Меньшеилиравно", "Неравенства с целочисленной неизвестной", "Разбор случаев для выражения, принимающего натуральные значения"). Теорема приема имеет вид:

$\forall_{xn}(x \leq n$ & натуральное$(x) \rightarrow \exists_m(m \in \{1, \ldots, n\}$ & $x = m))$.

Фильтры "уровень(2)", "условие", "тип(описать)", "неизвестная(x)", "натуральное(n)" определяют общий контекст срабатывания приема. Заметим, что последний из фильтров не просто требует, чтобы выражение n имело натуральное значение, но чтобы n было десятичной записью конкретного натурального числа. Разумеется, в этом случае можно выписать конечную дизъюнкцию указанного выше вида. Однако, чтобы эта дизъюнкция не оказалась слишком большой, введен дополнительный фильтр "меньше(числзначение(n)6)", ограничивающий ее не более чем 6 членами. Этот фильтр - не более чем временная заглушка; при дальнейшем развитии приема можно будет, вероятно, выделить случаи, когда нужно выписывать и значительно большее число членов, либо создать дубликат данного приема, выписывающий длинные дизъюнкции на более высоких уровнях, когда прочих средств решения задачи не хватило.

Фильтр "контекст(условие(x1)заголовок(x1 натуральное)первыйсимвол(x1 x2)неизвестная(x2)входит(x2 x))" обеспечивает наличие принимающей натуральные значения неизвестной, встречающейся в выражении x.

Указатель "или(фикс(0)фикс(0 2 1))" нужен для того, чтобы квантор существования из теоремы приема определял при заменах дизъюнкции. Ссылка "фикс(0)" - на вхождение этого квантора общности; "фикс(0 2 1)" - на подкванторное утверждение $m \in \{1, \ldots, n\}$, определяющее перечисление значений параметра m при выписке дизъюнктивных членов. Это утверждение обрабатывается компилятором в точности так же, как программно реализуемые антецеденты.

Указатель "блокпроверок(2)" означает, что проверка "натуральности" значений выражения x выполняется проверочным оператором. Указатель "примечание(разборслучаев)" снабжает новую дизъюнкцию комментарием "разборслучаев", инициирующим немедленный разбор случаев. При его отсутствии перед разбором случаев возникала бы пауза, которая здесь могла бы привести к повторным срабатываниям того же самого приема.

13.1.5 Приемы нормализаторов

Существует множество различных типов нормализаторов, значительно отличающихся друг от друга своим назначением и архитектурой. Приведем здесь примеры приемов для основных их типов.

Нормализаторы общей стандартизации

Нормализаторы общей стандартизации вводятся практически для всех функциональных логических символов; иногда - также для предикатных символов. Они обеспечивают простейшую стандартизацию выражений, заголовком которых служат соответствующие символы. Все такие нормализаторы - корневые, то есть их приемы могут заменить только весь преобразуемый терм целиком. За редкими исключениями, нормализаторы общей стандартизации не должны вводить новых выражений, требущих специального сопровождения по области допустимых значений. При создании нового приема любого типа все новые выражения (быть может, кроме каких-то особых случаев) подвергаются обработке нормализаторами общей стандартизации, что позволяет устранять цепочки срабатываний простейших стандартизирующих приемов вслед за срабатыванием какого-либо более существенного приема. Так достигается весьма значительное ускорение преобразований.

Заголовок нормализатора общей стандартизации для логического символа F получается добавлением приставки "норм" к названию этого символа. Это соглашение упрощает ручную вставку обращений к нормализаторам общей стандартизации (хотя обычно обращения к ним можно добавлять к набранному вручную приему автоматически).

В качестве примера рассмотрим прием нормализатора общей стандартизации "нормумножение", осуществляющий перемножение степеней с одинаковыми основаниями. Путь к приему в оглавлении базы приемов - ("Элементарная алгебра", "Умножение", "Нормализатор общей стандартизации НОРМУМНОЖЕНИЕ", "Умножение степеней с одинаковыми основаниями"). Берется второй из двух представленных в этом разделе приемов. Теорема его имеет вид

$$\forall_{abc}(0 \le a \to a^b a^c = a^{b+c}).$$

Прием имеет заголовок "замена(второйтерм нормумножение)", что соответствует замене слева направо.

Указанная замена, хотя и применяется практически всегда, когда это только возможно, все же имеет некоторые ограничения. Прежде всего, здесь следует отметить использование записей вида $2\sqrt{2}$, которые обычно не преобразуются к виду $2^{3/2}$. Для блокировки такого рода замен в приеме служат фильтры "или(не(символ(b 1))не(десчисло(a))не(констдробь(c)))" и "или(не(символ(c 1))не(десчисло(a))не(констдробь(b)))". Они не разрешают проведение замены, если a есть числовая константа, один из показателей степени равен 1, а другой имеет вид простой числовой дроби.

Другое ограничение заключается в том, что при решении уравнений специально могло быть предпринято выделение в отдельный множитель степени с суммой всех неизвестных слагаемых исходного показателя (при известном основании степени). Например, это могло бы быть полезно для усмотрения квадратного уравнения относительно такого множителя. В этой ситуации необходимо заблокировать обратное преобразование, что и достигается с помощью фильтра "или(коммент(нормуравн) и(известно(b)известно(c)))". Здесь комментарий "нормуравн" указывает на наличие ранее предпринятого специального выделения множителя.

Фильтр "постпозиция(фикс(0 1 2)фикс(0 1 1))" используется для оптимизации; он отбрасывает попытки рассмотрения случаев, когда вторая степень расположена в произведении до первой, так как эти случаи избыточны.

Указатель "блокпроверок(1)"определяет обработку антецедента теоремы проверочным оператором; указатель "единица(1 b c)" разрешает вырожденные случаи, когда вместо соответствующей степени a имеется само a.

Наконец, указатель "комментарий(1 множители)" вводит комментарий "множители" при обращении к проверочному оператору, блокирующий попытку разложения a на множители - для нормализатора общей стандартизации такая попытка уже является слишком дорогостоящей.

Нормализаторы вычисления

Ряд нормализаторов общей стандартизации оказались чрезвычайно большими и превратились в нормализаторы, осуществляющие уже не предварительную стандартизацию, а символьное вычисление. Таковы, например, нормализаторы "нормпроизводная", "нормпредел" и "нормИнтграл", используемые, соответственно, для вычисления производных, пределов и неопределенных интегралов. Хотя они и имеют такие же названия, как прочие нормализаторы общей стандартизации, их формат пополнен рядом существенных элементов. В первую очередь, это указатели "контрольнормализации" и "коррекцияпосылок". Первый из них означает, что результаты обращений к нормализатору будут сохраняться в специальном буфере, и при каждом новом обращении сначала будет предприниматься попытка извлечь готовый результат из буфера. Преобразования нормализаторов, формат которых имеет такой указатель, отображаются на экране при пошаговом просмотре решения. Второй указатель означает, что нормализатор может пополнять свой список посылок, и новые посылки будут переданы внешней задаче.

Будем рассматривать прием нормализатора "нормпроизводная", выполняющий дифференцирование выражения степенного вида, у которого и основание, и показатель

зависят от переменной дифференцирования. Этот прием находится в разделе ("Математический анализ", "Производная", "Нормализатор НОРМПРОИЗВОДНАЯ", "Производная степени"). Он является последним в группе приемов данного раздела.

Теорема приема имеет вид:

$$\forall_{abfgx}(a = \frac{df(x)}{dx} \ \& \ \text{число}(a) \ \& \ b = \frac{dg(x)}{dx} \ \& \ \text{число}(b) \rightarrow \frac{d((f(x))^{g(x)})}{dx} = (f(x))^{g(x)} \cdot (b \ln f(x) + \frac{g(x)a}{f(x)})).$$

В ее первом и третьем антецедентах вычисляются производные основания и показателя; второй и четвертый антецеденты проверяют, что производные удалось вычислить - лишь после этого можно усмотреть, что полученные выражения имеют числовые значения.

Заголовок приема - "замена(второйтерм нормпроизводная)". Фильтров прием не имеет, если не считать формально отнесенного к указателям условия "уровень(4)" на уровень его срабатывания. Кроме данного приема, имеется еще несколько приемов для дифференцирования степени в частных случаях (константное основание либо константный показатель), уровни срабатывания которых меньше 4.

Указатели "идентификатор(1 3)" и "блокпроверок(2 4)" определяют уже объясненное выше использование антецедентов. Указатель "отображение(f g)" означает, что функциональные переменные $f(x), g(x)$ идентифицируются с произвольными выражениями.

Указатель "вывод(условие(коммент(нормпредел))меньше(0 значение(f x)))" определяет занесение новой посылки о положительности основания степени, необходимой для сопровождения по о.д.з. из-за появившегося логарифма. Эта посылка впоследствии будет передана во внешнюю задачу.

Нормализаторы "нормпроизводная", которыми сопровождаются правые части равенств в антецедентах, суть рекурсивные обращения к тому же самому пакету для вычисления соответствующих производных. Подтермы заменяющего терма снабжены обращениями к обычным нормализаторам общей стандартизации. Кроме того, новая посылка обрабатывается нормализатором общей стандартизации строгих неравенств и нормализатором, обеспечивающим стандартное упорядочение операндов коммутативных операций.

Следующий пример - прием нормализатора "нормпредел", вычисляющий предел дроби по правилу Лопиталя. Нормализатор "номпредел" представляет собой пакетный оператор, значительно более крупный, чем оператор "нормпроизводная". Он насчитывает более 200 приемов и имеет 9 уровней срабатывания.

Среди указателей формата этого оператора имеется элемент "разборслучаев", который позволяет при вычислении предела инициировать разбор случаев для параметров рассматриваемого выражения, если в разных ситуациях предел вычисляется по-разному. Результаты разбора отдельных подслучаев потом объединяются в одно условное выражение. При принятии решения о разборе случаев происходит откат к рассмотрению исходного выражения (быть может, с подстановкой в него значений тех параметров, которые фиксированы в текущем подслучае). Новые случаи относятся к последнему выделенному для рассмотрения надслучаю. При откате сохраняются только результаты рассмотрения ранее разбиравшихся подслучаев и схема этих

подслучаев, а весь промежуток вычислений от начала рассмотрения надслучая до инициализации разбора случаев пропадает. На первый взгляд, это может показаться не очень экономным способом вычислений. Однако, так как все результаты последних обращений к пакетным операторам сохраняются в буферах, то при повторе они просто берутся оттуда в готовом виде, и существенных потерь не происходит.

Прием для правила Лопиталя расположен в разделе ("Математический анализ", "Предел", "Нормализатор НОРМПРЕДЕЛ", "Предел дроби", "Правило Лопиталя"). Теорема этого приема имеет следующий вид:

$$\forall_{fgabcdehui}(d = \lim_{y\to a\backslash b} f(y)\ \&\ e = \lim_{y\to a\backslash b} g(y)\ \&\ (d = 0\ \&\ e = 0 \vee (d = \infty \vee d = -\infty)\ \&\ (e = \infty \vee e = -\infty))\ \&\ h = \lambda_y(\tfrac{df(y)}{dy}, \text{число}(y))\ \&\ u = \lambda_y(\tfrac{dg(y)}{dy}, \text{число}(y))\ \&\ c = \lim_{y\to a\backslash b} \tfrac{h(y)}{u(y)}\ \&\ (\text{число}(c) \vee c = \infty \vee c = -\infty)\ \&\ i = c \to \lim_{y\to a\backslash b} \tfrac{f(y)}{g(y)} = i)$$

Первые два антецедента вычисляют пределы числителя и знаменателя дроби; третий проверяет, что либо оба эти пределы равны 0, либо бесконечны. Четвертый и пятый антецеденты вычисляют производные; шестой - находит предел отношения производных; седьмой - проверяет, что предел действительно найден - либо конечен, либо бесконечен. Наконец, восьмой антецедент нужен для обращения к вспомогательной задаче на упрощение найденного для предела выражения.

Запись $y \to a\backslash b$ означает, что y стремится к a, причем b - тип стремления (двусторонний, слева либо справа). Это - общая форма указания типа предела для формульной записи. В конкретных случаях вместо нее используются обычные $y \to a$, $y \to a + 0$, $y \to a - 0$. Указатель b при двустороннем пределе равен 0, при пределе слева - 1, и при пределе справа - 2. Однако, такое кодирование используется лишь на уровне внутреннего, скобочного представления термов.

Заголовок приема - "замена(второйтерм нормпредел)". Фильтры его, в основном, эвристические, и предназначены для блокировки применения в особо сомнительных ситуациях. Первым идет фильтр "не(контекст(операнд(фикс(0 1 1 3) x11) не(длинаменее(x11 конъюнктчлен))))", который отсекает случаи чрезмерно длинных (более 80 символов) числителя либо знаменателя. Некоторым оправданием этому служит то, что часто дифференцирование приводит к дальнейшему усложнению выражений. Заметим, что число 80 представлено в фильтре как символьная константа "конъюнктчлен". Использование символьных констант упрощает компиляцию, но несколько усложняет ручную работу с приемом; по мере усовершенствования интерфейса редактора приемов предполагается его устранить. Пока этого не сделано, для того, чтобы найти нужную символьную константу для большого числа либо определить число, которое она представляет, приходится пользоваться интерфейсом "Ресурсы и установки" из главного меню. Если есть символьная константа, то находится номер ее логического символа и вычитается 260; если есть число, то к нему прибавляется 260, и находится логический символ с данным номером.

Следующий фильтр - "не(контекст(операнд(фикс(0 1 1 3)x11)вид(x11 умножение(x12 степень(x10 x13)))входит(y x10)не(константа(x13))единица(1 x12)контекст(равно(x14 терм(предел(отображение(y число(y)x10)x2 x1)))заголовок(x14 0 плюсбеск минусбеск))))". Он проверяет наличие сомножителя числителя либо знаменателя, имеющего вид степени с неконстантным показателем x13 и содержащим переменную y основанием x10. Фильтр предпринимает попытку вычислить предел x14 этого основания (используя нормализатор "нормпредел", применяемый к соответствующему

выражению внутри данного фильтра). Если предел оказался равен 0 либо бесконечности, то применение приема блокируется. Эвристической мотивировкой фильтра является то, что дифференцирование не устраняет наличия множителя указанного вида, причем этот множитель влияет на стремление числителя либо знаменателя к 0 или к бесконечности.

Дальше идет фильтр "не(контекст(позиция(x10 корень)символ(x10 факториал) входит(y x10)))". Он отменяет заведомо бессмысленную попытку дифференцирования, если выражение содержит целочисленную функцию "факториал", зависящую от y.

Следующий фильтр служит для предотвращения многократных последовательных дифференцирований. Он имеет вид "контекст(равно(x13 число(комментарий(x11)заголовок(x11 частнпроизв)))или(меньше(x13 2)контекст(операнд(фикс(0 1 1 3) x11)вид(x11 плюс(x12 умножение(x10 y)))единица(0 x12)единица(1 x10)заменазнака(минус x10)не(входит(y x10)))и(меньше(x13 3)не(контекст(операнд(фикс(0 1 1 3)x11) не(длинаменее(x11 соединение))))))не(контекст(тригаргумент(корень x15) подчинено(x15 x16)контекст(вид(x16 степень(x17 x18)) альтернатива(меньше(x13 2) не(константа(x18)) не(целое(x18))))входит(y x15))))". При каждом обращении к оператору "нормпредел" с целью вычислить предел отношения производных вводится новый экземпляр комментария "частнпроизв", которым снабжается это обращение. Таким образом, число этих комментариев равно числу внешних применений правила Лопиталя. Фильтр находит данное число x13 и проверяет выполнение хотя бы одного из следующих условий:

а) x13 менее 2;

б) Числитель либо знаменатель имеет слагаемое, линейное по y, и тогда x13 игнорируется;

в) x13 менее 3, причем числитель и знаменатель имеют каждый не более 40 символов.

Если одно из этих условий выполнено, то далее фильтр проверяет отсутствие тригонометрической функции от выражения с вхождением y, расположенной внутри степенного выражения, у которого при x13 большем 1 показатель степени не является десятичной записью целого числа, а при x13 меньшем или равном 1 - не является константой.

Разумеется, перечисленные условия не являются неоспоримыми и не претендуют на полноту - просто они возникли на тех примерах, которые до сих пор были рассмотрены. При продолжении обучения решателя эвристический образ "перспективной для применения правила Лопиталя ситуации", вероятно, будут развиваться дальше.

Фильтр "коммент(производная)" проверяет отсутствие комментария "производная", который предназначен специально для отмены применения правила Лопиталя и вводится там, где нужна ускоренная попытка вычисления предела. Фильтр "или(длинаменее(значение(h y)конецприставки) длинаменее(значение(uy)конецприставки))" проверяет, что хотя бы одна из производных числителя и знаменателя имеет не более 70 символов в своей записи. Наконец, фильтр "или(коммент(целое)не(контекст(позиция(x10 корень)вид(x10 степень(x11 x12))входит(y x12))))" проверяет, что при наличии комментария "целое" выражение под пределом не имеет степенных подвыражений с показателем, зависящим от y. Комментарий "целое" вводится приемом, который предпринимает попытку сведения вычисления предела последовательности к вычислению предела числовой функции; вероятность того, что правило

Лопиталя здесь окажется полезным в ситуациях с комбинаторного типа "показательными" подвыражениями, невелика.

Указатель "уровень(5)" определяет пятый уровень срабатывания приема (из возможных девяти). Указатель "идентификатор(1 2 4 5 6 8)" выделяет те равенства в антецедентах, которые обрабатываются компилятором как присвоения значений соответствующим переменным. Указатель "блокпроверок(3 7)" определяет использование проверочных операторов для антецедентов, анализирующих промежуточные результаты c, d, e вычисления пределов. Заметим, что эти антецеденты построены из элементарных утверждений с помощью конъюнкций и дизъюнкций. Такие ситуации поддерживаются компилятором, обеспечивающим необходимую логику обращений к операторам, проверяющим отдельные элементарные утверждения. В тех случаях, когда утверждение имеет вид равенства переменной X константе A, компилятор вместо обращения к проверочному оператору просто вставляет оператор ЛОСа, распознающий наличие заголовка A у терма, идентифицированного с X.

Указатель "отображение(f g h u)" определяет возможность идентификации функциональных переменных $f(y)$, $g(y)$, $h(y)$, $u(y)$ с произвольными выражениями.

Указатель "лимит(вариант(контекст(список(x10 значение(f y) значение(g y)) вид(x10 степень(y x11)) единица(1 x11)натуральное(x11)) 7000000 1500000)))" определяет органичение числа шагов интерпретатора, отводимых на попытку применения приема. Если оно оказывается превышено, попытка обрывается. Лимит шагов определяется условным выражением, учитывающим контекст: если числитель либо знаменатель представляет собой степень переменной y, имеющую константный натуральный показатель, то он берется значительно большим.

Указатель "титр(вариант(заголовок(d 0)набор(1)набор(2)))" нужен для выбора текстового пояснения, соответствующего конкретной ситуации. Текстформульный шаблон для создания сопровождающих пояснений имеет у данного приема следующий вид:

"¬1 Так как пределы числителя и знаменателя равны 0, применяем правило Лопиталя ¬2 Так как пределы числителя и знаменателя бесконечны, применяем правило Лопиталя ¬3 Производная числителя ¬4 Производная знаменателя ¬5 Предел отношения производных".

В этом шаблоне первые два фрагмента (начинающиеся с ¬1, ¬2) дают общий поясняющий текст. Указатель "титр(...)" определяет выбор первого из них при $d = 0$ и второго - при бесконечном d. Остальные фрагменты шаблона нужны для пояснения обращений к нормализаторам (см. ниже).

Среди нормализаторов приема выделим, прежде всего, рекурсивные обращения к вычислению пределов числителя и знаменателя. Они имеют вид "нормпредел(замечание(производная) замечание(титр(стоп)))". Запись "титр(стоп)" определяет комментарий к нормализатору, означающий, что обращение к нему вообще не будет выводиться на экран при просмотре срабатывания приема.

Обе производные снабжены нормализаторами "номпроизводная(замечание(нормпредел) замечание(титр(набор(i))))", где i есть 3 для первой производной и 4 для второй. Запись "титр(набор(i))" определяет комментарий к нормализатору, означающий, что при обращении к нему на экран будет выводиться поясняющий фрагмент текстформульного шаблона, имеющий номер i.

Отношение производных обрабатывается нормализатором общей стандартизации "нормдробь", к посылкам которого добавляется утверждение $y \to a\backslash b$. Оно может оказаться полезным при сокращении дроби - для усмотрения отличия множителя от нуля. Предел отношения производных обрабатывается нормализатором "нормпредел(примечание(частнпроизв)замечание(титр(набор(5))))", где комментарий "частнпроизв", как уже говорилось выше, играет роль счетчика числа вложенных применений правила Лопиталя, а запись "титр(набор(5))" извлекает из текстформульного шаблона пятый фрагмент для сопровождения обращения к нормализатору.

Наконец, переменная i в последнем антецеденте обрабатывается нормализатором "задача(5 упростить одз цель(условие(длинатекста(списокпеременных(теквхожд)1) ответ)))". Это - обращение к вспомогательной задаче на преобразование, решаемой до 5-го уровня. Если выражение под пределом не имело параметров, то упрощение в рамках приема считается излишним - тогда задача сопровождается целью "ответ", инициирующей немедленную выдачу ответа без каких-либо преобразований.

Для нормализатора "нормпредел" рассмотрим также пример приема, инициирующего разбор случаев. Такой разбор случаев может понадобиться, если дробное выражение зависит от параметров и априори не усматривается отличие предела его знаменателя от 0. Соответствующий прием находится в подразделе "Разбор случаев по равенству нулю предела знаменателя" того же раздела, где был расположен подраздел "Правило Лопиталя". Теорема приема имеет вид:

$\forall_{axfbcd}(a = \lim_{x\to c\backslash b} f(x)$ & число(a) & $d = \lim_{x\to c\backslash b} g(x)$ & число$(d) \to \lim_{x\to c\backslash b} \frac{f(x)}{g(x)} =$ разборслучаев$(d = 0 \vee \neg(d = 0)))$.

Разумеется, это еще один пример псевдотеоремы; равенство предела условной записи "разборслучаев(...)" является лишь удобным для компиляции техническим трюком. Впрочем, этот трюк выходит за рамки одного лишь пакета "нормпредел" и допустим в любом пакетном нормализаторе, описание формата которого снабжено элементом "разборслучаев".

Прием имеет заголовок "замена(второйтерм нормпредел)" - так, как будто он является обычным приемом замены. Указателем на то, что вместо замены происходит инициализация разбора случаев, служит правая часть "разборслучаев(...)" равенства в теореме приема - никаких других указателей для этого не требуется.

Фильтры "не(заголовок(d 0))", "не(константа(d))", "конец(не(легковидеть(не(равно(d 0)))))" отсекают ситуации, когда предел знаменателя не содержит параметров либо очевидно ненулевой. Заметим, что проверка последнего условия, сравнительно трудоемкая, проводится после идентификации остальных элементов ситуации - для этого соответствующий фильтр снабжен пометкой "конец(...)".

Указатель "уровень(3)" фиксирует уровень срабатывания приема; "идентификатор(1 3)" - определяет использование равенств в антецедентах для вычисления пределов числителя и знаменателя. Указатель "блокпроверок(2 4)" определяет использование проверочного оператора для усмотрения того, что пределы удалось вычислить. Наконец, указатель "отображение(f g)" разрешает идентификацию функциональных переменных $f(x), g(x)$ с произвольными выражениями.

Правые части равенств в антецедентах обрабатываются нормализаторами "нормпредел". Условие равенства предела знаменателя нулю обрабатывается простейшим

нормализаторами числовых равенств "нормчисло". Заметим, что нецелесообразно применять нормализатор общей стандартизации логических связок "нормлог" к вводимой для разбора случаев дизъюнкции, так как она, очевидно, тавтологична и будет им заменена на константу "истина".

Следующий пример относится к нормализатору "нормИнтеграл", обеспечивающему вычисление неопределенных интегралов. Он имеет несколько более простой формат, чем нормализатор "нормпредел", так как не предполагает режима разбора случаев. Число уровней срабатывания этого нормализатора равно 8, а число приемов - около полутора сотен, т.е. меньше, чем у нормализатора вычисления пределов. Однако, сами программы приемов здесь более громоздки, так как используется множество стандартных формул и подстановок. На этом материале преимущество программирования на ГЕНОЛОГе особенно наглядно - вместо набора вручную программы на ЛОСе, занимающей несколько полных "экранов", с помощью формульного редактора рисуется средней сложности формула с интегралом, сопровождаемая, как правило, весьма небольшим списком фильтров и указателей.

Рассмотрим сначала пример приема, основанного на табличном интеграле для дроби, числитель и знаменатель которой суть линейные комбинации синуса и косинуса. Этот прием находится в разделе ("Математический анализ", "Интегралы", "Нормализатор нахождения первообразной нормИнтеграл", "Тригонометрические функции", "Дроби с линейными комбинациями синуса и косинуса"). Прием является третьим от начала в списке приемов данного раздела. Теорема его имеет вид:

$$\forall_{abcde}(e = d^2+c^2 \to \int \frac{a\sin x + b\cos x}{c\sin x + d\cos x}dx = \lambda_x(\frac{(bd+ac)x + (bc-ad)\ln|c\sin x + d\cos x|}{e}$$

,число$(x)))$

Фильтров у приема нет; указатель "идентификатор(1)" определяет вычисление в первом антецеденте вспомогательного выражения e - суммы квадратов коэффициентов знаменателя. Указатели "единица(1 a b c d)" и "заменазнака(минус a b c d)" определяют возможность вырожденных коэффициентов и наличия минуса перед коэффициентом. Указатели "подстановка(фикс(0 1 1 3 1 1)a 0)" и "подстановка(фикс (0 1 1 3 1 2)b 0)" определяют возможность отсутствия в числителе слагаемого с синусом либо косинусом. Наконец, указатель "пересечениесписков(фикс(0 1 1 3 1))" нужен для того, чтобы обратить внимание компилятора на возможное отличие заголовка числителя подынтегрального выражения от символа "плюс" - иначе компилятор потребует, чтобы числитель обязательно был суммой, и этим отсечет указанные выше вырожденные случаи отсутствия одного из слагаемых.

Кроме нормализаторов общей стандартизации, расставляемых, по существу, автоматически, можно выделить нормализатор "стандинтеграл(замечание(переменная x))", которым обрабатывается результирующее выражение для первообразной. Этот нормализатор возник для предварительного упрощения результатов интегрирования - чтобы учесть специфику возникающих здесь типичных подвыражений, а также чтобы добиться ускорения по сравнению с решением вспомогательной задачи на преобразование. Постепенно развиваясь, он достиг сейчас достаточно больших размеров - свыше 100 приемов, некоторые из которых попросту дублируют стандартные общие приемы упрощения, а некоторые ориентированы на частные случаи, появление которых при интегрировании представляется сравнительно вероятным. Обращение

к нормализатору должно сопровождаться вводом комментария (переменная x), указывающим на переменную интегрирования.

Если сравнить объем программы приема на ЛОСе (перейти к ней можно нажатием Home, а вернуться - нажатием End) и на ГЕНОЛОГе, то видно, что первый в 4-5 раз больше второго. При этом запись на ГЕНОЛОГе существенно нагляднее.

Следующий пример приема нормализатора "нормИнтеграл" - простейшая логарифмическая замена переменной интегрирования. Прием расположен в подразделе ("Замена переменной интегрирования", "Логарифмическая функция", "Простейшая замена") корневого раздела нормализатора; берется последний прием списка приемов данного подраздела. Усматривая в знаменателе подынтегрального выражения некоторый множитель $f(x)$, прием пытается перейти к новой переменной $y = \ln f(x)$. Теорема его имеет вид:

$$\forall_{fghuv}(\lambda_x(\frac{g(x)}{h(x)\frac{df(x)}{dx}}, \text{число}(x)) = \lambda_x(u(\ln f(x)), \text{число}(x)) \ \& \ \int u(x)dx = \lambda_x(v(x),$$

$$\text{число}(x)) \rightarrow \int \frac{g(x)}{f(x)h(x)}dx = \lambda_x(v(\ln f(x)), \text{число}(x))).$$

Смысл первого равенства в антецедентах - убедиться в том, что выражение, остающееся после сокращения подынтегрального выражения на производную заменяющей функции $\ln f(x)$, можно выразить только через эту функцию, как некоторое $u(\ln f(x))$. Далее в теореме приема предпринимается экономия на вводе новой переменной y, и интеграл для $u(y)$ записывается как интеграл для $u(x)$. В возникающую после интегрирования функцию $v(x)$ подставляется обратно $\ln f(x)$.

Фильтр "контекст(подчинено(x1 фикс(0 1 1 3)) символ(x1 логарифм) контекст(подчинено(x2 x1) равно(x2 значение(f x))))" определяет проверку того, что подынтегральное выражение содержит логарифм, внутри которого имеется вхождение выражения, идентифицированного с $f(x)$. Эта проверка предшествует обработке первого антецедента и служит для отсечения случаев, в которых он почти наверняка не будет реализован.

Первый и второй антецеденты выделены указателями "идентификатор(1 2)" - у каждого из них сначала формируется левая часть, а затем происходит идентификация ее с правой частью. Левая часть первого антецедента теоремы имеет вид $\lambda_x(A(x),$ число$(x))$, где $A(x)$ - результат обработки указанного в теореме выражения нормализаторами "задача(5 упростить одз)", "посылки(число(x))". Указатель "новаргумент(u x извлечение)" определяет применение к выражению $A(x)$, перед идентификацией его с $u(\ln f(x))$, нормализатора "извлечение". Этот нормализатор пытается преобразовать $A(x)$ к виду $B(x)$, в котором переменная x встречается только внутри подвыражений $\ln f(x)$. Если это удается, то далее $u(t)$ понимается компилятором как результат замены всех таких подвыражений в $B(x)$ на терм t.

Левая часть второго антецедента обрабатывается нормализатором "нормИнтеграл" - предпринимается попытка вычисления интеграла после замены переменной. Результирующее выражение $v(\ln f(x))$ упрощается с помощью нормализатора "станд-интеграл".

Нормализаторы усмотрения зависимости от заданного выражения

Приемы замены переменной при интегрировании либо при решении дифференциальных уравнений часто связаны с усмотрением возможности представить некото-

рое выражение A в таком виде, чтобы все вхождения заданной переменной x в A были расположены только внутри вхождений заданного подвыражения t. Не всегда изначально A представлено в таком виде явно, и тогда предварительно приходится выполнять преобразования, приводящие его к данному виду. Здесь используются специальные нормализаторы, наиболее развитым из которых является нормализатор "извлечение". Кроме преобразуемого терма, нормализатору должны быть переданы также x и t. Это делается с помощью вводимого при обращении комментария (новаргумент t x), у которого x представлено в формате терма.

Рассмотрим простой прием нормализатора "извлечение", выражающий четную степень косинуса через тангенс. Этот прием расположен в разделе ("Математический анализ", "Общие свойства числовых функций", "Нормализатор ИЗВЛЕЧЕНИЕ", "Выражение через тангенс"); он является четвертым от конца списка приемов этого раздела. Теорема приема имеет вид:

$$\forall_{ab}((\cos a)^{2b} = \frac{1}{((\mathrm{tg}\, a)^2 + 1)^b}).$$

Заголовок приема - "замена(второйтерм извлечение)"; фильтров прием не имеет. Указатель "единица(1 b)" разрешает принимать параметру b вырожденное единичное значение. Указатель "вход(новаргумент c d)" определяет дополнительную идентификацию, состоящую в том, что находится комментарий (новаргумент A B), и переменные c, d идентифицируются, соответственно, с A и B. Указатель "контекст(вид(c тангенс(a)))" также определяет дополнительную идентификацию; если заголовком выражения, идентифицированного с c, является "тангенс", то переменная a идентифицируется с аргументом тангенса. После того, как прием убедился, что требуется переходить к тангенсам заданного аргумента, и идентифицировал вхождение четной степени косинуса этого аргумента, он применяет указанное выше тождество. В его заменяющей части лишь знаменатель обрабатывается нормализатором общей стандартизации "нормстепень". Вообще, обработка нормализаторами заменяющих термов в случаях перехода к специальным подвыражениям должна предприниматься осторожно, чтобы эти подвыражения после нее не пропали.

Нормализаторы приведения к заданным заголовкам

В некоторых случаях нормализатор бывает нужен для преобразования терма к виду, у которого заголовок или примыкающая к нему корневая часть терма удовлетворяют специальному условию. Типичный пример - разложение выражения на множители, когда требуется получить один из заголовков "умножение", "степень", "дробь", быть может, после отбрасывания корневого минуса. Формально, такое преобразование всегда можно ввести фиктивными средствами (например, умножив выражение на единицу). Однако, имеются определенные эвристические ограничения на разумные средства, применяемые при переходе к заданным заголовкам. По-видимому, их можно выделить из анализа примеров и объяснить генератору приемов, который самостоятельно отбирал бы тождества для создания новых приемов таких нормализаторов. Пока же, при обучении решателя вручную, например, тому же разложению на множители, подобные вопросы не возникают, так как каждому "очевидно", какого рода переходы к произведениям, дробям и степеням здесь полезны, а какие - бессмысленны.

Приведем несколько примеров приемов нормализатора разложения на множители. Этот нормализатор получил название "видумножение", оставшееся за ним после серий экспериментов по автоматическому комплектованию нормализаторов преобразования к заданным заголовкам. Список заголовков в них определялся по имеющимся теоремам для декомпозиции уравнений с заданным заголовкам одной из своих частей (для равенств нулю произведения, дроби либо степени такие теоремы как раз имеются). Нормализатор имеет 6 уровней срабатывания; в описание его формата входят упоминавшиеся выше указатели "контрольнормализации" и "коррекцияпосылок". В частности, они предопределяют сохранение результатов обращений к нормализатору (как удачных, так и неудачных) в специальном буфере, и повторные попытки разложения на множители одного и того же выражения в идущих подряд попытках применения различных приемов оказываются почти бесплатными. Это - очень большой нормализатор; число его приемов более 320. Почти все тригонометрические тождества при обучении оказались отнесенными именно к нему.

В качестве первого примера рассмотрим разложение на множители по формуле суммы кубов. Прием расположен в разделе ("Элементарная алгебра", "Умножение", "Нормализатор разложения на множители ВИДУМНОЖЕНИЕ", "Тождества для непосредственного разложения на множители", "Сумма кубов"). Берется второй из двух приемов данного раздела. Теорема приема имеет вид:

$$\forall_{ab}(a^3 + b^3 = (a + b)(a^2 - ab + b^2)).$$

Прием имеет заголовок "замена(второйтерм видумножение)", указывающий на замену слева направо.

Так как заменяемая часть симметрична, используется фильтр, отсекающий половину попыток применения приема: рассматриваются только случаи, когда выражение a лексикографически предшествует выражению b. Этот фильтр имеет вид "лексикопредшествует(фикс(0 1 1)фикс(0 1 2))".

Фильтр "коммент(числоценка)" блокирует применение приема в ситуациях, когда разложение на множители предпринято для упрощения неравенства с нулем в одной из частей: здесь целесообразнее перенести один из кубов в противоположную часть неравенства и извлечь кубический корень. Комментарий "числоценка" создается тем приемом, который предпринял указанную попытку упрощения неравенства.

Фильтр "или(входит(нормИнтеграл комментарии) не(контекст(комментарий(нормИнтеграл x3) тригаргумент(корень x4)входит(x3 x4))))" блокирует применение приема при предварительном преобразовании подынтегрального выражения, если сумма кубов содержит тригонометрическую операцию от аргумента, содержащего переменную интегрирования. Это - эвристическое соображение, основанное на том, что обычно при интегрировании степени тригонометрических функций преобразуются к функциям кратного аргумента. Другие приемы разложения на множители в таких ситуациях оказались сравнительно безвредны, и лишь приемы для суммы и разности кубов начали приводить к нежелательным последствиям. Далее расположено еще несколько аналогичных эвристических фильтров, блокирующих применение данного приема в тех специальных случаях, когда водимое им довольно ощутимое усложение выражения нежелательно. Все эти фильтры были подсказаны совсем небольшим числом примеров, имеют достаточно грубый характер, и при продолжении обучения возможен цикл их уточнений.

Надо сказать, что любой сколь-нибудь сомнительный или непонятный фильтр легко можно проанализировать заново, определив все те задачи, в которых он блокирует срабатывание приема, или в которых прием срабатывает несмотря на этот фильтр. Для этого применяется прокрутка по разделам задачника, в которых возможно применение приема, после чего возникает исчерпывающая информация, на основании которой можно отбросить фильтр, скорректировать его либо вообще предложить другую схему решения рассматриваемых задач. Таким образом, несмотря на огромное число приемов в решателе, он вполне допускает локальную оптимизацию, и изменения любого элемента любого приема обычно легко контролируются, не приводя к разрушению регулировки базы приемов в целом. В приводимых ниже упражнениях мы проиллюстрируем технику анализа ранее созданных эвристических фильтров.

Указатель "уровень(4)" определяет размещение программы приема в группе приемов данного нормализатора, срабатывающих на четвертом уровне. Указатель "знак-суммы(минус фикс(0 1))" вводит попытку одновременного изменения знака всех слагаемых при применении данного приема. Указатель "модификатор" вводит проверку того, что преобразуемая сумма не имеет других слагаемых - иначе вместо разложения на множители произошла бы попытка группировки, для которой предусмотрен другой прием.

Указатель "замечание(6 фикс(0 2 2))" вводит комментарий к данному нормализатору, означающий, что второй множитель заменяющего терма не должен подвергаться попыткам разложения на множители по формуле квадратного трехчлена (дискриминант неположителен). Прием нормализатора, осуществляющий разложение по формуле квадратного трехчлена, имеет фильтр, учитывающий этот комментарий.

Указатель "замечание(условие(и(входит(нормчислитель комментарии) контекст (тригаргумент(корень x4)))) фильтрудвоения)" вводит комментарий "фильтрудвоения", если преобразуемое выражение содержит тригонометрическую операцию, причем попытка разложения на множители относится к числителю дроби, возникшей после сложения двух дробей. Этот комментарий блокирует попытки перехода к двойному тригонометрическому аргументу, так как в данной ситуации они могут привести к нежелательным последствиям, например, измененить множитель числителя, имеющийся в знаменателе, и сделать невозможным сокращение дроби. При необходимости переход к двойному тригонометрическому аргументу будет выполняться после выхода из рассматриваемого нормализатора.

Расстановка нормализаторов в приеме стандартная; в ряде случаев применение нормализатора общей стандартизации сопровождается применением нормализатора, упрощающего выражение относительно неизвестных: могут быть приведены подобные члены с известными коэффициентами либо выполнена группировка сомножителей под общий неизвестный показатель степени - действия, ускоряющие последующие рассмотрения уравнений. Если обращение к разложению на множители произошло не из задачи с неизвестными, то такой нормализатор ничего не изменяет.

Следующий пример - разложение на множители путем попыток группировки пар слагаемых. Имеется множество приемов такого типа, двойственных приемам непосредственного разложения на множители. Рассмотрим один из них - группировку при усмотрении разности квадратов. Прием расположен в подразделе ("Частичные группировки", "Разность квадратов") корневого раздела нормализатора "видумножение". Теорема его имеет вид:

$$\forall_{abcde}(e = a(b-c)(b+c)+d \to ab^2 - ac^2 + d = e).$$

Выглядит она несколько искусственно, так как сочетает в себе два преобразования - переход от $ab^2 - ac^2$ к $a(b-c)(b+c)$ и разложение на множители суммы, полученной после данной группировки. Это разложение (с учетом указателей нормализации) выполняется первым антецедентом; если результат e действительно имеет вид произведения, дроби либо степени, то прием заменяет на него преобразуемое выражение. Таким образом, прием прибегает к рекурсии, и по завершении ее сразу выдает результат разложения на множители.

Заголовок приема - "замена(второйтерм видумножение)". Фильтр "или(заголовок(e умножение степень дробь) и(заголовок(e минус) первыйсимвол(e умножение степень дробь)))" проверяет, что выражение e - результат разложения на множители после группировки - действительно имеет требуемый вид "обобщенного произведения".

Фильтр "или(коммент(группировка) контекст(вид(d плюс(х6 умножение(х7 степень(плюс(b минус(c))х8)))) единица(1 х7 х8) заменазнака(минус х7) единица(0 х6)) контекст(вид(d плюс(х6 умножение(х7 степень(плюс(b c) х8)))) единица(1 х7 х8) заменазнака(минус х7) единица(0 х6)))" регулирует применение приема после того, как уже были предприняты какие-то другие группировки. Заметим, что при обращении к разложению на множители после группировки, выполняемое в первом антецеденте, сопровождается комментарием "группировка". При наличии такого комментария рассматриваемый фильтр проверяет, что некоторое слагаемое преобразуемого выражения уже имеет своим множителем степень суммы либо разности b и c - чтобы данная группировка продолжала начатую попытку выделения сквозного общего множителя.

Фильтр "или(не(коммент(группировка)) не(меньше(количествооперандов(теквхожд) 4)))" проверяет, в случае первой группировки пары слагаемых, что общее число слагаемых не менее 4 - для малого числа слагаемых разложение на множители обеспечивается другими приемами, без предварительных группировок.

Фильтр "меньше(количествооперандов(теквхожд)8)" введен для блокировки замедляющих действия решателя попыток разложения на множители слишком длинных сумм путем группировок. Фильтр "не(константа(корень))" блокирует применение группировок при разложении на множители константных выражений. Фильтр "коммент(нормуравнение)" проверяет отсутствие комментария "нормуравнение" - этот комментарий используется для указания на ослабленный режим разложения.

Фильтр "не(контекст(вид(d плюс(дробь(х6 х7)х8))единица(0 х8)заменазнака(минус х6)))" проверяет, что сумма не имеет дробных слагаемых - в этом случае речь идет о сложении дробей, а не о разложении на множители, и применяется другой прием нормализатора.

Фильтр "или(не(константа(b))не(константа(c)))" блокирует группировку для подобных членов ab^2, ac^2, если b, c оказались константами.

Фильтры "не(контекст(вид(b степень(х6 х7)) не(известно(х7))))" и "не(контекст(вид(c степень(х6 х7)) не(известно(х7))))" блокируют группировки при разложении на множители разности частей показательного уравнения, так как обычно в этих случаях показательное выражение заменяется на новую неизвестную.

Наконец, фильтр "коммент(тригонометрия)" проверяет отсутствие комментария "тригонометрия", вводимого нормализатором после того, как было принято решение о попытке разложения на множители путем перехода от степеней тригонометрических функций к функциям кратного аргумента.

Указатель "уровень(4)" откладывает применение приема до четвертого уровня - на случай, если имеется более простой способ разложения, чем группировки. Указатель "идентификатор(1)" объясняет, что первый антецедент нужен для ввода вспомогательного выражения e; указатель "единица(1 a)" - что возможно вырожденное единичное значение коэффициента a.

Кроме обычных нормализаторов общей стандартизации, в приеме применяется обработка правой части первого антецедента нормализатором "видумножение(замечание(группировка) замечание(титр(набор(2))))". Организация текстформульного сопровождения срабатывания приема с помощью указателя "титр(...)" здесь аналогична ранее разобранной.

Для разложения на множители многочленов с целыми коэффициентами, степень которых не превосходит 5, нормализатор "видумножение" применяет (если неприменимы более простые средства) метод неопределенных коэффициентов. В качестве примера рассмотрим прием для разложения многочлена степени 3 на линейный и квадратичный множители. Этот прием размещен в подразделе ("Разложение многочленов путем подбора целочисленных коэффициентов", "Представление многочлена третьей степени в виде произведения линейного множителя и квадратного трехчлена") корневого раздела нормализатора. Берется второй из двух имеющихся в разделе приемов. Теорема приема имеет вид:

$\forall_{abcdefghijk}(fh = d \,\&\, gj = a \,\&\, 0 < g \,\&\, gi = b - fj \,\&\, gh + fi = c \rightarrow ae^3 + be^2k + cek^2 + dk^3 = (fk + ge)(hk^2 + iek + je^2))$.

Она возникла из следующих соображений. Для разложения на множители однородного многочлена $ae^3 + be^2k + cek^2 + dk^3$ степени 3 относительно переменных e, k с целыми коэффициентами a, b, c, d рассматривается произведение линейного и квадратичного множителей $fk + ge$, $hk^2 + iek + je^2$ с неопределенными целыми коэффициентами f, g, h, i, j. После раскрывания скобок, для неопределенных коэффициентов возникают соотношения $fh = d, gj = a, gi + fj = b, gh + fi = c$. Первое из них позволяет организовать перечисление всевозможных f как делителей известного a, при этом определяется и h. Второе, аналогичным образом, позволяет перечислять g, j. После того, как определены f, j, третье соотношение, переписанное в виде $gi = b - fj$, позволяет перечислять всевозможные g, j. Для текущих найденных значений следует проверять выполнение последнего, четвертого соотношения. Таким образом получаем первый, второй, четвертый и пятый антецеденты теоремы. Третий антецедент нужен для отсечения избыточных случаев - всегда можно поменять знаки всех неопределенных коэффициентов так, чтобы g оказалось положительным (при нулевом g многочлен можно разложить и без данного приема).

Прием имеет заголовок "замена(второйтерм видумножение)". Фильтры "целое(a)", "целое(b)", "целое(c)", "целое(d)" требуют, чтобы коэффициенты были идентифицированы с десятичными записями целых чисел.

Ускоряющий фильтр "постпозиция(фикс(0 1 4)фикс(0 1 1))" отбрасывает симметричные случаи при идентификации - всегда член с третьей степенью переменной k

должен идти после члена с третьей степенью переменной e. Фильтр "меньше(количествооперандов(теквхожд) 5)" также ускоряющий - он блокирует попытки идентификации, если число слагаемых больше 4, и таким образом многочлен заведомо не имеет рассматриваемого вида.

Фильтр "коммент(группировка)" блокирует попытки воспользоваться приемом, если реализуется попытка разложения на множители после предпринятой ранее группировки слагаемых.

Фильтр "контекст(разряд(теквхожд степень x12) не(второйсимвол(x12 2)))" также ускоряющий - он отменяет попытку применения приема, если выражение не имеет степеней, отличных от второй. Фильтр "коммент(6 теквхожд)" блокирует бесперспективную попытку применения приема, если имеется комментарий, указывающий, что многочлен был получен как "длинный" множитель в разложении суммы либо разности кубов.

Фильтр "не(константа(корень))" блокирует попытку применения приема при обработке константных выражений. Этот фильтр, конечно, может отсечь какие-то случаи, в которых прием мог бы оказаться полезен. Однако, потребность в разложении константных выражений на множители возникает сравнительно редко, а замедление попытки такого разложения вносят ощутимое. Поэтому данный фильтр просто означает, что для тех особых случаев, когда разложение константной суммы на множители все-таки необходимо, нужно создать специальный прием, переобозначающий какие-то константные подвыражения на вспомогательные переменные, и затем обращающийся к разложению на множители уже неконстантной суммы.

Фильтр "коммент(нормуравнение)" блокирует применение приема, если есть указание о применении ослабленных средств разложения. Фильтр "меньше(2 количествооперандов(теквхожд))" отсекает вырожденные случаи, когда число слагаемых менее трех.

Указатель "уровень(3)" определяет третий уровень применения приема; "модификатор" - требует, чтобы многочлен не имел слагаемых, кроме приведенных в теореме. Указатель "программа(1 2 3 4 5)" означает, что все антецеденты теоремы являются программно реализуемыми. Первый, второй и четвертый из них используют программу, перечисляющую все множители целого числа. Интересен тот факт, что трудоемкость даже такого неэкономичного перечисления, на фоне трудоемкости общего процесса сканирования задач, невелика, и срабатывание приема выглядит обычно почти мгновенным. Это является следствием того, что архитектура современных процессоров хорошо приспособлена к численным процедурам и никак не приспособлена к логическим процессам.

Указатели "подстановка(фикс(0 1 2)b 0)" и "постановка(фикс(0 1 3)c 0)" определяют возможность отсутствия второго и третьего слагаемых многочлена, причем в этих случаях коэффициенты b, c берутся равными 0.

Указатели "перечень(a десчисло(a))", "перечень(d десчисло(d))" упрощают работу компилятора, фиксируя идентификацию коэффициентов a, d как числовых множителей соответствующих слагаемых.

Указатель "пересечениесписков(фикс(0 1 2)фикс(0 1 3))" объясняет компилятору, что второе и третье слагаемые не обязательно имеют своим заголовком (после отбрасывания возможного минуса) символ умножения. Такое пояснение нужно здесь

из-за того, что эти слагаемые имеют по три множителя, и компилятор пока не способен анализировать эту ситуацию самостоятельно.

Указатели "единица(1 a b c d)" и "заменазнака(минус a b c d)" определяют возможность вырожденных единичных и отрицательных значений коэффициентов.

Прием использует только нормализаторы общей стандартизации; его указатель "титр(...)" обычным образом вводит текстформульное сопровождение.

Нормализаторы утверждений с неизвестными

В некоторых случаях для ускорения разрешения условия задачи на описание относительно неизвестных можно вводить специальные нормализаторы. Обычно это имеет смысл делать тогда, когда условия задачи слабо взаимодействуют между собой, и сранительно длительное изолированное преобразование одного из них не нарушает общего хода решения. Например, так бывает при решении неравенств - обычно каждое из них преобразуется к разрешенному относительно неизвестной виду без использования других. В качестве примера рассмотрим прием нормализатора решения строгих неравенств. Этот нормализатор называется "уравнменьше". Он имеет три уровня срабатывания и снабжен буфером сохранения результатов последних обращений. Шаги его применения в обычной трассировке отображаются на экране. Число приемов этого нормализатора равно 74 - не очень много, так как в него включены лишь простейшие переходы, позволяющие ускорять работу решателя при разборе случаев, часто воникающем во время решения неравенств.

Выберем прием решения простейших неравенств вида $ax < b$. Он расположен в разделе ("Элементарная алгебра", "Неравенства", "Меньше", "Нормализатор решения строгих неравенств УСММЕНЬШЕ", "Решение неравенств $XA < B$,$B < XA$"). Берется последний из приемов раздела.

Теорема приема имеет вид:

$$\forall_{abc}(ac < b \leftrightarrow c < 0 \ \& \ \frac{b}{c} < a \ \vee \ 0 < c \ \& \ a < \frac{b}{c} \ \vee \ c = 0 \ \& \ 0 < b).$$

Заголовок приема - "замена(второйтерм уравнменьше)". Фильтры "не(извест но(a))", "известно(b)", "известно(c)" указывают, что выражение a содержит неизвестные, а выражения b, c - не содержат. Заметим, что список переменных, которые нормализатор будет считать неизвестными, определяется его комментарием (неизвестные $x_1 \ldots x_n$), автоматически вводимым при обращении к нормализатору из задачи, имеющей неизвестные, либо из внешнего пакетного оператора, обладающего таким комментарием. Такая передача нормализатору списка неизвестных обуславливается наличием в описании его формата указателя "неизвестные".

Указатель "перечень(a не(известно(a)))" фиксирует способ идентификации выражения a путем выделения всех неизвестных множителей левой части неравенства. Указатель "заменазнака(минус a)" разрешает относить минус в левой части неравенства к выражению a.

Выражения b,c при формировании заменяющего утверждения обрабатываются нормализатором "видумножение" - это делается, чтобы чаще происходило сокращение нормализатором "нормдробь" дроби b/c. Последнему нормализатору, применяемому в первом и втором подслучаях заменяющей дизъюнкции, придается дополнительная

посылка $c \neq 0$, истинная в обоих этих подслучаях. Неравенства $c < 0$, $0 < c$ и $0 < b$, не содержащие неизвестных, обрабатываются нормализатором "нормменьше" общей стандартизации строгих неравенств. Аналогичным образом, равенство $c = 0$ обрабатывается нормализаторами "нормусм" и "нормчисло", первый из которых предпринимает попытку усмотрения истинности либо ложности утверждений с помощью проверочных операторов, заменяя их после этого на логическую константу, а второй - служит нормализатором общей стандартизации для числовых равенств. Наконец, неравенства с неизвестными $b/c < a$, $a < b/c$ обрабатываются рекурсивным образом - к ним применяется нормализатор "уравнменьше", которому передается дополнительная посылка - неравенство для c. Чтобы исключить логические константы, которые могут возникнуть в заменяющем утверждении после работы перечисленных нормализаторов, ко всему этому утверждению применяется нормализатор "нормлог".

Текстформульный шаблон, сопровождающий применение приема, имеет вид:

"¬1 Так как множитель () положителен, делим на него обе части неравенства, и далее преобразуем полученное неравенство ¬2 Так как множитель () отрицателен, делим на него обе части неравенства и меняем знак неравенства. Далее преобразуем полученное неравенство ¬3 Рассматриваем случаи в зависимости от знака множителя () ¬4 Дальнейшее преобразование неравенства в случае отрицательного множителя ¬5 Дальнейшее преобразование неравенства в случае положительного множителя".

Кроме общей ситуации, в нем предусмотрены случаи однозначно определяемого в контексте знака множителя c - тогда нормализаторы исключат в заменяющей дизъюнкции все случаи, кроме одного, вводя для невыполнимых членов дизъюнкции логическую константу "ложь", которая затем отбрасывается нормализатором "нормлог". Первый фрагмент текстформульного шаблона ориентирован на случай положительного c, второй - на случай отрицательного, третий - соответствует общему случаю неустановленного знака c. Четвертый и пятый фрагменты используются при рекурсивных обращениях к нормализатору "уравнменьше" - эти обращения сопровождаются элементами "замечание(титр(набор(4)))", "замечание(титр(набор(5)))".

Указатель "титр(вариант(заголовок(фикс(0 2 2 1)истина)набор(1 терм(c))вариант(заголовок(фикс(0 2 1 1)истина)набор(2 терм(c)))набор(3 терм(c))))" является диспетчером, выбирающим из текстформульного шаблона нужный фрагмент и передающим ему для подстановки вместо скобок выражение c.

Нормализатор выделения повторяющихся подвыражений с неизвестными

При решении систем уравнений с несколькими неизвестными часто помогает прием перехода к новым неизвестным. Он основан на обнаружении некоторой группы подвыражений с неизвестными, через которые можно выразить все рассматриваемые уравнения. Чтобы такую группу подвыражений можно было усмотреть, обычно приходится прибегать к предварительным преобразованиям уравнений, явным образом выделяя в них повторяющиеся вхождения одного и того же подвыражения. Для этих преобразований используется специальный нормализатор "повторчисло". Этот нормализатор имеет 6 уровней срабатывания. Буфера для сохранения результатов обращений к нему не предусмотрено; при пошаговом показе решения его действия не отображаются. Тем не менее, он относится к числу сравнительно больших нормализаторов, насчитывая более сотни приемов.

Нормализатор "повторчисло" используется в итеративном цикле обработки выделенной группы утверждений с неизвестными A, для которых нужно усмотреть список подвыражений, выбираемых в качестве новых неизвестных. Здесь возможны два случая - несколько уравнений либо одно неравенство. Нормализатору передается текущее утверждение, которое он и будет изменять, но при этом сообщается и весь список утверждений - через комментарий (внешвхождение A), в котором позиция набора A, соответствующая обрабатываемому утверждению, заменяется на 0. Кроме того, нормализатору передается комментарий (новыенеизвестные B), у которого B - изначально пустой накопитель для заполнения его ссылками на те утверждения из A, к рассмотрению которых следует вернуться после обработки текущего утверждения.

В качестве примера рассмотрим прием нормализатора, выражающий сумму квадратов двух выражений через сумму и произведение этих выражений. Такого рода переход может быть полезен, если в обрабатываемых утверждениях уже встречались (быть может, в слегка замаскированном виде) данные сумма и произведение. Прием расположен в разделе ("Элементарная алгебра", "Число", "Нормализатор выделения повторяющихся вхождений числовых выражений ПОВТОРЧИСЛО", "Плюс", "Сумма и разность квадратов", "Выражение суммы квадратов через сумму и произведение"). Берется последний прием раздела. Теорема приема имеет вид:

$$\forall_{bc}(b^2 + c^2 = (b + c)^2 - 2(bc)).$$

Заголовок приема - "замена(второйтерм повторчисло)". Фильтры "не(известно(b))", "не(известно(c))" указывают на то, что выражения b, c содержат неизвестные; фильтр "неизвестные(2)" - что число неизвестных задачи не меньше 2. Последнее требование является естественным, так как прием предполагает выделить для последующего обозначения новыми неизвестными сразу два выражения - сумму и произведение b и c.

Фильтр "лексикопредшествует(b c)" отбрасывает избыточный, в силу симметрии, случай обратного порядка выделения слагаемых b^2, c^2.

Фильтр "контекст(внешвхождение(x4) вид(x4 умножение(x5 b c)) единица(1 x5) буфер(новыенеизвестные базавхождения(x4)))" требует, чтобы хотя бы одно из рассматриваемых для перехода к новым неизвестным утверждений имело вхождение x4 произведения, содержащего множители b,c. Идентифицирующий терм "внешвхождение(x4)"перечисляет вхождения x4, встречающиеся вне текущего вхождения в обрабатываемом нормализатором терме, а также вхождения x4 в другие термы, перечисленные в комментарии (внешвхождение ...). Указатель "буфер(новыенеизвестные базавхождения(x4))", расположенный в данном фильтре, обеспечивает передачу в накопитель комментария (новыенеизвестные ...) всех утверждений, в которых выделено требуемое вхождение x4.

Таким образом, при применении приема будет сохранено указание на повторный просмотр всех утверждений списка A, имеющих вхождение произведения b на c. Этот просмотр будет необходим, если данный прием выделит произведение bc в явном виде - тогда в других утверждениях его тоже придется выделять в явном виде из произведений, содержащих, кроме b и c, дополнительные множители.

Фильтр "контекст(внешвхождение(x4) вид(x4 плюс(x5 умножение(x6 b) умножение(x6 c))) единица(0 x5) единица(1 x6) буфер(новыенеизвестные базавхождения(x4)))" аналогичен предыдущему - он проверяет наличие в рассматриваемых утверждениях суммы вида $kb + kc$.

Фильтр "не(контекст(внешоперанд(x4) вид(x4 степень(x5 2)) контекст(список(x6 b c) контекст(внешвхождение(x7) вид(x7 умножение(x8 x5 x6)) единица(1 x8)) контекст(внешвхождение(x7) вид(x7 плюс(x8 умножение(x9 x5) умножение(x9 x6))) единица(0 x8) единица(1 x9)))))" проверяет, нельзя ли подобрать для b либо c другую пару - некоторое новое слагаемое e^2 той же суммы, в которую входят b^2, c^2 , причем так, чтобы снова были выполнены оба предыдущих фильтра. Если это сделать можно, то ситуация с преобразованием неоднозначная, и применение данного приема тогда просто отменяется.

Указатель "уровень(3)" определяет третий уровень для попыток применения приема. Нормализаторами общей стандартизации обрабатываются только явно выделенные в правой части равенства сумма и произведение выражений b, c. Заметим, что в этой части возникает вложенное произведение - двойка умножается на произведение bc. Это необходимо для последующего переобозначения выделенных вхождений на новые неизвестные, и применение каких-то других нормализаторов общей стандартизации к правой части может ее испортить.

Нормализатор группировок

В процессе развития решателя в ГЕНОЛОГ был введен еще один тип нормализаторов, который впоследствии так и не был востребован. Однако, в определенных ситуациях он может оказаться полезным, и здесь мы вкратце опишем его работу. При выполнении цепочки преобразований выражения, направленных на уменьшение некоторого функционала, например, на уменьшение длины этого выражения, можно перед каждым шагом изменения выражения сначала накапливать множество различных альтернативных преобразований, и лишь по исчерпании новых вариантов принимать окончательное решение. Это решение заключается в определении наиболее ценных, с точки зрения принятого функционала, преобразований обрабатываемого терма, причем отбирается максимальная группа попарно не пересекающихся наиболее ценных локальных изменений, которые и выполняются одновременно. Такой режим работы нормализатора задается указателем "группировка" в описании его формата. Следует заметить, что нормализаторы данного типа обладают ощутимо меньшим быстродействием по сравнению с обычными нормализаторами, выполняющими свои преобразования сразу по обнаружению их возможности. Эффект отбора лучшего варианта обеспечивается у последних распределением приемов по различным уровням и уточнением логики принятия решения. Единственное преимущество нормализаторов с указателем "группировка" (будем называть их нормализаторами группировки) - возможность почти не заботиться о фильтрах их приемов. По существу, такой нормализатор представляет собой просто список тождеств, крайне слабо обработанных с точки зрения управления. Разумеется, это удешевляет нормализатор, но сказывается на качестве его работы.

По этой причине, единственным сохранившимся нормализатором группировки в решателе является нормализатор "группмножество", используемый для сокращенной переформулировки выражений из алгебры множеств. Его можно найти в разделе ("Алгебра множеств", "Множество", "Нормализатор группировки ГРУППМНОЖЕСТВО"). Приемы его практически ничего, кроме теоремы, заголовка и указателя уровня срабатывания, не содержат, и здесь мы их не приводим.

13.1.6 Приемы проверочных операторов

Приемы проверочных операторов обычно просты; рассмотрим сначала несколько приемов оператора "усмменьшеилиравно", предназначенного для проверки нестрогих неравенств. Это - большой оператор, число приемов которого приближается к трем с половиной сотням. Хотя формально он обслуживает произвольные нестрогие неравенства $a \le b$, однако, если a и b ненулевые, один из его приемов сразу сводит проверку данного неравенства к проверке неравенства $a - b \le 0$. Поэтому практически все остальные приемы оператора имеют дело только с неравенствами, одна из частей которых - нулевая. Выберем прием, используемый для усмотрения неположительности произведения. Он расположен в разделе ("Элементарная алгебра", "Неравенства", "Меньшеилиравно", "Проверочный оператор УСММЕНЬШЕИЛИРАВНО", "Алгебраические операции", "Усмотрение знака произведения"); берется последний прием раздела. Теорема приема имеет вид:

$$\forall_{ab}(a \le 0 \ \& \ 0 \le b \to ab \le 0).$$

Заголовок приема - "спуск(усмменьшеилиравно)". Единственный фильтр "уровень(2)" определяет второй уровень срабатывания приема. Указатель "блокпроверок(1 2)" определяет рекурсивное применение того же самого оператора для проверки первого и второго антецедентов. Указатель "титр(набор(1 терм(b a)))" определяет подстановку выражений b и a вместо скобок в шаблон текстформульного сопровождения, имеющий вид "Неположительность произведения неотрицательного множителя () на неположительный множитель ()".

Очевидно, что кроме данного приема, для проверки нестрогого знака произведения необходимы еще прием, устанавливающий неотрицательность произведения неположительных множителей, а также прием, устанавливающий неотрицательность произведения неотрицательных множителей. Оба эти приема расположены в том же разделе. Последний из них интересен тем, что в нем все сомножители рассматриваются одновременно, вместо выделения их по одному, как в первых двух приемах. Так как это может несколько ограничить способность усмотрения неотрицательности (если по отдельности знаки множителей какого-либо фрагмента не устанавливаются, а известен лишь знак всего фрагмента целиком), то дополнительно введен прием усмотрения неотрицательности произведения неотрицательных множителей, аналогичный первым двум - с увеличенным на единицу уровнем срабатывания.

Прием для усмотрения неотрицательности произведения группы неотрицательных сомножителей имеет такого же вида теорему, как и первые два приема:

$$\forall_{ab}(0 \le a \ \& \ 0 \le b \to 0 \le ab).$$

Хотя в ней всего два множителя, однако указатель "дистрибразвертка(фикс(0 2))" обобщает ее на случай произвольного числа множителей. В остальном этот прием ничем не отличается от рассмотренного выше.

Иногда неотрицательность выражения нужно установить лишь в произвольно малой окрестности заданной точки. Тогда к списку посылок оператора "усмменьшеилиравно" добавляется утверждение "стремится(x a b)", или, в формульной записи, $x \to a\backslash b$; здесь b - указатель типа окрестности (двусторонняя, левая, правая). Это утверждение - лишь техническая условность, позволяющая избежать навешиваний

кванторов на весь текущий контекст. В качестве примера приема, учитывающего наличие данного утверждения, рассмотрим прием, который для установления неотрицательности выражения в окрестности точки вычисляет предел его в данной точке. Теорема приема имеет вид:

$$\forall_{abcx}(x \to a\backslash b \ \&\ c = \lim_{x \to a\backslash b} f(x) \ \&\ (c = \infty \vee 0 < c) \to 0 \le f(x)).$$

Заголовок приема - "спуск(усмменьшеилиравно)"; фильтр "уровень(3)" определяет его применение на третьем уровне.

Фильтр "не(входит(предел c))" проверяет, что предел удалось вычислить. Фильтр "входит(x фикс(0 2))" проверяет, что переменная x, для которой задано стремление к a, входит в рассматриваемое выражение - иначе вычисление предела ничего не изменит. Фильтр "коммент(быстрпроверка)" блокирует применение приема в ситуациях, когда есть указание на проведение ускоренной проверки неотрицательности, так как вычисление предела может оказаться достаточно трудоемкой процедурой.

Комментарий "перестановка" вводится приемами, которые для проверки неравенства предпринимают попытку перестановки его частей одновременно с изменением знаков всех слагаемых этих частей. В данном случае имеются два независимых аналогичных приема - один для проверки неотрицательности, другой для проверки неположительности. Поэтому применение приема при наличии комментария "перестановка" излишне, и отмена его обеспечивается фильтром "коммент(перестановка)".

Фильтр "не(контекст(комментарий(слагаемое меньше d)))" аналогичен предыдущему. Комментарий (слагаемое меньше ...) вводится, например, если проверка неравенства $0 \le a$ сводится, с помощью неравенства $0 \le a + b$ из посылок, к проверке $0 \le b$, а также в ряде аналогичных случаев. Таких переходов может оказаться достаточно много, и для ускорения проверок в этих случаях вычисление предела блокируется.

Указатели "идентификатор(2)", "блокпроверок(3)", "отображение(f)" - стандартные. Указатель "потвор(1)" отменяет ограничения на повторное использование утверждения "стремится(...)". Такие ограничения вводятся автоматически для всех утверждений списка посылок, уже идентифицированных некоторым внешним (по цепочке обращений) пакетным оператором. Это делается лишь для предотвращения зацикливаний в рекурсивных обращениях, и в особых случаях может отменяться.

Для вычисления предела правая часть второго антецедента обрабатывается нормализатором "нормпредел".

Кроме проверочных операторов "усмменьшеилиравно" и "усмменьше", обеспечивающих сравнительно быструю проверку нестрогих либо строгих неравенств, введены еще два проверочных оператора - "провменьшеилиравно" и "провменьше", предназначенные для усиленной проверки, с привлечением ряда специальных неравенств типа неравенства для среднего геометрического и среднего арифметического. Хотя число приемов в этих операторах и невелико, но работают они в режиме перебора группировок, и обращаться к ним слишком часто нецелесообразно. В качестве примера рассмотрим прием оператора "провменьшеилиравно", позволяющий группировать в правой части неравенства неотрицательную сумму. Эта сумма составляется из частично используемых двух положительных членов, коэффициенты которых

выбираются так, чтобы получилась сумма квадратов некоторых выражений, а модуль третьего члена суммы оказался равен удвоенному произведению этих выражений. Прием расположен в разделе ("Элементарная алгебра", "Неравенства", "Меньшеилиравно", "Проверочный оператор ПРОВМЕНЬШЕИЛИРАВНО", "Поглощение одного слагаемого двумя с применением квадратичной оценки"). Берется второй из двух приемов раздела. Теорема приема имеет вид:

$\forall_{abcdefghij}(0 \le a \;\&\; 0 \le b \;\&\; c^2 = ab \;\&\; d = 2e - |f| \;\&\; g = 2h - |f| \;\&\; 0 \le d \;\&\; 0 \le g \;\&\; 2j \le da + gb + 2i \to j \le ea + hb + fc + i)$

Эта теорема возникла следующим образом. Для проверки неравенства $j \le ea + hb + fc + i$, у которого e, h, f - численные коэффициенты; выражения a, b неотрицательны, причем c равно произведению квадратных корней из a и b (что формулируется в теореме приема как $c^2 = ab$), предпринимается представление правой части в виде

$$(e - \frac{|f|}{2})a + (h - \frac{|f|}{2})b + (\frac{|f|}{2}a + \frac{|f|}{2}b + f\sqrt{a}\sqrt{b}) + i.$$

Третье слагаемое, очевидно, представляет собой полный квадрат и поэтому неотрицательно. Обозначая коэффициенты первых двух слагаемых через $d/2, g/2$ и отбрасывая неотрицательное третье слагаемое, получаем, что нужно установить неравенство $j \le (d/2)a + (g/2)b + i$, или, после устранения знаменателей, $2j \le da + gb + 2i$.

Заголовок приема - "спуск(провменьшеилиравно)". Фильтры "десчисло(e)", "десчисло(h)", "десчисло(f)" указывают компилятору, что e, h, f должны идентифицироваться с десятичными числами. Фильтр "не(легковидеть(меньшеилиравно(0 умножение(f c))))" блокирует применение приема в случае, когда поглощаемое слагаемое fc само является неотрицательным. Фильтр "уровень(1)" определяет применение приема на первом уровне сканирования. Обычно такой фильтр размещался вначале, однако действительный порядок размещения фильтров в программе приема не очень сильно зависит от порядка размещения их в описании приема; в данном случае расположение фильтра совершенно несущественно.

Фильтр "или(меньше(количествоопер андов(фикс(0 2)) 6)меньше(1 число(вид(фикс(0 2) плюс(x11 x12)) единица(0 x11) не(легковидеть(меньшеилиравно(0 x12))))))" блокирует применение приема, если число слагаемых правой части слишком велико (больше пяти), а число тех из них, для которых не усматривается неотрицательность, не более одного. Фильтр "контекст(входит(x11 параметры(фикс(0 2))) меньше(1 число(разряд(фикс(0 2)x11 x12))))" требует, чтобы правая часть неравенства имела хотя бы одну переменную, встречающуюся неоднократно.

Указатель "блокпроверок(1 2)" определяет использование проверочного оператора для усмотрения неотрицательности выражений a, b. Заметим, что здесь будет использован проверочный оператор "усмменьшеилиравно". Оператор "провменьшеилиравно", как оператор усиленной проверки, требует указателя "проверка(...)". Такой указатель "проверка(8)" определяет рекурсивное обращение для проверки результирующего неравенства $2j \le da + gb + 2i$. Напомним, что различие между обычным и усиленным проверочными операторами заключается в описании их формата. Первый задается термом вида "легковидеть(...)", по которому создается справочник "легковидеть"; второй - термом вида "проверка(...)", по которому создается справочник "проверка".

Указатель "идентификатор(3)" определяет использование третьего антецедента для идентификации переменной b по уже идентифицированным a, c. Указатель "программа(4 5 6 7)" определяет использование антецедентов, начиная с четвертого по

седьмой, для непосредственных вычислений с десятичными числами. Указатели "единица(1 b e h f)", "заменазнака(минус e h g)", "единица(0 i)" уточняют вырожденные случаи при идентификации переменных.

Указатели "определено(фикс(0 2 2)b)" и "определено(фикс(0 2 1)c)" блокируют идентификацию слагаемого hb до тех пор, пока переменная b не будет идентифицирована из третьего антецедента, а слагаемого ea - пока не будет идентифицировано c. Это уменьшает перебор в попытках применения приема - сначала будет идентифицироваться слагаемое fc, затем - слагаемое ea, далее будет рассматриваться третий антецедент, который, во-первых, отсечет те слагаемые ea, у которых c^2 не делится на a, а, во-вторых, определит b для однозначной идентификации слагаемого hb.

Указатель "пересечениесписков(фикс(3 2))" явно указывает компилятору, что правая часть третьего антецедента может идентифицироваться с выражением, заголовок которого отличен от символа "умножение".

Указатели "перечень(f десчисло(f))", "перечень(e десчисло(e))" относят к e, f все числовые множители слагаемых, идентифицируемых с ea,fc, таким образом позволяя однозначно выделить e и c. Оставшиеся указатели используются для формирования текстформульных пояснений.

В заключение рассмотрим проверочный оператор из геометрии - оператор "однасторона" для усмотрения того, что две точки плоскости лежат по одну сторону от заданной прямой. В случае размещения точек по разные стороны имеется двойственный оператор "разныестороны". Оба оператора - сравнительно большие, имеют около сотни приемов каждый, и весьма часто используются в геометрических приемах. Выберем прием оператора "однасторона", усматривающий размещение двух точек по одну сторону от прямой из соображений двукратного перехода через эту прямую при движении по ломаной. Он расположен в разделе ("Элементарная геометрия", "Однасторона", "Проверочный оператор ОДНАСТОРОНА", "Двойной переход через прямую"). Опуская несложный чертеж, приведем сразу теорему приема:

$\forall_{ABCDEFG}(D \in$ отрезок(CE) & $F \in$ отрезок(CG) & $D \in$ прямая(AB) &
$F \in$ прямая(AB) & разныеточки$(C, D) \rightarrow$ однасторона$(E, G,$ прямая$(AB)))$.

Здесь точка E является концом отрезка CE, на котором лежит точка D рассматриваемой прямой AB. Последний антецедент требует, чтобы точки C, D различались, так что либо точка E сама лежит на прямой, и тогда любая точка оказывается по отношению к ней лежащей по ту же сторону от прямой (отношение "однасторона" нестрогое), либо точка E и точка C не лежат на прямой и находятся от нее по разные стороны. В любом случае конец G отрезка CG, пересекающегося с прямой в некоторой точке F, будет лежать с точкой E по одну сторону от прямой AB.

Прием имеет заголовок "спуск(однасторона)". Фильтр "уровень(3)" определяет уровень его срабатывания; фильтры "не(равно(C D))" и "не(равно(C F))" - ускоряющие. Первый из них отбрасывает случаи совпадения точек C, D еще до обращения к проверочному оператору "разныеточки". Второй - отбрасывает вырожденный случай совпадения точек C, F, в котором нет надобности применять данный прием, так как здесь точка E должна будет лежать на прямой AB, и сработает другой прием.

Указатель "усм(1 2 3 4)" определяет использование идентифицирующих операторов при обработке первых четырех антецедентов, указатель "блокпроверок(5)" - обращение к проверочному оператору для обработки последнего антецедента.

13.1.7 Приемы синтезаторов

Синтезатор представляет собой пакетный оператор для подбора значений одного или нескольких операндов заданного отношения, при которых это отношение становится истинным. Здесь возможна либо однократная выдача результата, либо перечисление серии результатов, отбираемых на основе каких-либо эвристических соображений приемами синтезатора. Как уже говорилось ранее, синтезатор является пакетным аналогом задачи на описание, причем для случая, когда эта задача имеет единственное условие - реализуемое синтезатором отношение. Свое название синтезатор получил из-за того, что обычно его прием сводит рассматриваемое отношение к одному или нескольким более простым отношениям, и после реализации их составляет ("синтезирует") свой ответ из полученных элементов.

В качестве первого примера синтезатора рассмотрим оператор "верхняяоценка", используемый для нахождения константных верхних оценок численных выражений. Реализуемое синтезатором утверждение имеет вид $a \le b$, причем переменная a рассматривается как входная, а переменная b - как выходная. Указатель "перечисление" в описании формата оператора определяет использование его в режиме перечисления верхних оценок. В действительности, такое перечисление обычно не очень велико и зависит от того, сколько различных способов оценивания выражения a оказалось предусмотрено в приемах синтезатора. Какая именно из версий найденных верхних оценок окажется достаточной - определяется внешним приемом, обратившимся к синтезатору.

Оператор "верхняяоценка" имеет четыре уровня срабатывания; он не очень велик - насчитывает около 50 приемов. Рассмотрим прием этого оператора, получающий верхнюю оценку для линейной комбинации синуса и косинуса. Он расположен в разделе ("Элементарная алгебра", "Неравенства", "Меньшеилиравно", "Синтезатор ВЕРХНЯЯОЦЕНКА", "Линейная комбинация синуса и косинуса"). Теорема приема имеет вид:

$$\forall_{abc}(a \sin b + c \cos b \le \sqrt{a^2 + c^2}).$$

Заголовок приема - "значение(верхняяоценка)". Прием имеет два фильтра - "константа(a)" и "константа(c)", так что применим только к линейным комбинациям синуса и косинуса, имеющим постоянные коэффициенты. При необходимости его легко обобщить на случай неконстантных коэффициентов, обратившись рекурсивным образом к получению верхней оценки для радикала в правой части неравенства. Указатели и нормализаторы приема - стандартные.

Другой пример использования синтезатора - определение периода числовой функции. Заголовок синтезатора - "период"; реализуемое отношение - "периодична(a b)", истинное, если b есть какой-либо из периодов функции a. Приведем три простых приема этого синтезатора. Первый из них определяет период синуса; второй - отбрасывает при определении периода внешнюю одноместную операцию; третий - находит период результата двуместной операции. Эти приемы расположены в разделе ("Математический анализ", "Общий свойства числовых функций", "Периодичность", "Синтезатор определения периода ПЕРИОД") - соответственно, в подразделах "Синус и косинус"; "Одноместная операция - общий случай"; "Двуместная операция". Теорема первого из них имеет вид:

$$\forall_{abc}(\text{периодична}(\lambda_x(\sin(\frac{ax}{b} + c), \text{число}(x)), \frac{2\pi b}{|a|})).$$

Заголовок приема - "значение(период)"; единственный фильтр - "уровень(1)". Указатели и нормализаторы - стандартные. Этот прием относится к категории "завершающих" приемов, предназначенных для немедленной выдачи результата. Второй и третий приемы обеспечивают рекурсию, подготавливающую возможность применения завершающих приемов. Теорема второго приема имеет вид:

$$\forall_{fga}(\text{периодична}(\lambda_x(f(x), \text{число}), a) \rightarrow \text{периодична}(\lambda_x(g(f(x)), \text{число}), a)).$$

Указатель "символ(g)" определяет идентификацию переменной g с символом одноместной операции. Эта операция отбрасывается, и указатель "значения(1)" определяет рекурсивное обращение для определения периода "упрощенной" таким образом функции.

Теорема третьего приема имеет вид:

$\forall_{fgabcdmn}(\text{периодична}(\lambda_x(f(x), \text{число}(x)), a)$ & $\text{периодична}(\lambda_x(g(x), \text{число}(x)), b)$ & $a = nc/pd$ & $b = mc/qd$ & целое(n) & целое(m) & целое(p) & целое(q) $\rightarrow$ периодична $(\lambda_x(f(x) + g(x), \text{число}(x)), \text{нок}(mp, nq)c/pqd))$.

Для определения периода суммы двух функций определяются периоды a, b слагаемых, и если эти периоды соизмеримы, то находится их наименьшее общее кратное, которое и выдается в качестве результата. Указатель "вариант(фикс(0 1 3)умножение дробь степень)" обобщает этот прием на случай других двуместных арифметических операций. Фильтры "входит(x фикс(1 1 3))" и "входит(x фикс(2 1 3))" ограничивают применение приема лишь случаями, когда оба слагаемых содержат переменную x. В случае константного слагаемого применяется другой прием, просто отбрасывающий это слагаемое. Указатели "значения(1)" и "значения(2)" определяют рекурсивные обращения к нахождению периодов слагаемых. Прочие указатели приема и его нормализаторы - стандартные.

В заключение рассмотрим геометрический синтезатор "вычислениедлины", который используется для вычисления длины некоторого расстояния. Он несколько отличается от разобранных выше, так как результатом его применения служит не выражение для искомого расстояния, а лишь конъюнкция соотношений, из которых такое расстояние может быть найдено. Синтезаторы этого типа обычно используются в контексте вывода следствий при анализе некоторой системы утверждений. Фактически, синтезатор здесь лишь составляет план вычислений, а сами вычисления, после перенесения найденной конъюнкции соотношений в список посылок текущей задачи на исследование, реализуются обычными приемами сканирования задачи. Это упрощает приемы синтезатора и делает планирование вычислений более гибким.

В качестве примера рассмотрим прием синтезатора, использующий теорему синусов для вычисления стороны треугольника по известным стороне и двум углам. Этот прием расположен в разделе ("Элементарная геометрия", "Расстояние", "Синтезатор ВЫЧИСЛЕНИЕДЛИНЫ", "Теорема синусов"). Берется первый прием раздела.

Опуская чертеж (изображение треугольника ABC), приведем сразу теорему приема:

$\forall_{ABC}(\text{актив}(\angle(ABC))$ & $\text{актив}(\angle(ACB))$ & $\text{актив}(l(AB)) \rightarrow$ вычислениедлины($l(AC), \sin(\angle(ABC))l(AB) = \sin(\angle(ACB))l(AC))$).

Заголовок приема - "значение(вычислениедлины)". Фильтр "уровень(3)" определяет уровень срабатывания. Фильтры "конец(известно(терм(расстояние(AB))))", "конец(известно(терм(угол($A\ B\ C$))))", "конец(известно(терм(угол($A\ C\ B$))))" указывают, что длина стороны AB и углы ABC, ACB известны. Размещение выражений

"расстояние(...)", "угол(...)" под символом "терм" означает, что рассматриваться будут не сами эти выражения, а результаты обработки их нормализаторами приема. В данном случае это нормализаторы "нормрасстояние", "нормугол", которые не делают практически ничего, кроме усмотрения в контексте равенства, дающего своей правой частью выражение R для рассматриваемого расстояния либо угла, и замены преобразуемого терма на R. Обработка фильтров отнесена на конец программы приема - сначала выполняется идентификация антецедентов теоремы. На это указывает размещение фильтров под символом "конец".

Фильтры "усм(актив(прямая(AB)))", "усм(актив(прямая(AC)))", "усм(актив(прямая(BC)))" означают, что прямые для сторон треугольника должны быть введены в рассмотрение.

Указатель "усм(1 2 3)" определяет обработку первых трех антецедентов теоремы идентифицирующими операторами. Нормализаторы приема - стандартные.

13.1.8 Приемы анализаторов

Анализаторы представляют собой пакетные операторы для быстрого вывода следствий с определенной целью. По достижении данной цели вывод обрывается, и отобранные утверждения переносятся из накопителя анализатора в список посылок внешней задачи. Экономия от использования анализатора состоит в том, что каждое новое утверждение здесь рассматривается существенно меньшим числом приемов, чем это было бы при занесении его в список посылок задачи. В результате появляется возможность просматривать весьма значительные списки следствий за умеренное время; перенесение всех этих следствий в основную задачу привело бы к непомерному замедлению ее решения - эффекту "увязания" решателя в массиве малополезных длинных утверждений. Роль накопителя играет в анализаторе вспомогательная задача на исследование, копирующая список посылок текущей задачи.

В качестве примера рассмотрим, прежде всего, анализатор "сложениеуравнений", используемый для вывода следствий при решении систем уравнений в элементарной алгебре. Он имеет совсем немного приемов (около десятка). Включение анализатора предпринимается дважды - сначала на третьем уровне текущей задачи на исследование он активизируется для применения приемов нулевого и первого уровней; затем, по достижении седьмого уровня сканирования задачи, активизируется уже до наибольшего своего уровня, равного двум.

Выберем прием этого анализатора, предпринимающий создание линейных комбинаций двух уравнений, имеющих "подобные" неизвестные члены. Этот прием расположен в разделе ("Элементарная алгебра", "Плюс", "Анализатор СЛОЖЕНИЕ-УРАВНЕНИЙ", "Линейная комбинация уравнений, имеющих неизвестные члены, отличающиеся лишь известными коэффициентами"). Теорема приема имеет вид:

$$\forall_{abcdefgh}(ag + b = c \;\&\; ah + d = e \;\&\; f = bh - dg - ch + eg \rightarrow f = 0).$$

Здесь первый и второй антецеденты суть исходные уравнения; коэффициенты g, h предполагаются в приеме не содержащими неизвестных, а выражение a - содержащим. Из второго уравнения, домноженного на g, вычитается первое, домноженное на h; все члены переносятся в левую часть, и определяется результат f обработки этой части различными нормализаторами (раскрывание скобок, и т.п.).

Заголовок приема - "внутрвывод(сложениеуравнений)". Фильтр "не(известно(a))" указывает, что общая часть двух подобных членов ag,ah должна содержать неизвестные. Фильтр "не(контекст(список(x9 b d)вид(x9 плюс(x10 дробь(x11 x12)))заменазнака(минус x11)единица(0 x10)))" означает, что уравнения не должны иметь дробных слагаемых. Обычно для уравнения с дробными слагаемыми в задаче на исследование выводится следствие - результат устранения у него знаменателей. Естественно при получении линейных комбинаций уравнений источники таких следствий отбрасывать.

Фильтр "не(заголовок(f 0))" отсекает случаи, когда рассматриваемые уравнения были пропорциональны. Фильтр "известно(h)" требует, чтобы коэффициент h не содержал неизвестных. Аналогичное требование для коэффициента g излишне, так как он изначально формируется в виде произведения всех известных множителей - согласно указателю "перечень(g известно(g))".

Последним размещен фильтр, сравнивающий новое уравнение с исходными. Он имеет вид "или($A_1A_2A_3$)", где A_1 - подфильтр "упрощение(x9 и(f фикс(1 1) фикс(2 1)) число(вид(x9 плюс(x10 x11)) единица(0 x11) не(известно(x10))))"; A_2 - подфильтр "упрощение(x9 и(f фикс(1 1) фикс(2 1)) числонеизвестных(x9))"; A_3 - подфильтр "и(входит(полный комментарии) контекст(неизвестная(x9) линейно(f x9)))". В этих подфильтрах указатели вхождения "фикс(1 1)" и "фикс(2 1)" ссылаются на левые части первого и второго исходных уравнений; f есть левая часть нового уравнения. Первый подфильтр означает, что левая часть каждого из исходных уравнений имеет строго больше неизвестных слагаемых, чем f. Второй означает, что левая часть каждого из исходных уравнений имеет большее число неизвестных, чем f. Наконец, третий означает, что при обращении к анализатору был введен комментарий "полный", причем результат f линеен хотя бы по одной из неизвестных задачи, входящих в f. Заметим, что комментарий "полный" вводится автоматически в тех случаях, когда максимальный уровень текущей задачи на исследование не менее 7. При решении систем алгебраических уравнений это означает вторую попытку вывода следствий из объединенного списка условий и посылок.

Указатели "единица(1 g h)" и "единица(0 d)" разрешают вырожденные единичные значения коэффициентов g, h и обращение в 0 суммы остальных членов левой части второго уравнения. Левая часть хотя бы одного (пусть оно и будет первым) уравнения должна иметь хотя бы два неизвестных слагаемых - иначе результирующее равенство не будет содержать неизвестных. Указатель "идентификатор(3)" определяет использование третьего антецедента для вычисления f. Указатель "заменазнака(минус g h)" разрешает отнесение минуса, если он есть, к соответствующему коэффициенту g, h.

Указатель "перечень(g известно(g))" определяет идентификацию коэффициента g из совокупности всех сомножителей, не содержащих неизвестных. Указатель "набороперандов(фикс(1 1 1)фикс(2 1 1))" введен для ускорения идентификации. Он отменяет нахождение общей неизвестной части a двух слагаемых ag, ah с использованием трудоемкого оператора "алгебрпересечение", находящего наибольший общий делитель двух одночленов, и a идентифицируется как пересечение двух групп сомножителей этих слагаемых. Так как g, h не должны содержать неизвестных, то при отсутствии неизвестных показателей степеней это не ограничивает общности идентификации.

Указатели вида "обрыв(...)" используются для формулировки условий обрыва работы анализатора. При этом только что выведенное утверждение снабжается ком-

ментарием "внешвывод", и происходит передача всех выделенных таким комментарием посылок анализатора во внешнюю задачу на исследование. В данном приеме имеем следующие три условия обрыва вывода. Во-первых, это "обрыв(равно(число-неизвестных(f)1))" - получение уравнения с единственной неизвестной. Во вторых, "обрыв(контекст(вид(f плюс(умножение(x9 степень(x10 x13)) умножение(x11 степень(x12 x13)))) единица(1 x9 x11 x13) заменазнака(минус x9 x11) неизвестная(x10) неизвестная(x12) известно(x9) известно(x11) известно(x13)))" - получение уравнения, выражающего пропорциональность двух неизвестных X, Y: $pX^n + qY^n = 0$; p, q, n - известны. В-третьих, "обрыв(заголовок(результат или))" - при обработке нормализаторами левой части нового уравнения ее удалось разложить на несколько неизвестных множителей, что привело к преобразованию этого уравнения к виду дизъюнкции более простых уравнений.

Указатель "прообраз(1 посылки(не(равно(h 0))))" нужен для регистрации в комментариях того факта, что первое уравнение, в случае ненулевого коэффицента h, является следствием второго исходного уравнения и нового уравнения. Эта информация может оказаться полезной, если при выводе следствий в блоке анализа будут найдены значения неизвестных. Тогда будет предпринята попытка установить с ее помощью, что текущие уравнения задачи на описание являются следствиями равенств R, определяющих значения неизвестных, а также, быть может, ряда других простых сопровождающих утверждений U блока анализа. В этом случае проверка текущих уравнений задачи на описания будет излишней, и они просто будут заменены на утверждения R, U. Указатель "прообраз(2 посылки(не(равно(g 0))))" используется аналогичным образом для второго уравнения.

Указатель "примечание(условие(и(не(контекст(неизвестная(x9) список(x10 фикс(1 1)фикс(2 1)) линейно(x10 x9))) контекст(неизвестная(x9) линейно(f x9))) внешвывод))" помечает новое уравнение комментарием "внешвывод", если оно оказывается линейным по некоторой неизвестной, по которой оба исходных уравнения были нелинейны. Такой комментарий предопределяет перенесение данного уравнения в список посылок внешней задачи на исследование по окончании работы анализатора, без немедленного ее обрыва.

Следующий указатель "примечание(условие(и(контекст(неизвестная(x9) список(x10 фикс(1 1) фикс(2 1)) не(линейно(x10 x9))) не(контекст(неизвестная(x9) не(линейно(f x9))))) внешвывод))" - аналогичен. Он помечает новое уравнение комментарием "внешвывод", если оно линейно по всем своим неизвестным, а хотя бы одно из исходных уравнений - не линейно по какой-либо неизвестной.

Правая часть третьего антецедента теоремы обрабатывается нормализаторами "нормплюс", "стандплюс", "уравнплюс". Первые два из них обеспечивают раскрывание скобок в линейной комбинации; третий - приведение подобных членов с известными коэффициентами. После определения f оно обрабатывается нормализатором "факторизация" упрощенного разложения на множители. Наконец, само новое уравнение $f = 0$ обрабатывается нормализатором "нормчисло", который, в частности, может преобразовать его к виду дизъюнкции, если левая часть имеет вид произведения.

Следующий пример анализатора - оператор "смплощадь", используемый в геометрии для вывода соотношений, связывающих между собой площади выделяемых на чертеже фигур. В основном, здесь применяются приемы, составляющие площадь фигуры из площадей подфигур, а также приемы, усматривающие пропорциональность площадей. Анализатор имеет 5 уровней срабатывания и около сорока приемов.

Рассмотрим один из простейших его приемов, разбивающий треугольник ABC на два подтреугольника ABD,BDC, если вершина B соединена отрезком с точкой D на стороне AC, причем известно, в каком отношении эта точка делит сторону AC. Прием расположен в разделе ("Элементарная геометрия", "Фигуры", "Площадь", "Анализатор СМПЛОЩАДЬ", "Разбиение треугольника на два треугольника линией, проведенной из вершины к противоположной стороне"). Берется первый из приемов раздела. Опуская несложный чертеж, приводим сразу теорему приема:

$\forall_{ABCDab}$(актив(S(фигура(ABC))) & $D \in$ отрезок(AC) & $al(AD) = bl(CD)$ & $\neg(a + b = 0) \rightarrow$ актив(S(фигура(ABD))) & актив(S(фигура(BCD))) & S(фигура(ABD)) = bS(фигура(ABC))/$(a + b)$ & S(фигура(ABC)) = S(фигура(ABD)) + S(фигура(BCD))).

Заметим, что соотношения пропорциональности в теоремах геометрических приемов обычно записываются так, как это сделано в третьем антецеденте данной теоремы, т.е. $al(AD) = bl(CD)$ вместо $l(AD)/l(CD) = b/a$. Это объясняется, во-первых, тем, что общая стандартизация посылок приводит к исключению дробей в равенствах, определяющих пропорциональность величин. Во-вторых, для усмотрения пропорциональности используется синтезатор "пропорциональны", определяющий коэффициенты пропорциональности a, b по заданным величинам A, B, и его входной шаблон имеет вид с исключенными дробями - $aB = bA$. В качестве допустимого результата этот синтезатор может выдавать только выражения a, b, не имеющие нечисловых переменных, причем каждая переменная этих выражений должна быть либо известным параметром задачи на исследование, либо неизвестной внешней задачи на описание.

Первый антецедент означает, что площадь треугольника ABC уже входила в какую-либо из посылок. В этой ситуации срабатывает прием, вводящий вспомогательную посылку "актив(площадь(...))", чтобы упростить последующие ссылки на явно рассматриваемые в задаче площади. Теорема приема вводит в рассмотрение площади треугольников ABD,BCD и регистрирует в посылках анализатора соотношения, связывающие их с площадью треугольника ABC.

Прием имеет заголовок "внутрвывод(смплощадь)". Его фильтр "комментпосылки(1 смплощадь D)" нужен для предотвращения повторного срабатывания с тем же самым разбиением треугольника ABC. После применения данного приема вводится комментарий (смплощадь D) к посылке "актив(S(фигура(ABC))", - это делает указатель "примечпосылки(1 смплощадь D)", - и повторные срабатывания оказываются заблокированы.

Фильтры "не(равно(D A))", "не(равно(D C))" отсекают неинтересные случаи совпадения точки D с концами отрезка AC. Наконец, фильтр "усм(актив(прямая(B D)))" требует, чтобы прямая, соединяющая точки B, D, уже была введена в рассмотрение. Заметим, что в том же самом разделе имеется версия данного приема, у которой такая прямая еще не ведена в рассмотрение; тогда вводится ряд дополнительных фильтров, и уровень срабатывания приема повышается.

Указатель "уровень(1)" определяет первый уровень срабатывания анализатора. Указатель "значения(3)" определяет использование синтезатора "пропорциональны" для определения коэффициентов a, b в третьем антецеденте. Перед обращением к нему выражения $l(AD), l(CD)$ обрабатываются нормализатором "нормрасстояние". Указатель "блокпроверок(4)" определяет использование проверочного оператора для проверки отличия $a + b$, которое далее выступает как знаменатель, от нуля. Указа-

тель "усм(2)" определяет обработку второго антецедента идентифицирующим оператором.

Указатель "список(фикс(1 1 1 1))" нужен для того, чтобы треугольник ABC идентифицировался с произвольным порядком перечисления его вершин в первом антецеденте. Заметим, что в скобочном представлении этот антецедент имеет вид "актив(площадь(фигура(набор(A B C))))", так что указатель вхождения "фикс(1 1 1 1)" относится к подвыражению "набор(...)".

Указатели "копия(фикс(0 1 1))", "копия(фикс(0 2 1))" отменяют обработку нормализатором "нормплощадь" выражений S(фигура(ABD)), S(фигура(BCD) в утверждениях "актив(...)" правой части теоремы. Если бы эти площади были выражены через какие-либо параметры задачи, то утверждения "актив(...)", как отметки о рассмотрении площадей конкретных треугольников, оказались бы испорчены.

Указатель "титр(...)", обеспечивающий текст-формульное сопровождение срабатывания приема, и нормализаторы приема - стандартные.

13.1.9 Операторы фильтра

Операторы фильтра используются в решателе крайне редко, и общее число их едва ли насчитывает сейчас полтора десятка. Одна из веских причин прибегнуть к оператору фильтра - появление чрезмерно большой дизъюнкции, перечисляющей подслучаи, в которых применение приема допустимо. Помимо того, что эта дизъюнкция может не поместиться в окне описания приема, успешная компиляция ее также не гарантирована. В то же время, оператор фильтра позволяет компилировать программы проверки отдельных подслучаев по отдельности, и каких-либо существенных ограничений на число его приемов нет. В качестве примера рассмотрим оператор фильтра "фильтркквадратов", используемый для принятия решения о переходе от произведения суммы двух выражений на их разность к разности квадратов этих выражений. Он имеет всего 4 приема. Первый из них проверяет наличие внешней дроби, у которой переход к разности квадратов дает возможность сокращения. Второй - проверяет, не являются ли слагаемые константами, хотя бы одна из которых - радикал четной степени. Третий прием проверяет, не является ли заменяемое вхождение корневым, и четвертый - проверяет, не приводит ли данное преобразование к устранению нигде более не встречающихся радикалов. При срабатывании хотя бы одного из приемов оператор получает значение "истина". Рассмотрим запись первого приема (см. раздел "Элементарная алгебра", "Умножение", "Усмотрение разности квадратов", "Определение целесообразности перехода к разности квадратов", "Возможность сокращения дроби при переходе к разности квадратов").

Приемы операторов фильтра имеют фиктивную теорему - логический символ "блок". Однако, эта теорема на экран не выводится (чтобы посмотреть на нее, нужно войти в режим редактирования теоремы текстовым редактором, нажав Ctr-т). Вместо теоремы выводится текст первого пункта оглавления для раздела оператора фильтра - он содержит напоминание о том, какие переменные определены на момент обращения к оператору и что они означают. В данном случае имеем текст: "x1,x2 - вхождения суммы и разности выражений x3,x4. x5 - результат нормализации разности квадратов".

Прием имеет заголовок "контекст(фильтркквадратов)". Его единственный фильтр - "подтерм(дробь(умножение(x6 теквхожд(x1))умножение(x5 x7)))". Заметим, что для

фильтров приема оператора фильтра не обязательно размещение их внутри конструкции "контекст(...)" - здесь она играет роль отбрасываемых по умолчанию внешних скобок. Компилятор восстанавливает эти внешние скобки автоматически. Поэтому указанный выше фильтр "подтерм(...)" и не помещен внутри "контекст'а", хотя он вводит новые переменные x6,x7. Смысл этого фильтра - усмотрение внешней дроби, у которой преобразуемые выражения служат множителями числителя, а знаменатель имеет множитель x5, т.е. разность квадратов. Указатель "дробь" (фактически возникший из-под отброшенного "контекст'а") разрешает менять в этой идентификации местами числитель и знаменатель. Указатель "единица(1 x7)" разрешает отсутствовать множителю x7.

13.2 Упражения по работе на ГЕНОЛОГе

13.2.1 Поиск приемов

Приемы решателя распределены по разделам подробного оглавления, устроенного по тематическому принципу. Ссылка на каждый прием осуществляется из единственного пункта оглавления, причем иногда бывают возможны различные естественные решения о выборе такого пункта. Поэтому для быстрой ориентации в накопленном материале полезна некоторая тренировка. Приведем ряд простых упражнений по поиску нужного приема либо раздела.

1. Найти корневое меню оглавления базы приемов. Найти разделы "Тригонометрия"; "Окружность"; "Эллипс"; "Вычисление определенных интегралов"; "Признаки сходимости знакопеременных рядов"; "Степень матрицы"; "Уравнение в полных дифференциалах"; "Суммы и разности мощностей множеств"; "Закон Пуассона"; "Движение по наклонной плоскости".

2. Найти приемы логарифмирования показательных уравнений. Выбрать те из них, у которых уравнение имеет ноль в правой части, а основанием логарифма служит основание неизвестной степени.

3. Найти приемы решения строгих квадратных неравенств. Объяснить назначение каждого из них.

4. Найти приемы, выводящие соотношение пропорциональности для длин сторон треугольника и длин отрезков, на которые биссектриса делит сторону треугольника. Определить отличие контекстов, в которых срабатывают эти приемы.

5. Найти приемы, выражающие площадь треугольника по формуле Герона. Объяснить контексты срабатывания этих приемов.

6. Найти прием проверочного оператора "усмменьшеилиравно", усматривающий неотрицательность синуса угла, заключенного между 0 и пи.

7. Найти прием для доказательства индукцией по натуральному параметру.

8. Найти приемы для вычисления суммы геометрической прогрессии.

9. Найти приемы интегрирования с помощью подстановок Эйлера.

10. Найти прием вычисления математического ожидания по известной плотности распределения случайной величины.

11. Найти все приемы вывода, теорема которых содержит символы "биссектриса" и "трапеция".

12. Найти все приемы, вводящие комментарий "промежуток" к текущему терму задачи.

Указания

1. Из главного меню системы нажатием клавиши "г" войти в оглавление приемов. Затем нажимать клавишу "курсор влево" до тех пор, пока картинка не перестанет изменяться. Это и будет корневое меню оглавления базы приемов. Дальнейший поиск ведется сообразно названиям разделов - например, к разделу "Тригонометрия" через пункт "Элементарная алгебра". Раздел "Окружность" можно найти как в подразделе "Элементарная геометрия" (через подраздел "Фигуры"), так и в разделе "Аналитическая геометрия" (через подраздел "Линии второго порядка"). Естественно, приемы последнего раздела связаны только с уравнением окружности.

2. Нужные приемы расположены в подразделе ("Элементарная алгебра", "Степени", "Решение уравнений", "Показательные уравнения", "Логарифмирование показательных уравнений"). В этом разделе имеется около десяти различных приемов. При записи нового приема в раздел он попадает в начало раздела, поэтому просмотр приемов раздела "в хронологическом" порядке нужно начинать с конца раздела, перейдя к нему нажатиями клавиши "курсор вниз" - пока картинка не перестанет изменяться.

   Наличие нескольких различных приемов логарифмирования уравнения объясняется следующими причинами. Во-первых, по-разному можно выбирать основание логарифма - иногда это основание одной из имеющихся в уравнении степеней с неизвестным показателем, иногда - основание другого логарифма, уже имеющегося в задаче, иногда, если из прочих соображений разумного основания выбрать не удается, просто константа 2. Далее, по-разному может происходить группировка членов в уравнении. Если есть всего два неизвестных слагаемых, то оба они группируются в левой части, а правая часть - нулевая. Если одно слагаемое неизвестное, а другое - известное, то они находятся в разных частях уравнения. Наконец, выделено несколько особо простых случаев, для которых предусмотрены специальные приемы, срабатывающие на малом уровне.

   Приемы, которые требуется найти в упражнении - это второй и седьмой приемы, считая с конца.

3. Приемы находятся в разделе ("Элементарная алгебра", "Неравенства", "Меньше", "Решение неравенств", "Квадратные неравенства"). Их всего шесть. Просмотр их начинаем, как обычно, с конца списка.

   Первый и второй приемы относятся к случаю, когда вычисленный дискриминант d оказался отличным от нуля, причем не удалось усмотреть его отрицательность. В первом приеме квадратный двучлен расположен справа от знака "меньше", во втором - слева от этого знака. Заметим, что известный свободный

член при стандартизации неравенств попадает в противоположную часть неравенства. Оба приема совершенно аналогичны. Иногда в таких случаях удается создать один общий прием, используя указатели идентификации. Однако, не всегда для этого хватает возможностей текущей версии ГЕНОЛОГа. С другой стороны, интерфейс предоставляет возможность быстрого "тиражирования" уже введенной версии приема - она копируется, и вводятся необходимые изменения. Таким образом могут появляться достаточно большие группы (десяток и более) однотипных приемов. Надобность в этом "тиражировании" возникает не очень часто, а на быстродействии решателя оно практически никак не отражается, так что особых стимулов в развитии указателей ГЕНОЛОГа для борьбы с ним пока не имеется.

Третий и четвертый приемы относятся к случаю, когда удалось усмотреть равенство дискриминанта нулю; пятый и шестой - к случаю отрицательного дискриминанта. Хотя во всех этих приемах дискриминат вычисляется как бы заново, но в действительности он будет вычислен лишь однократно, а затем занесен в буфер, и все последующие приемы на протяжении достаточно большого отрезка времени будут брать его прямо из буфера. Это же замечание относится и к проверке знака дискримината - результаты ее (в том числе при неудаче) тоже хранятся в буфере.

4. Приемы расположены в разделе ("Элементарная геометрия", "Фигуры", "Треугольник", "Отрезок, соединяющий вершину треугольника с точкой на противоположной стороне или ее продолжении", "Биссектрисы треугольника", "Отрезки, на которые основание биссектрисы делит противоположную сторону, пропорциональны боковым сторонам"). Всего имеется три таких приема. Первый прием (считая с конца) анализирует четыре участвующих в соотношении пропорциональности величины - длины боковых сторон и длины отрезков, на которые сторона делится биссектрисой. Если не менее трех из них выражены через числовые параметры, известные или неизвестные, то уровень срабатывания приема равен 6; в противном случае прием срабатывает, если две из них известны, но уровень срабатывания тогда повышается до 8. Второй прием срабатывает, если удается усмотреть соотношение пропорциональности с известными коэффициентами для длин боковых сторон; уровень срабатывания его равен 4. Третий прием аналогичен второму, но соотношение пропорциональности усматривается для длин отрезков, на которые сторона делится биссектрисой.

5. Приемы находятся в разделе ("Элементарная геометрия", "Фигуры", "Площадь", "Треугольник", "Формула Герона"). Число их равно трем. Первый прием срабатывает, если все три длины сторон треугольника известны, а соотношение для его площади еще не выписывалось; уровень срабатывания равен 6. Второй срабатывает, если выражения a, b, c для длин сторон либо содержат неизвестную y внешней задачи, входящую также в некоторое соотношение с площадью треугольника, либо известны, а указанная неизвестная y входит в соотношение с площадью треугольника. Наконец, третий срабатывает, если площадь треугольника известна, а выражения для длин сторон пропорциональны некоторой неизвестной.

6. Прием находится в разделе ("Элементарная алгебра", "Неравенства", "Меньшеилиравно", "Проверочный оператор УСММЕНЬШЕИЛИРАВНО", "Тригонометрические функции", "Синус"). Он является третьим с конца списка. Фи-

льтр этого приема - ускоряющий; он разрешает попытку применения приема лишь тогда, когда в посылках имеется обозначение какого - либо угла либо имеется неравенство, содержащее хотя бы одну общую переменную с аргументом синуса.

7. Прием находится в разделе ("Элементарная алгебра", "Целые числа", "Доказательство индукцией по целочисленному параметру", "Простейшая схема индукции по натуральному параметру").

8. Приемы расположены в разделе ("Элементарная алгебра", "Комбинаторные функции", "Суммавсех", "Сумма геометрической прогрессии и суммы, извлекаемые из нее почленным дифференцированием"). Первый из них (с начала списка) дает общую формулу суммирования, включающую случай равенства знаменателя прогрессии единице; второй - рассчитан на случаи, когда усматривается отличие этого знаменателя от 1.

9. Приемы расположены в разделе ("Математический анализ", "Интегралы", "Нормализатор нахождения первообразной НОРМИНТЕГРАЛ", "Замена переменной интегрирования", "Степенная функция", "Квадратичная функция под радикалом"). Для каждой из подстановок Эйлера выделен свой подраздел.

10. Прием находится в разделе ("Теория вероятностей", "Случайные величины", "Математическое ожидание", "Вычисление математического ожидания случайной величины, для которой известна плотность распределения"). Он является приемом вывода, заносящим в посылки равенство для математического ожидания. Если бы этот прием был реализован как прием замены, то при возникновении того же самого математического ожидания в той же задаче (например, при вычислении дисперсии) выкладки пришлось бы повторять.

11. Помимо оглавления базы приемов, для нахождения нужных приемов можно использовать реализованную на ЛОСе процедуру "текприем". Собственно говоря, программа этой процедуры представляет собой бланк запроса на поиск приемов, и для очередного поиска переписывается. Обращения к ней происходят в цикле полного просмотра всех приемов решателя, реализованных на ГЕНОЛОГе. При рассмотрении очередного приема процедуре сообщаются следующие данные: x1 - логический символ, за которым закреплен прием; x2 - номер узла статьи этого символа в 8-м информационном блоке, хранящем описания приемов ГЕНОЛОГа; x3 - теорема приема; x4 - заголовок приема; x5 - описание приема. Кроме того, ей сообщаются некоторые дополнительные сведения о хранении компонент описания приема в 8-м информационном блоке. Подробнее о том, как организовано это хранение, будет рассказано в следующей главе.

    Программа процедуры "текприем" должна начинаться с операторов "обращение(0) иначе 1 метка(икс(десять))", которые при написании очередного запроса на поиск изменять не следует. Заканчиваться эта программа должна оператором "ответ(набор(См))", обращение к которому вызывает прерывание и выход в просмотр описания текущего приема. После этого продолжение поиска вызывается нажатием клавиши "курсор влево". Если же нужно прервать поиск и начать работу с найденным приемом, то нажимается клавиша "Esc", выводящая в главное меню, далее нажимается клавиша "г", возвращающая в оглавление базы приемов, причем в том именно концевой пункт, где находится найденный прием, и нажимается "курсор вправо" для входа в просмотр этого приема.

В нашем случае нужно отобрать приемы, теорема которых содержит символы "трапеция" и "биссектриса". Для этого входим в просмотр программы ЛОСа для логического символа "текприем" (войти в просмотр и редактирование этой программы можно также нажатием F9 из любой точки оглавления базы приемов). Нажимаем "р" (кир.) и набираем текстовым редактором, после оператора "метка(...)", операторы "не(логсимвол(х3)) входит(трапеция х3) входит(биссектриса х3)". Затем должен быть размещен указанный выше оператор "ответ(...)". Заметим, что значением переменной х3 служит теорема приема, однако для приемов операторов фильтра вместо нее используется логический символ "блок". Чтобы избежать замечаний о некорректном обращении к оператору, перед проверкой вхождения в х3 какого - либо символа, сначала нужно убедиться, что х3 действительно является термом, что и делает оператор "не(логсимвол(х3))".

По окончании ввода программы, завершаемого, как обычно, нажатием Enter, для возвращения в оглавление базы приемов (если мы вышди из нее по F9), достаточно нажать "End". Далее поиск активируется нажатием F8 из любого пункта оглавления. В нашем случае будут найдены три приема. По завершении поиска произойдет выход в отладчик ЛОСа на операторы "трассировка(стоп 0) трассировка(0)"; далее для возвращения в главное меню нажимается Esc.

12. Как и в предыдущем случае, используем процедуру "текприем". Примечание к текущему терму задачи вводится указателем приема "примечание(...)". Поэтому в программе "текприем" помещаем единственный (кроме стандартных начала и конца программы) оператор - "входит(запись(примечание промежуток) х5)". Напомним, что х5 - набор указателей приема, к которому добавлен элемент "условие(и($F_1 \dots F_n$))", перечисляющий все фильтры приема $F_1, \dots, F_n$. При просмотре обнаруживается единственный прием, вводящий комментарий "промежуток".

Заметим, что процедуру "текприем" можно использовать не только для поиска приемов нужного типа, но и для автоматического изменения всей базы приемов, согласно заданной программе. Это позволяет предпринимать различные глобальные преобразования решателя при его оптимизации. Впрочем, после любого такого преобразования необходима полная прогонка по задачнику - для выявления остаточных случаев, требующих дополнительной проработки. Часто эти случаи возникают из-за небольших изменений траектории решения, приводящих к необходимости применять отсутствующий пока прием. Вообще, даже обычная перекомпиляция нескольких приемов не в том порядке, в каком они были скомпилированы изначально, может (хотя и достаточно редко) изменить поведение решателя. Разумеется, по мере развития исследований по взаимодействию между приемами устойчивость поведения решателя (в том числе и по отношению к случайным изменениям порядка размещения приемов после перекомпиляций) будет повышаться. Даже сейчас она не так уж мала. Однако, единичные случаи потери или замедления решений задач могут возникать, и при перекомпиляции лучше либо сохранять первоначальное размещение приемов, либо убеждаться прогонкой по задачнику, что нежелательные ситуации не возникли.

13.2.2 Просмотр описания приема

Первые упражнения этого раздела относятся к приему решения квадратного уравнения. Поэтому перед их выполнением следует найти данный прием в оглавлении базы приемов, нажать клавишу "курсор вправо", и выбрать последний из приемов списка.

1. Перейти от формульного режима просмотра теоремы к текстовому и обратно.

2. Выделить цветовой указкой в теореме приема левую часть эквивалентности. Убрать выделение, перейти в текстовый режим просмотра, и в нем снова выделить цветовой указкой данную часть.

3. Активизировать для просмотра последовательно первое, второе, третье и четвертое окна приема.

4. Убрать с экрана первое окно приема, затем восстановить его на экране.

5. Выделить многоцветной указкой фильтр "альтернатива(...)". Внутри этого фильтра выделить терм "неизвестные(2)". Прочитать пояснения к символу "неизвестные". Прочитать пояснения к подтерму "числонеизвестных(корень)", встречающемуся в фильтрах приема. Продолжить рассмотрение фильтров и указателей приема, получая информацию о всех встречающихся в них служебных словах ГЕНОЛОГа.

6. Просмотреть нормализаторы к подтерму $0 \leq e$ теоремы приема. Просмотреть все нормализаторы приема.

7. Просмотреть текстформульный шаблон пояснений к срабатыванию приема. Найти ссылки на его фрагменты из различных точек описания приема.

8. Прочитать пояснения к заголовку приема. Войти в оглавление заголовков приемов ГЕНОЛОГа и найти в нем тот же поясняющий текст.

9. Войти в оглавление типов фильтров ГЕНОЛОГа и найти в нем несколько типов фильтров, использованных в приеме. Аналогичное упражнение проделать для оглавления типов указателей приемов.

10. Из просмотра приема перейти в просмотр справочной информации о логическом символе "целое", затем вернуться обратно.

11. Перейти к просмотру программы приема. Прочитать несколько первых операторов этой программы и объяснить смысл выполняемых ими действий. Вернуться к просмотру описания приема.

12. Перейти из просмотра описания приема в оглавление задачника, затем снова вернуться к просмотру описания приема. Аналогичное упражнение - для перехода в оглавление программ.

13. Найти последний из приемов раздела "Соотношение пропорциональности длин отрезков, отсекаемых параллельными прямыми". Просмотреть все фильтры этого приема, учитывая то, что список их разбит на несколько последовательно выдаваемых на экран фрагментов.

Указания

1. Для перехода к текстовому режиму нажимаем "т", для перехода к формульному - "ф". Текущий режим будет сохранен при переходе через оглавление базы теорем к другому приему. Однако, при возвращении в главное меню установка на него будет забыта, и при последующем переходе к просмотру приемов будет, по умолчанию, выбран формульный режим.

2. Чтобы выделить цветовой указкой часть теоремы, нужно сначала инициировать ее просмотр - либо нажать левую кнопку мыши, курсор которой находится в первом окне, либо нажать клавишу "1", либо нажать два раза клавишу "курсор вправо". Первое нажатие клавиши выделит малиновым цветом номер "1" первого окна; второе - переведет в режим просмотра теоремы. Сначала малиновым цветом будет выделен первый антецедент. Используя клавишу "курсор вправо", выделим сначала весь консеквент. Затем нажмем "курсор вниз", выделяя первый операнд консеквента, что и требуется. Для сброса режима просмотра теоремы нажимаем клавишу пробела. Чтобы перейти в текстовый режим просмотра, нажимаем далее "т". После этого можно либо войти в режим просмотра теоремы двумя нажатиями "курсор вправо", и далее манипулировать клавишами курсора для выделения нужного подтерма, либо сразу подвести курсор мыши к началу нужного подтерма теоремы и нажать левую кнопку мыши. Для сброса режима просмотра снова нажимаем клавишу пробела.

3. Для входа в режим просмотра i - го окна; $i = 1, 2, 3, 4$, можно нажать клавишу "i" номера этого окна. Другой способ - нажать сначала клавишу "курсор вправо". Тогда будет выделен малиновым цветом номер "1" первого окна. С помощью клавиш "курсор вверх" - "курсор вниз" можно менять текущее окно. Однако, выделение номера окна - это еще не вход в режим просмотра окна. Для перехода к такому просмотру нужно, выделив текущее окно, нажать клавишу "курсор вправо". Заметим, что для второго окна, состоящего только из заголовка приема, надобности в просмотре с применением цветовой указки не возникает, и в этом случае нажатие "курсор вправо" игнорируется. При входе в просмотр 1,3 либо 4 окон происходит выделение первого терма цветовой указкой (для первого окна выделяется малиновым цветом первый антецедент теоремы, для 3 и 4 - светло-голубым цветом выделяется первый фильтр либо первый указатель). Для выхода из просмотра нажимается клавиша пробела.

4. Если описание приема не помещается на экране целиком, то можно убирать некоторые из окон. Для этого служат клавиши Ctr-Fi; i - номер окна, $i = 1, 2, 3, 4$. Восстановление убранных окон - нажатием той же клавиши, либо за счет выхода в оглавление базы приемов и повторного входа в просмотр приема.

5. Есть две возможности - либо выделить слово "альтернатива" курсором мыши и нажать ее левую кнопку, либо войти в просмотр третьего окна и выделить фильтр "альтернатива(...)", используя клавишу "курсор вправо". Подтерм "неизвестные(2)" теперь можно выделить, нажав клавишу "курсор вниз", либо переведя курсор мыши на слово "неизвестные" и нажав ее левую кнопку. Мышью можно было выделить этот подтерм и сразу, не выделяя "альтернатива(
...)".

Чтобы прочитать пояснения к выделенному терму, нажимаем либо правую кнопку мыши (безотносительно к тому, использовалась ли мышь при выделении), либо клавишу "Ctr-y" (кир.). Снизу, под четвертым окном, появляется текст пояснений. Иногда одно и то же служебное слово используется в различных контекстах, и тогда можно просмотреть пояснения ко всем возможным его использованиям. Сначала выдается какой - то один текст; для смены текстов списка используются клавиши "курсор вверх" - "курсор вниз". В нашем случае сначала будет выдан текст пояснений к использованию "неизвестные(x1)" в качестве фильтра, уточняющего число неизвестных задачи. После нажатия "курсор вниз" появится другой текст - "неизвестные(x1)" может служить обозначением списка неизвестных задачи. Обычно такая неоднозначность легко устраняется из контекста. Очевидно, "неизвестные(2)" подпадает под первый текст пояснений.

Чтобы убрать с экрана тексты пояснений, нажимаем либо произвольную кнопку мыши, либо клавишу пробела. После этого выделяем слово "числонеизвестных" курсором мыши и нажимаем правую ее кнопку. Сначала появится текст "Число неизвестных задачи"; после нажатия "курсор вниз" - "Число неизвестных в терме, определяемом выражением x1". Так в нашем случае символ "числонеизвестных" относится к выражению "корень", то берем второй поясняющий текст. Можно посмотреть пояснения к слову "корень" - будут выданы тексты "Заменяемый подтерм является корневым" и "Текущий терм задачи". Первый из них явно относится к случаю использования слова "корень" как фильтра, у нас же оно используется как выражение. Поэтому берем второй текст. Окончательно получаем, что "числонеизвестных(корень)" означает число неизвестных в текущем терме задачи, т.е. в рассматриваемом квадратном уравнении.

Чтобы привыкнуть к простейшим конструкциям ГЕНОЛОГа, можно рекомендовать последовательный просмотр фильтров и указателей приема с выделением их левой клавишей мыши и вызывом сопровождающих текстов правой клавишей.

6. Чтобы просмотреть нормализаторы к заданному вхождению в теорему, фильтр либо указатель, сначала нужно выделить это вхождение. Заметим, что не все вхождения в теорему могут быть выделены при формульном режиме. Поэтому в сомнительных случаях (они бывают сравнительно редко и связаны с различными специальными логическими конструкциями) лучше переходить в текстовый режим просмотра теоремы. В нашем примере можно выделить $0 \leq e$ и непосредственно из формульного просмотра.

   После того, как нужное вхождение выделено, нажимается Enter. Тогда под четвертым окном возникает текст нормализаторов, относящихся к выделенному вхождению. При этом появляется также курсор текстового редактора, чтобы при необходимости изменить нормализаторы. Если вхождение не сопровождалось нормализаторами, то курсор текстового редактора возникает на пустой строке. Так как в данном упражнении не нужно изменять нормализаторы, то после просмотра нажимается Esc либо Enter (последнее - только при отсутствии изменений в тексте).

   Чтобы последовательно просмотреть все нормализаторы приема, следует нажать клавишу "5", и далее менять текущий выделенный подтерм клавишами

"PageUp" - "PageDown". Как и ранее, нормализаторы будут прорисовываться под четвертым окном. Выход из такого просмотра - только по нажатию "End" (!).

7. Просмотр текстформульного шаблона вызывается нажатием клавиши "6". Этот шаблон оказывается состоящим из двух фрагментов: "¬1 Решаем квадратное уравнение относительно (). Дискриминат равен ()"; "¬2 Разложение на множители дискриминанта". Очевидно, первый из них комментирует срабатывание приема "в целом", два последних - комментируют вспомогательные преобразования, выполняемые приемом. Для первого легко находится относящийся к нему указатель "титр(набор(1 терм(x4 x5)))". Согласно этому указателю, вместо первых скобок будет подставляться неизвестное выражение d, вместо вторых - результат e вычисления дискриминанта. Второй текст указывает на обращение к нормализатору разложения на множители дискриминанта $b^2 + 4ac$. Выделяя в теореме его вхождение, просматриваем список нормализаторов. В этом списке находим элемент "видумножение(замечание(нормстепень) замечание(новый) замечание(титр(набор(2))))", содержащий ссылку на второй фрагмент.

8. Для просмотра пояснений к заголовку нужно выделить номер "2" окна заголовка малиновым цветом и нажать "Ctr-з" (кир.). Пояснение появится под четвертым окном. Для входа в оглавление типов заголовков нажимаем ту же клавишу "Ctr-з", но без выделения второго окна приема. Находим корень оглавления, и от него прослеживаем цепочку переходов ("Замена подтерма", "Замена слева направо"). Нажимая далее "курсор вправо", находим соответствующий заголовок "второйтерм".

9. Для входа в оглавления типов фильтров и типов указателей используем, соответственно, клавиши "Ctr-у" (кир.) и "Ctr-х". Заметим, что эти оглавления немного более полны, чем приведенные в книге списки типов фильтров и указателей, так как они постоянно пополняются в процессе развития языка.

10. Для получения общей справочной информации о логическом символе нажимаем клавишу "с", и далее вводим текстовым редактором этот логический символ. После нажатия Enter, завершающего ввод символа, появляется требуемая информация. По нажатии любой клавиши, отличной от используемых для просмотра информации служебных клавиш (например, при нажатии пробела) - возвращение в просмотр описания приема.

11. Для просмотра программы приема нажимаем клавишу Home. Сразу после этого появляется текст фрагмента программы, содержащий ключевой оператор "контрольприема(...)", адресующий данный прием. В нашем случае это оператор "контрольприема(степень набор(1 6 0) второйтерм)". Здесь логический символ "степень" и число 160 - координаты описания приема в восьмом информационном блоке. Заметим, что хотя прием закреплен в этом блоке за символом "степень", программа его закреплена за символом "плюс", и попытка применения приема будет предприниматься при обнаружении вхождения в терм задачи последнего символа.

    Напомним, что при сканировании задачи имеются следующие стандартные значения программных переменных: x1 - задача; x2 - вхождение текущего логического символа; x3 - вхождение текущего терма задачи (посылки либо условия) в

соответствующий список задачи; x4 - указатель посылки (0) либо условия (1); x5 - текущий уровень сканирования. Переменные, начиная с переменной x6, изначально не определены. Однако, до того момента, как программа обратилась к фрагменту, содержащему оператор "контрольприема(...)", уже могли быть предприняты некоторые проверки и присвоения. Поэтому, для анализа той ситуации, которая сложилась на момент входа в фрагмент, прежде всего нужно просмотреть цепочку предшествовавших ему фрагментов. Это делаем, используя клавиши курсора: "курсор вверх" - к надфрагменту; "курсор влево - курсор вправо" - выделение желтым цветом того оператора перехода в текущем фрагменте, через который требуется войти в подфрагмент; "курсор вниз" - вход в подфрагмент.

В нашем случае, нажав "курсор вверх", оказываемся в фрагменте, имеющем начальные операторы "ветвь 1 уровень(1 2) новый равно(x4 1) ветвь 2 тип(x1 описать) равно(второйоперанд(x2) последнийоперанд(x2)) ветвь 3", причем через "ветвь 3" переходим обратно к начальному фрагменту приема. Эти несколько операторов оказались общими у нашего приема с другими приемами, и они вынесены наружу. Оператор "уровень(1 2)" проверяет равенство текущего уровня сканирования единице либо двойке; оператор "новый" блокирует попытки применения приема при локальных циклах повторного сканирования, предшествующих переводу терма задачи из "теневой" зоны в активную; оператор "равно(x4 1)" проверяет, что символ "плюс" встретился в условии задачи. Оператор "тип(x1 описать)" фиксирует тип текущей задачи; оператор "равно(второйоперанд(x2)последнийоперанд(x2))" - проверяет, что сумма имеет два слагаемых. Одно из них соответствует квадрату неизвестного выражения, другое - его первой степени. Если уравнение изначально имело несколько членов с квадратом либо с первой степенью, то после срабатывания приемов приведения подобных членов с известными коэффициентами слагаемых останется ровно два.

Можно было бы просмотреть и фрагменты, предшествующие данному, однако в них ничего существенного не происходит - идет "главный ствол" программы символа "плюс", ответвления от которого соответствуют различным значениям уровня срабатывания. Поэтому возвращаемся через "ветвь 3" в начальный фрагмент приема. Пропуская фиктивный оператор "контрольприема", переходим к следующим операторам. Оператор "равно(x6 неизвестные(x1))" присваивает переменной x6 цель задачи x1, перечисляющую ее неизвестные. Эта цель имеет вид (неизвестные $x_1 \dots x_n$). Оператор "альтернатива(не(длинаменее(x6 3)) равно(x5 1) равно(x5 2))" проверяет, имеет ли задача одну неизвестную - тогда длина набора x6 будет меньше 3, или больше одной - тогда эта длина будет не менее 3. Соответственно, в первом случае текущий уровень сканирования должен быть равен 2, а во втором случае - 1.

Оператор "операнд(x7 x2)" переводит от текущего символа "плюс" к внешнему символу, присваивая переменной x7 вхождение этого символа. Оператор "символ(x7 равно)" проверяет, что по вхождению x7 расположен символ "равно". Оператор "корень(x7)" проверяет, что вхождение x7 - корневое, т.е. рассматриваемое условие имеет вид равенства, и т.д.

По окончании анализа программы можно вернуться в просмотр описания приема, нажав клавишу End. Заметим, что нажатие End приведет к этому результату вне зависимости от того, в каком фрагменте программы символа "плюс"

она была нажата. Если при просмотре окрестности начального фрагмента программы нужно быстро вернуться к этому начальному фрагменту, то нажимается клавиша F8.

12. Для перехода к оглавлению задачника нажимаем Shift-3. Чтобы вернуться оттуда в оглавление приемов, нажимаем Shift-2. Для перехода к оглавлению программ далее нажимается Shift-1; обратно - снова Shift-2.

13. Прием находится в разделе ("Элементарная геометрия", "Параллельны", "Пропорциональность отрезков, отсекаемых параллельными прямыми", "Соотношение пропорциональности длин отрезков, отсекаемых параллельными прямыми"). Нажимая "3", входим в просмотр окна фильтров. Перемещая с помощью клавиши "курсор вправо" выделенный текущий терм, доходим таким образом до запятой, и снова нажимаем "курсор вправо". В результате появляется второй фрагмент списка фильтров. Проходим через него аналогичным образом, и попадаем в третий, последний фрагмент. Обратные переходы - с помощью "курсор влево". Этот процесс можно ускорить, используя мышь: нажатие ее левой кнопки на запятой приводит к переходу через эту запятую - вперед либо (для запятой в начале фрагмента) назад.

13.2.3 Редактирование приема

На нескольких простых упражнениях продемонстрируем технику ввода нового приема, изменения старого, а также тестирования приема после редактирования. Для ввода новых приемов следует сначала создать специальный концевой пункт оглавления базы приемов, разместив его там, где покажется удобнее (можно и в соответствии с имеющейся классификацией разделов). При первом знакомстве с редактором приемов лучше не размещать учебные приемы в "старых" концевых пунктах, чтобы случайно не удалить имеющиеся там приемы. Впрочем, такие случайности не очень страшны - первоначальная версия удаленного либо измененного приема попадает в буфер, откуда ее можно вернуть обратно.

1. Ввести прием, упрощающий тригонометрические выражения с помощью тождества для суммы квадратов синуса и косинуса. Сначала набрать теорему приема $\forall_{ax}(a(\sin x)^2 + a(\cos x)^2 = a)$. Затем ввести заголовок приема "второйтерм". Далее ввести единственный фильтр - "уровень(0)". Ввести указатели "единица(1 x1)" и "заменазнака(минус x1)". Сохранить введенный прием и откомпилировать его.

2. Проверить, что прием работает, введя какую-либо тестовую задачу на упрощение, например, $2(\sin b)^2 + 2(\cos b)^2$, и найдя в трассировке по шагам решения задачи срабатывание введенного приема. Войти в просмотр описания приема из текущего шага трассировки. Перейти к просмотру текущего фрагмента программы и найти программные переменные, значениями которых служат: а) вхождение косинуса; б) вхождение операции сложения; в) терм, идентифицированный с квадратом синуса.

3. После тестирования вернуться в редактор приемов и проделать следующие действия: а) удалить программу приема, не удаляя самого приема; б) восстановить

программу приема; в) добавить фильтр "тип(преобразовать)" и сохранить измененный прием, не изменяя пока его программу; г) еще раз перекомпилировать прием, причем провести компиляцию так, чтобы при тестировании можно было определять те термы, с которыми идентифицируются переменные теоремы приема.

4. После выполнения предыдущего упражнения снова войти в тестирование приема и, остановившись на моменте его применения, найти идентифицированные с теоремными переменными термы, не переходя в отладчик ЛОСа.

5. Создать в оглавлении базы приемов какой-нибудь новый концевой пункт и перенести в него созданный прием из того концевого пункта, где он находился до этого. Старый концевой пункт (если он после этого оказался пустым) удалить.

6. Сбросить буфер базы приемов. Затем удалить введенный прием, после чего вернуть его на старое место из буфера базы приемов. Снова сбросить буфер базы приемов. Изменить прием, убрав из него фильтр "тип(преобразовать)". Посмотреть старую версию приема в буфере. Восстановить по буферу версию приема, имевшуюся до изменения.

7. Создать копию приема, изменив в ней фильтр "уровень(0)"на "уровень(1)".

8. Создать простую версию приема, использующего теорему Пифагора. Сначала изобразить на чертеже прямоугольный треугольник ABC с прямым углом B. Ввести теорему приема:

$$\forall_{ABC}(\text{прямая}(AB)\perp\text{прямая}(BC) \rightarrow (l(AC))^2 = (l(AB))^2 + (l(BC))^2).$$

Затем ввести заголовок приема "вывод" и фильтры "уровень(1)", "посылка", "тип(исследовать)", "лексикопредшествует(х26 х28)", "усм(актив(расстояние(х26 х28)))", "усм(актив(расстояние(х26 х27)))", "усм(актив(расстояние(х27 х28)))". Ввести указатели приема: "усм(1)", "теквхожд(1)", "биключ(перпендикулярно х27 интервал(х26 х28))". Ввести нормализаторы общей стандартизации для обработки подтермов выводимого утверждения. Ввести нормализатор "норминтервал" для обработки терма "интервал(х26 х28)", встречающегося в указателе. Сохранить прием и откомпилировать его.

9. Разбить список фильтров приема на две части, оборвав первую часть на фильтре "лексикопредшествует(х26 х28)". Найти в оглавлении базы приемов старый прием на теорему Пифагора, срабатывающий на уровне 10, и перенести из него в только что созданный прием фильтр "контекст(посылка(х1) заголовок(х1 равно) вхождениетерма(х1 расстояние(х26 х28)))".

10. Ввести простую геометрическую задачу на нахождение длины одного из катетов прямоугольного треугольника, у которого длина гипотенузы равна 5, а другого катета - 4. Протестировать на ней созданный прием, установивши прерывание при его применении.

11. Удалить все нормализаторы приема, после чего ввести их в режиме автоматического пополнения описания приема.

12. Ввести прием для сложения дробных выражений. Теорема приема имеет вид:

$$\forall_{abcdef}(\neg(c=0)\ \&\ \neg(d=0)\ \&\ \neg(f=0) \to \frac{ab}{cd}+\frac{ae}{cf}=\frac{a(bf+de)}{cdf}).$$

Заголовок приема - "второйтерм"; фильтры - "уровень(1)", "условие", "тип(преобразовать)", "или(заголовок(фикс(0 1 1)дробь) и(заголовок(фикс(0 1 1)минус) первыйсимвол(фикс(0 1 1)дробь)))", "или(заголовок(фикс(0 1 2)дробь) и(заголовок(фикс(0 1 2) минус) первыйсимвол(фикс(0 1 2) дробь)))". При вводе фильтров использовать автоматическое выписывание в третьем окне приема указателей вхождения "фикс(...)" после выделения этих вхождений в теореме цветовой указкой, заметив, что "фикс(0 1 1)", "фикс(0 1 2)" здесь ссылаются на вхождения первого и второго дробных слагаемых. Смысл этих фильтров в том, чтобы при обращении в единицу некоторых переменных из знаменателей не пропали дроби. Указатели приема - "блокпроверок(1 2 3)", "единица(1 x1 x2 x3 x4 x5 x6)", "заменазнака(минус x2 x5)". Снабдить прием указателями нормализации, причем сумму в числителе результата снабдить, кроме обычного нормализатора "нормплюс", также нормализатором "видумножение", выполняющим попытку разложения на множители. Ввести в прием какое-либо текст-формульное сопровождение, поясняющее срабатывание приема. Создать тестовую задачу для приема, выйти в точку срабатывания при трассировке и убедиться в том, что пояснение выдается.

13. Удалить все приемы, введенные в перечисленных выше упражнениях.

Указания

1. Выберем для нового приема какой-нибудь подходящий раздел. Например, в подразделе "Тригонометрия" раздела "Элементарная алгебра" выберем подраздел "Синус", далее - "Общая стандартизация выражений", и здесь введем новый концевой пункт, нажав клавишу "к". При появлении курсора текстового редактора нужно набрать какое-либо название пункта; пусть это будет "Тест". Далее нажимаем "курсор вправо" и попадаем в чистый экран - обычное начало ввода приема. Для набора теоремы формульным редактором нажимаем Ctr-ф. Возникает курсор формульного редактора, и далее набираем указанное выше тождество, не забыв начать его с квантора общности и перечислить в кванторной приставке все переменные теоремы. Напомним, что неполное перечисление переменных в кванторной приставке может, хотя и редко, привести к неправильной компиляции. Если переменная оказывается связана внутри теоремы, то она в общую кванторную приставку не выносится. Если при наборе формулы нужно вспомнить какую-либо клавишу формульного редактора, нажимается F1, переводящее в справочник по системе, и нужная информация находится в подразделе "Формульный редактор" корневого меню справочника. Чтобы продолжить набор формулы, далее нажимается End (сначала - для возвращения в оглавление справочника, затем - для возвращения в набор теоремы).

   По завершении набора теоремы нажимается Enter, и под теоремой прорисовывается горизонтальная линия. Далее нажимается Enter, и во втором окне набирается заголовок приема. После того, как заголовок введен и нажато Enter, появляется следующая горизонтальная линия, причем сразу же под ней появляется курсор текстового редактора для набора фильтров. Вводим фильтр "уро-

вень(0)"; после завершающего набор нажатия Enter вводим указатели "единица(1 x1)" и "заменазнака(минус x1)", и снова нажимаем Enter. Под четвертым окном появляется голубая линия, указывающая на то, что новый либо измененный прием еще не сохранен. Чтобы сохранить его без немедленной компиляции (что естественно делать, если прием еще не полностью проработан), нажимается F4, и тогда голубая линия меняет цвет на красный. В нашем случае сохраняем прием и одновременно компилируем его, нажимая F3.

2. Переходим из просмотра приема в задачник - либо через главное меню, либо за один переход, нажимая Shift-3. В задачнике выбираем какой-либо подходящий раздел, например, ("Элементарная алгебра", "Упрощение выражений", "Тригонометрические выражения", "Упрощение выражений - 2"). Прокрутив меню номеров пунктов этого оглавления вверх так, чтобы внизу появилось свободное место, нажимаем "к" (кир.). Тогда появляется поле для ввода новой задачи, ограниченное сверху горизонтальной линией. Нажимаем "ц" для входа в оглавление типов целевых установок, выбираем пункт ("Преобразовать выражение", "Упростить выражение в области допустимхы значений") и нажимаем "курсор вправо". Далее вводим упрощаемое выражение $2(\sin b)^2 + 2(\cos b)^2$ - сначала нажимаем Enter, и используем формульный редактор. После набора этого выражения начинаем трассировку процесса решения. Здесь имеются две возможности. Одна - начать нажатием "р" (кир.) и просмотреть все шаги решения задачи, нажимая каждый раз Enter для перехода к следующему шагу. В нашем тривиальном случае это сразу выведет на срабатывание тестируемого приема. Другая возможность - сразу установить прерывание при срабатывании нужного приема. Для этого нужно сначала нажать "г" - появится оглавление базы приемов, затем войти в просмотр нужного приема, и нажать либо Enter, либо Ctr-Enter. В первом случае прерывание произойдет после срабатывания приема, во втором - при выходе на начальный оператор "контрольприема(...)" программы приема, причем здесь включается отладчик ЛОСа и устанавливается пошаговый режим трассировки. При отладке применяется именно второй способ, так как он позволяет понять причины несрабатывания приема. Мы нажимаем Enter.

   Чтобы войти из просмотра текущего преобразования в просмотр описания примененного приема, нажимается клавиша "б". Это делается не только для приемов, реализованных на ГЕНОЛОГе, но и для приемов, запрограммированных непосредственно на ЛОСе. В первом случае появится описание приема, во втором - пункт оглавления программ, соответствующий примененному приему. Чтобы вернуться в просмотр текущего преобразования, нажимаем End. Отсюда можно выйти в отладчик ЛОСа для просмотра текущего фрагмента программы - нажатием клавиши "ф".

   Чтобы теперь проанализировать значения программных переменных, нужно помнить смысл первых пяти переменных, определенных при сканировании задачи: x1 - текущая задача; x2 - вхождение в нее текущего символа (название его прорисовано в верхней левой части экрана); x3 - вхождение текущего терма задачи (посылки либо условия) в его внешний список; x4 - указатель посылки (0) либо условия (1); x5 - текущий уровень сканирования. Сразу ясно, что вхождение косинуса является значением переменной x2 - так как прием инициализируется при обнаружении косинуса. Для получения предыстории текущего фрагмента нажимаем Home - переходим в просмотр фрагмента, из которого по

"иначе 2" перешли в данный фрагмент. Замечая оператор "операнд(x6 x2)", запоминаем, что значением переменной x6 является вхождение внешней по отношению к косинусу операции. После этого нажимаем End и возвращаемся в первоначально выданный на экран фрагмент программы. Видно, что в начале этого фрагмента расположены операторы "символ(x6 степень)", "равно(x2 первыйоперанд(x6))", фиксирующие размещение косинуса в основании некоторой степени. Можно не читать программу оператор за оператором, а сразу искать оператор "символ(... плюс)", находящий корневую сумму преобразуемого выражения. Такой оператор имеется - это "символ(x9 плюс)", так что вхождение суммы является значением переменной x9. Аналогичным образом, не читая подробным образом всего текста программы, обнаруживаем операторы "заголовок(x19 степень)", "первыйсимвол(x19 синус)", что и дает нам программную переменную x19, значением которой служит терм $(\sin x)^2$.

3. Для обрыва трассировки нажимаем Esc, возвращаясь к просмотру исходного условия задачи. Возвращаемся в базу приемов - либо через главное меню, либо выйдя в оглавление задачника и нажа Shift-2. Переходим к просмотру нашего приема. Чтобы удалить его программу, не удаляя приема, достаточно нажать F7. Нижняя горизонтальная линия перекрасится из черного цвета в красный - сигнал об отключении приема. Для восстановления программы нажимаем F5. Надо заметить, что время на восстановление программы приема, затрачиваемое компилятором ГЕНОЛОГа, совсем невелико, и если сравнивать его со временем решения иных задач средней сложности, то станет ясно, что во многих случаях можно было бы прибегать к помощи компилятора и генератора приемов "в реальном времени", практически не замедляя хода решения задачи. Это означает, что по мере развития техники автоматического синтеза приемов можно будет сохранить в базе приемов лишь малую ее часть - наиболее часто работающие приемы, а остальные создавать по мере надобности на основе базы теорем для одноразового применения.

   Для вставки нового фильтра нажимаем Ctr-3 (это легко запомнить - Ctr и номер изменяемого окна, за исключением случая первого окна). В третьем окне открывается текстовый редактор; с его помощью вводим фильтр "тип(преобразовать)". В данном приеме этот фильтр, конечно, совсем не нужен, и вводится здесь лишь для знакомства с редактором приемов. Чтобы сохранить измененное описание приема, не изменяя его программы, нажимаем F4. Заметим, что при этом нижняя линия приема перекрасится из голубого цвета в черный - так как программа приема все-таки есть, хотя это и старая версия. Для приведения программы приема в соответствие с его описанием нужно было бы нажать F5. Однако, в упражнении требуется применить другой режим компиляции, после которого просмотр термов, идентифицированных с теоремными переменными, станет возможен без помощи отладчика ЛОСа. Такая компиляция включается нажатием клавиши F6. Внешне она ничем не отличается от обычной, но в специальном информационном блоке запоминается соответствие между теоремными и программными переменными, устанавливаемое компилятором. Если впоследствии перекомпилировать прием с помощью F5, то эта информация сбрасывается.

4. Начнем с того момента, как при трассировке появляется кадр преобразования, выполняемого нашим приемом. Заметим, что в отсутствии специально создан-

ного текст-формульного сопровождения, при срабатывании приема будет выдаваться текст того концевого пункта оглавления базы приемов, где размещен прием. Этот текст заключается в скобки и располагается после описания преобразования.

Для входа в просмотр описания примененного приема нажимаем "б", и после прорисовки приема нажимаем клавишу "п" ("переменные"). Тогда все окна приема, кроме первого, удаляются с экрана, и под теоремой прорисовывается равенство $a = 2$ - т.е. теоремная переменная a идентифицирована с двойкой. Чтобы перейти к просмотру следующей переменной, нажимаем PageDown - появляется равенство $x = b$, т.е. теоремная переменная x идентифицирована с термом b. Для возвращения к ранее просмотренным переменным используется клавиша PageUp. Для обрыва просмотра нажимаем любую другую клавишу, например, пробел. Описанная возможность уточнения идентификации переменных при срабатывании приема - отладочная и обычно дополняется информацией, предоставляемой отладчиком ЛОСа. При наличии подробного поясняющего срабывание приема текста она становится излишней. По окончании упражнения выходим из трассировки и возвращаемся в просмотр приема.

5. Чтобы перенести прием в другое место оглавления базы приемов, сначала нажимаем Insert. Это можно делать, даже если тот концевой пункт, куда предполагается перенести прием, еще не создан. Например, выйдя в оглавление, переходим в раздел ("Косинус", "Общая стандартизация выражений"), и вводим новый концевой пункт "Тест" в этом разделе. Введя его и выделив голубым цветом, но не входя внутрь, снова нажимаем Insert. Теперь можно зайти внутрь пункта - наш прием окажется там. Из старого пункта он при этом удаляется. Возвращаемся к старому пункту "Тест", проверяем, что он пуст, выделяем его в оглавлении и нажимаем для удаления Ctr-Del.

6. Для сброса буфера базы приемов нужно выйти в оглавление этой базы и нажать "О" (кир.). Если текущие пункты оглавления относились к сбрасываемому буферу, то после сброса буфера произойдет выход в главное меню, иначе - сохранится текущий пункт оглавления. Обычно буфер сбрасывается тогда, когда все новые или измененные приемы уже протестированы - например, предпринята прокрутка по тем разделам задачника, где они могут проявить активность. После сброса буфера возвращаемся в просмотр приема, и для удаления его нажимаем Ctr-Del. Прием будет удален, программа его - тоже. Однако, исходная версия удаленного приема будет сохранена в буфере базы приемов. Чтобы перейти к ней, нужно выйти в оглавление базы приемов и нажать "ф". Буфер представляет собой автоматически вводимую ветвь корневого меню оглавления базы приемов. Нажатие "ф" переводит в некоторый раздел этой ветви. В корневой части ветви находим раздел "Удаленные приемы", в нем - подраздел "Тест". Входим в подраздел и обнаруживаем в нем только что удаленный прием, причем нижняя его линия - красная, так как программа приема отсутствует. Далее обычными средствами переносим прием на старое место - сначала нажимаем Insert в описании данного приема, затем выходим в оглавление, находим в разделе "Косинус" пустой теперь концевой пункт "Тест", и снова нажимаем Insert, выделив этот пункт.

   Чтобы еще раз сбросить буфер, перед входом в просмотр описания приема нажимаем "О".

Войдя в просмотр описания приема, нажимаем Ctr-З и удаляем текстовым редактором фильтр "тип(преобразовать)". Затем сохраняем прием и компилируем его, нажав клавишу F3. Чтобы просмотреть старую версию, сохраненную в буфере, далее нажимаем "б". Появляется старая версия приема, у которй нижняя линия - красная (программы это версии нет). Чтобы вернуться от нее к текущей (основной) версии, нажимаем "о" (кир.). Наконец, для восстановления старой версии приема нажимаем "Б". Новая версия уничтожается; старая извлекается из буфера, компилируется и переносится на ее место. Нужно предупредить, что такое восстановление старой версии приема не гарантирует восстановления исходного размещения ее программы. Обычно компилятор присоединяет ветвь новой программы, при прочих равных условиях, после ранее введенных ветвей. Если точка входа в программу измененного приема располагалась в середине некоторого линейного списка точек входа в другие приемы, то после перекомпиляции она попадет в конец этого списка. Чаще всего, это несущественно, однако иногда имеются "состязания" между приемами одного списка, и тогда перекомпиляция может существенно повлиять на поведение решателя в отдельных задачах (не всегда, впрочем, в худшую сторону). В особых случаях после перекомпиляции можно переставить ветви программ вручную, используя редактор ЛОСа, хотя более радикальный способ здесь - обычное уточнение решающих правил приемов.

7. Чтобы создать копию приема, нажимаем из просмотра его клавишу Ctr-д. Появляется тот же самый текст описания, но нижняя его линия - синия. На самом деле, это уже копия приема. Сначала сохраняем ее без компиляции (F4). Затем нажимаем Ctr-З, входим в редактирование окна фильтров и изменяем первый фильтр на "уровень(1)". Далее сохраняем прием и компилируем (F3). Заметим, что можно было откомпилировать копию приема и не изменяя его - тогда программы приемов склеятся вплоть до последних операторов, выполняющих преобразование, а перед ними произойдет ответвление одной программы от другой.

8. Прием поместим в том же самом разделе "Тест", что и предыдущий. Для инициализации его нажимаем "Ctr-п" (когда раздел был пустой, нажатия специальной клавиши для инициализации приема не требовалось - можно было сразу вводить теорему). Появляется чистый экран, и начинается ввод компонент приема. Заметим, что если оборвать набор приема до его сохранения и выйти, например, в оглавление либо даже в просмотр соседних приемов того же самого концевого пункта (это делают клавиши "курсор вверх", "курсор вниз"), то вся работа по набору начальной части приема пропадет - как если бы клавиша Ctr-п не нажималась вовсе.

   В нашем случае новым является наличие чертежа. Он вводится до набора теоремы. Нажимаем Ctr-ч, после чего появляется синяя рамка геометрического редактора. Выбираем курсором мыши в верхней левой части экрана пункт меню "Точка" и нажимаем левую кнопку мыши. В правом верхнем окне рамки редактора возникает указатель режима "Точка". Теперь можно вводить точки чертежа. Выбираем подходящее место для вершины A треугольника и нажимаем левую кнопку мыши. Кроме точки, появляется ее обозначение, автоматически вводимое редактором - берутся последовательные большие буквы латинского алфавита. Далее выбираем место для вершины прямого угла B и место для

вершины C, нажимая каждый раз левую кнопку мыши. Введя точки, включаем режим "Отрезок" (см. меню в верхней части экрана). Далее проводим стороны треугольника: подводим курсор к точке A и нажимаем левую кнопку мыши; затем подводим курсор к точке B и снова нажимаем левую кнопку мыши. Появляется отрезок AB. Таким же образом создаем отрезки AC и BC. Если точки легли не очень удачно, то выбираем ту из точек, которую желательно сдвинуть - опять нажатием левой кнопки мыши. Для сдвига используем клавиши курсора - каждое нажатие сдвигает выделенную точку на один пиксел в соответствующем направлении. Чтобы одновременно сдвигать и букву, обозначающую точку, аналогичным образом используем клавиши "Ctr-курсоры". Наконец, чертеж сдвигаем в верхнюю часть экрана, чтобы как можно больше места оставить для прочих компонент описания приема. Для этого включаем режим "перемещение". Перемещение производится указанием начала и конца вектора перемещения - нажатиями левой кнопки мыши. Этот вектор можно откладывать от любой точки на экране, не связывая ее с чертежом. После указания конца вектора смещается все изображение, причем если места для сдвига не хватает, то он блокируется. Для инициализации нового сдвига - повторное указание его вектора. Ввод чертежа завершается нажатием Enter, и сразу возникает курсор текстового редактора для ввода теоремы.

Все остальные шаги по вводу приема аналогичны первому разобранному примеру - только при вводе нормализатора "норминтервал" придется выделять цветовой указкой не вхождение в теорему, а вхождение в указатель "биключ(...)" из четвертого окна приема. Когда это вхождение выделено, нажимается Enter и вводится нормализатор.

9. Чтобы разрезать список фильтров на две части, выделяем нажатием левой кнопки мыши фильтр "лексикопредшествует(x26 x28)" и нажимаем "р" (кир.). После данного фильтра появляется запятая, а оставшиеся фильтры переносятся "на следующую страницу". Для перенесения фильтра из другого приема сначала находим последний - это второй от конца прием раздела "Теорема Пифагора", расположенного в конце цепи ("Элементарная геометрия", "Фигуры", "Треугольник", "Прямоугольный треугольник"). Выделяем нужный фильтр "контекст(...)" левой кнопкой мыши и нажимаем "у" (кир.). Затем клавишей пробела сбрасываем выделение. Возвращаемся в просмотр "своего" приема и нажимаем клавишу "у" повторно, но не выделяя каких-либо окон или фильтров. Тогда фильтр "контекст(...)" автоматически добавится к списку фильтров рассматриваемого приема. Так как наш список фильтров разрезан на две части, а новый фильтр заносится в конец списка фильтров, то на экране он не появится, пока не будет вызвана для просмотра последняя "страница" списка фильтров. Чтобы сохранить добавленный фильтр и перекомпилировать прием, лучше до этого нажать F3.

10. Вводим посылки задачи - "прямая(AD) $\perp$ прямая(DC)"; $l(AC) = 5$; $l(AD) = 4$. Входим в меню выбора целевой установки и выбираем пункт ("Найти значения неизвестных", "Выразить значения неизвестных через заданные параметры"). После подсказки "Выразить" набираем x и нажимаем Enter. Так как известных параметров в задаче нет, подсказку "через" игнорируем и сразу нажимаем Enter. Под целевой установкой набираем условие: $x = l(CD)$. Чертеж вручную строить не нужно - для его автоматического создания нажимаем Ctr-ч. На

точку срабатывания приема попадаем так, как это уже объяснялось выше.

11. Для удаления нормализаторов входим в просмотр описания приема и нажимаем Ctr-End. Чтобы восстановить эти нормализаторы, нажимаем "н" (кир.). Тогда включается режим ручного анализа последовательности предложений, вносимых генератором приемов. Здесь могут появляться предложения на добавление либо удаление нормализаторов и указателей. Если предлагается добавить элемент, то первый пункт меню в левой верхней части экрана имеет вид "Новый указатель", иначе - вид "Старый указатель". Чтобы ввести новый элемент либо сохранить старый, нужно нажать Insert; чтобы отказаться от нового элемента либо удалить старый - нажать Delete. В нашем случае последовательно появляются нормализаторы, так что естественно все время нажимать Insert - пока предложения не будут исчерпаны. После этого нажимаем F3.

12. Ввод теоремы приема сложения дробных выражений и его заголовка происходит обычным образом. При наборе фильтра "или(заголовок(...)...)" после открывающей скобки символа "заголовок" нажимаем клавишу Shift-1, приводящую к появлению цветовой указки в первом окне. Выбираем этой указкой первое слагаемое и нажимаем клавишу "ф". Сразу вслед за этим в третьем окне начиная с текущей позиции курсора будет прорисован указатель вхождения "фикс(0 1 1)", а цветовая указка в первом окне пропадет. Остальные указатели вхождения вводятся таким же образом. Нормализаторы приема можно создать частично с помощью клавиши "н", а частично - добавить вручную.

    Для ввода текст-формульного шаблона нажимаем клавишу "6" и вводим, например, следующий текст: "Выражение () равно сумме дробей () и ()". Чтобы первая буква слова "Выражение" не пропала при выдаче на экран, отступаем при наборе этого слова от левого края экрана на один символ. По завершении набора нажимаем Enter. Если понадобится изменить текст шаблона, то снова нужно будет нажать "6", затем - Enter для входа в текстовый редактор.

    Чтобы определить выражения, подставляемые в шаблон вместо скобок, вводим указатель "титр(набор(1 терм(фикс(0 2))терм(фикс(0 1 1))терм(фикс(0 1 2))))", в котором указатели вхождений выделяют, соответственно, правую часть тождества и вхождения дробных слагаемых. Здесь снова применяется описанный выше интерфейс полуавтоматического набора указателей вхождения. Для тестирования приема вводим, например, задачу на упрощение выражения $a/3b + c/b^2d$. При срабатывании приема, кроме поясняющего текста, появятся также сообщения о вспомогательных проверках отличия от нуля знаменателей. При желании, эти проверки можно будет тут же повторить. Однако, ввиду тривиальности проверок, сообщения о них лучше вообще не выдавать. Чтобы убрать сообщения, следует вернуться в прием и добавить указатели "комментарий(1 титр(стоп))", "комментарий(2 титр(стоп))", "комментарий(3 титр(стоп))", передающие комментарии (титр стоп) проверочным операторам первых трех антецедентов.

13. Для удаления всех введенных в упраженениях приемов последовательно нажимаем Ctr-Del из просмотра их описаний. Затем убираем пустой концевой пункт "Тест", выходим в оглавление базы приемов и нажимаем "О".

13.2.4 Анализ приема

Разберем ряд простых упражнений на анализ описаний уже созданных приемов. Так как многие компоненты приема имеют эвристический характер, то чтобы понять их назначение, нужно прежде всего найти задачи, ради которых они были введены. Если возникло предположение о целесообразности изменения данных компонент, то необходимо провести тестирование измененного приема на тех же задачах, а при ослаблении ограничений на срабатывание - также и на дополнительных.

1. Найти раздел "Углы при параллельных прямых и секущей", и в этом разделе войти в просмотр описания третьего с конца приема. Объяснить фильтры этого приема. Найти в разделе задачника ("Элементарная геометрия", "Задачи на вычисление") все задачи, в которых данный прием срабатывает. Определить, для каких из них его срабатывание избыточно. Предложить дополнительный фильтр, блокирующий хотя бы часть ненужных срабатываний.

2. Найти раздел "Произведение в основании степени", и в этом разделе войти в просмотр описания последнего приема. Объяснить фильтры этого приема. Найти в разделе задачника "Исследование сходимости числовых рядов - 3" задачу, для которой удаление фильтра "не(контекст(...))" приводит к зацикливанию процесса решения.

3. Найти раздел "Потенцирование логарифмических уравнений", и в этом разделе войти в просмотр второго с конца приема. Объяснить фильтры этого приема. Найти в задачнике задачу, в которой данный прием срабатывает, и выйти на точку его срабатывания.

4. Найти раздел "Возведение в квадрат уравнения с одним неизвестным радикалом", и в нем войти в просмотр последнего приема. Объяснить фильтры этого приема. Придумать систему уравнений, в которой фильтр "или(равно(число-неизвестных(корень)1)...)" ускоряет получение ответа. Найти такую систему в задачнике.

5. Найти в ветви раздела "Плюс" концевой подраздел "Вычитание уравнения, домноженного на известный коэффициент, с целью получения линейного уравнения, либо уравнения с одной неизвестной, либо разложения на множители левой части уравнения", и войти в просмотр единственного приема этого подраздела. Объяснить фильтры приема. Придумать задачу, в которой прием сработал бы, занести ее в задачник, и выйти на точку срабатывания приема.

6. Чтобы найти прием, суммирующий квадраты синусов элементов арифметической прогрессии, занести в задачник пример на такое суммирование, выйти при трассировке на срабатывание искомого приема, и определить его место в оглавлении базы приемов.

7. Найти в ветви раздела "Описанная около фигуры окружность" концевой подраздел "Теорема синусов". Рассмотреть последние четыре приема этого подраздела, и для каждого из них найти все задачи на вычисление по планиметрии, в которых он срабатывает. Определить, в каких задачах срабатывание приема являлось избыточным. Предложить фильтры, отсекающие хотя бы часть таких срабатываний.

8. Найти в ветви раздела "Касательная" концевой подраздел "Длина касательной и длины отрезков секущей". Рассмотреть последние три приема этого подраздела, и для каждого из них найти все планиметрические вычислительные задачи, в которых он срабатывает. Объяснить различия между контекстами срабатывания этих приемов. Определить, в каких задачах срабатывание приема было избыточным.

9. Найти раздел ("Тангенс", "Уравнения", "Переход к тангенсу половинного аргумента") и войти в просмотр его последнего приема. Объяснить фильтры этого приема. Придумать задачу, в которой прием сработал бы; занести ее в задачник, и выйти на точку срабатывания.

10. В ветви раздела "Плюс" найти концевой подраздел "Выражение неизвестной X из уравнения $X + A = B$" и войти в просмотр единственного приема этого подраздела. Определить последствия удаления фильтра "или(и(тип(описать) уровень(5))...)".

Указания

1. Прием находится в разделе ("Элементарная геометрия", "Параллельны", "Углы при параллельных прямых и секущей"). Уровень срабатывания его равен 9, причем срабатывание происходит при рассмотрении некоторой посылки задачи на доказательство либо на исследование. Заметим, что почти все приемы по элементарной геометрии предназначены для срабатывания в списке посылок задач этих типов. Задачи на вычисление либо на построение, изначально оформляемые как задачи на описание, решаются (в особенности первые) с помощью вывода следствий в своем блоке анализа.

   Фильтр "неизв(...)" означает, что величина угла DFE между касательной и секущей должна быть не известной и связанной с "основными" неизвестными, т.е. неизвестными внешней задачи на описание. Такая связь понимается либо как вхождение неизвестной в выражение t для величины угла, либо как наличие цепочки уравнений, в первое из которых входит какой-либо угол или расстояние из t, в последнее - неизвестная, а любые два соседних уравнения имеют какое - либо общее расстояние либо угол. Заметим, что выражение t для величины угла DFE находится в приеме с помощью нормализатора "нормугол".

   Следующий фильтр отсекает вырожденный случай совпадения рассматриваемых параллельных прямых (хотя и не гарантирует этого несовпадения, так как совпадающие прямые могут иметь различные обозначения).

   Последний фильтр отсекает случаи, когда отрезок секущей является стороной параллелограмма, трапеции либо ромба - для углов этих фигур имеются свои приемы. Квадрат и прямоугольник не рассматриваются потому, что для них угол DFE известен.

   В общем-то, этот фильтр не доведен до конца - нужно было бы еще уточнить, что смежные стороны не лежат на наших параллельных прямых. Такого рода недоработок в текущей версии решателя имеется немало; обычно они легко устраняются, хотя иногда для этого может понадобиться дальнейшее пополнение ГЕНОЛОГа. Наличие их объясняется, во-первых, объемом системы, а во-вторых, теми целями, которые ставились при ее развитии - проверкой и уточнением общих принципов моделирования логических процессов, выявлением

общих архитектурных и технологических подходов. На этом этапе работы решатель должен рассматриваться не как готовый продукт, а лишь как первая черновая версия проработки предметных областей, которая могла бы стать организующим началом для последующего систематического их освоения. Впрочем, принцип преодоления при обучении решателя "всех задач из задачника подряд" делает свое дело - в тех случаях, где недоработки действительно ощутимы на реальном потоке задач, они достаточно быстро обнаруживаются и, так или иначе, подправляются. Это приводит к тому, что даже данная версия уже может рассматриваться как некоторая заявка на создание системы компьютерной математики нового типа, в особенности если сравнивать ее возможности по решению плохо алгоритмизируемых задач с возможностями других систем компьютерной математики.

Следующий пункт упражнения - поиск задач, на которых прием срабатывает. Чтобы можно было выполнить быстрый поиск в заданном разделе задачника всех таких задач, должна была иметь место хотя бы одна прокрутка решателя на этом разделе после того, как прием был введен. Если после прокрутки прием был изменен, то поиск даст задачи, в которых прием срабатывал до изменения. Для инициализации поиска следует перейти от просмотра приема к задачнику, нажав клавишу Shift-3. Затем следует выделить тот раздел, для которого будет выполняться поиск, нажать Shift-2 и таким образом снова оказаться в оглавлении базы теорем. Здесь нужно войти в просмотр приема и нажать "А" (кир.). Если нужно получить данные для серии приемов, то после нажатия "А" можно перейти к другим приемам, нажимая при просмотре каждого из них на "А". После нажатия на "А" происходит просмотр протоколов задачника, который может создавать небольшую паузу (обычно только для первого нажатия "А"). По завершении указанных действий нажимается клавиша Shift-3, снова переводящая в оглавление задачника. Данные просмотра сохраняются в буфере задачника, и для перехода к нему нажимается "ф".

В нашем примере, после выполнения перечисленных действий, находим в буфере раздел "Углы при параллельных прямых и секущей" - в нем и будут собраны все найденные задачи. Нажимаем "курсор вправо" и обнаруживаем номера 8 отобранных задач. В действительности это лишь повторные ссылки на задачи, уже имеющиеся в основных разделах задачника. Поэтому изменять задачи в буфере не следует. Однако, можно запускать процессы решения их так же, как для основной части задачника - поштучно либо сразу серией из 8 задач.

Нам нужно определить, в каких из 8 задач рассматриваемый прием в действительности не нужен. Чтобы сделать это, можно было бы зайти в оглавление базы приемов, отключить прием нажатием клавиши F7, запустить процессы решения отобранных задач, и посмотреть, каковы будут результаты. Для начала рекомендуем реализовать этот способ. Здесь надо иметь в виду, что при решении задачи с отключенным приемом решение ее может замедлиться настолько, что для перехода к следующей задаче серии нужно будет нажать последовательно Break и Esc (впрочем, в нашем примере этого делать не нужно). Напоминаем, что серийный запуск выполняется, после возвращения в буфер, нажатием Ctr-з.

По окончании серийного решения просматриваем его итоги, нажав клавишу "з". Видно, что одна задача вообще не решена, а решение трех - очень сильно замедлилось. Для выхода на задачу с утерянным решением нажимаем "у", для

просмотра серии задач с замедлением решения (упорядоченных по убыванию замедления) - нажимаем "з", и для перехода к каждой очередной задаче - "ш". Любую из этих задач можно выбрать для пошагового просмотра решения и анализа возможностей его доработки. Впрочем, нам известна причина перечисленных ухудшений; интерес представляют в первую очередь другие задачи - те, где ответ не изменился, а решение ускорилось. Если анализ точек срабатывания приема в этих задачах покажет, что оно бессмысленно или крайне слабо мотивировано, то появится возможность ввести дополнительный фильтр, отсекающий данные точки. Разумеется, после этого нужно будет еще проверить, не оказались ли отсечены срабатывания приема в тех задачах, которые без него не решаются либо замедляются. Здесь могут появиться еще какие-то коррекции фильтра, и т.д. В этом итеративном процессе нужно помнить, что замедление либо потеря решения после отключения приема, вообще говоря, еще не означают, что он действительно необходим. К сожалению, достаточно часто задача решается из-за случайного "рикошета", возникшего после срабатывания приема, и тогда приходится продолжать работу с ней до выявления тех средств, которые в действительности ей нужны.

Мы не будем использовать данные серийной прокрутки, полученные указанным выше способом. Вместо этого восстановим программу приема нажатием клавиши F5 (из просмотра приема) и повторим прокрутку отобранных 8 задач при подключенном приеме - для восстановления исходной ситуации. Дело в том, что анализ избыточности приема на отобранных задачах можно выполнить автоматически. Для этого достаточно нажать клавишу "й", инициирующую серийный запуск с блокировкой срабатываний нашего приема без удаления его программы. Если время решения задачи с заблокированным приемом оказывается значительно больше исходного, то процесс ее решения обрывается автоматически, и начинается рассмотрение следующей задачи. За ходом процесса можно следить по тому, как выделенный синим цветом номер текущей задачи перемещается вниз. В тех случаях, когда ответ на задачу сохранился, а время решения сократилось, после номера задачи ставится знак вопроса - здесь применение приема не нужно.

После прокрутки возникают знаки вопроса на третьей, четвертой и шестой задачах. Чтобы теперь анализировать контексты срабатывания приема, лучше всего дополнить его указателем "стоп" и перекомпилировать. Тогда при каждом его применении, непосредственно до выполнения преобразования, будет происходить выход в отладчик ЛОСа. Чтобы просмотреть преобразование, достаточно будет в этой ситуации нажать "пробел" для перехода в трассировку по шагам решения задачи и Enter - для начала такой трассировки. На первом же шаге будет выведена информация о текущем применении приема. В дальнейшем, по завершении работы с приемом, указатель "стоп" следует удалить - вручную либо, если не было внесено других изменений в прием, нажатием "Б" для восстановления исходной версии.

Чтобы понять, какие фильтры можно было бы ввести для отсечения найденных трех ненужных срабатываний, сначала стоит посмотреть на те задачи, в которых преобразование оказалось нужным. Возможно, при этом выяснятся какие-то дополнительные связи величин, входящих в выводимое приемом соотношение, с прочими объектами задачи, при наличии которых увеличивается вероятность извлечь пользу из соотношения. Просматривая контексты срабатывания

приема в пяти задачах, не помеченных знаком вопроса, обнаружим, например, что в каждой из них вершина E нового угла BEF, вводимого приемом, уже являлась вершиной некоторого рассматриваемого в задаче угла. Просматривая после этого контексты срабатывания в остальных трех задачах, обнаруживаем, что для них это условие выполнялось не всегда. Это уже дает основание для ввода нового фильтра "контекст(усм(актив(угол(x2 x30 x3))))", означающего наличие выделенного в задаче угла с вершиной E; здесь вместо переменной E формульного редактора используем ее обозначение x30 текстовым редактором. Вводим данный фильтр в третье окно описания приема, удаляем указатель "стоп" и перекомпилируем прием нажатием F3. После тестовой прокрутки восьми задач из буфера обнаруживаем, что решение четвертой задачи ощутимо ускорилось - вместо 53 млн.шагов интерпретатора оно стало требовать лишь 40 млн. шагов.

Разумеется, проведенный здесь анализ весьма поверхностный, и в данном примере можно было бы попытаться продолжить оптимизацию решения выделенных задач, добиваясь более веской мотивировки применения приема. В тех случаях, когда эта мотивировка неясна, а без приема задача решается хуже, нужно искать другие хорошо мотивированные действия, идущие в обход применения данного приема, и создавать для них новые приемы. Материала для подобной оптимизации в решателе более чем достаточно. Продолжение ее нужно не только ради получения эффективной системы для пользователя, но главным образом для создания средств автоматического синтеза приемов. Многие компоненты описаний приемов вынужденным образом имеют эвристический характер, однако даже в них прослеживается определенная логика. Расшифровка такой логики неразрывно связана с оптимизацией решателя, отсечением "предрассудков" и выявлением тех взаимодействий между приемами, которые реально нужно учитывать при принятии решений. Можно надеяться, что продвижение в этом направлении позволит постепенно перейти к автоматическому развитию решателей.

2. Путь к разделу в оглавлении базы приемов - ("Элементарная алгебра", "Степени", "Общая стандартизация выражений", "Простейшие свойства степени", "Произведение в основании степени"). Прием предназначен для преобразования степени произведения двух неотрицательных сомножителей в произведение их степеней. Он срабатывает сразу же, как только предоставляется такая возможность - на уровне 0. Переход к степеням с более простыми основаниями рассматривается в решателе как направление общей стандартизации выражений, принимаемое на промежуточных этапах решения задачи. Как правило, перед выдачей ответа выполняется обратный переход - группировка под общий показатель степени. Однако, в ряде случаев разгруппировка степени произведения в произведение степеней сомножителей блокируется, и для ознакомления с тем, когда это признано целесообразным, рассматриваем фильтры приема.

   Прежде всего, идет фильтр, запрещающий срабатывание приема в условии задачи на преобразование, имеющей комментарий "длина". Такой комментарий означает, что в задаче уже предпринималась итоговая компактная переформулировка результата; в частности, могла быть предпринята группировка сомножителей под общую степень. После этого срабатывание данного приема, конечно, нежелательно. Аналогичным образом объясняется следующий фильтр: срабатывание приема блокируется при завершающем редактировании ответа

задачи на описание. Исключение могут составлять лишь случаи, в которых цель "редакция" появляется не на завершающем этапе поиска неизвестных, а для специального преобразования списка утверждений. В фильтре и указан один из таких случаев - наличие цели "редуцирование", используемой при логическом выводе в базе теорем.

Далее идет ускоряющий фильтр: если число сомножителей равно двум, то в силу симметрии можно потребовать, чтобы тот из них, который идентифицируется с a, лексикографически предшествовал идентифицированному с b. Таким образом вместо двух случаев будет рассматриваться только один. Заметим, что если число сомножителей более двух, то один из множителей (какой именно - решает компилятор) будет идентифицроваться как одиночный операнд произведения, а другой - как произведение всех оставшихся операндов.

Далее - фильтр, запрещающий разгруппировку основания степени, если показатель c ее связан внешним описателем "класс" либо "отображение", а основание не имеет связанных этим описателем переменных. В таких случаях выгоднее упрощать выражение в первую очередь относительно переменных связывающей приставки описателя, так как это может позволить устранить описатель. Например, применение разгруппировки степени при рассмотрении суммы геометрической прогрессии нецелесообразно - выражение потеряет "явный" вид геометрической прогрессии, и прием ее суммирования может не сработать.

Следующий фильтр аналогичен только что рассмотренному - он блокирует разгруппировку степени при рассмотрении показательных уравнений, если основание не содержит неизвестных, а показатель содержит. Общая эвристическая мотивировка похожая - в уравнениях выгодно, при отсутствии особых соображений, упрощать выражения по числу вхождений неизвестных. Для рассматриваемого приема такие конкурирующие соображения в задачах не возникали, что и нашло свое отражение в данном фильтре.

Наконец, последний фильтр нужен для указания на разделение труда между различными приемами. Данный прием срабатывает, если усматривается неотрицательность обоих сомножителей основания степени. Однако, если бы показатель степени имел вид числовой дроби с нечетным знаменателем, то проверки неотрицательности были бы излишни, и для этого случая предусмотрен другой прием (в данном разделе он идет вторым от конца). Так как уровни срабатывания обоих приемов одинаковы (равны 0), то приходится вставлять фильтр, блокирующий применение данного приема в ситуациях, подпадающих под область действия другого. Разумеется, этот фильтр имеет лишь ускоряющий характер - при его отсутствии задачи все равно решались бы, но с несколько большей трудоемкостью.

Чтобы протестировать фильтр "не(контекст(...))", нужно найти такие задачи, в которых степенное выражение входит в определение функции, причем показатель степени содержит варьируемую переменную, а основание - не содержит. Удалим этот фильтр из описания приема и перекомпилируем прием; затем (как сказано в задании) войдем в раздел задачника ("Математический анализ", "Ряды", "Исследование сходимости числовых рядов - 3"). Для серийного запуска решателя в этом разделе нажимаем Ctr-з. Через некоторое время одна из задач "залипает" на экране. Нажимаем Break и выходим в просмотр отладчиком ЛОСа какого-то момента выполнения программы. Для выхода на уровень трас-

сировки "по шагам решения" нажимаем клавиши "пробел" и Enter. Используя для смены кадров нажатия Enter, обнаруживаем зацикливание: сначала срабатывает анализируемый прием, заменяющий $(2(\sin x)^2)^i$ на $2^i(\sin x)^{2i}$), затем - прием, устраняющий возникающие после такой замены вложенные умножения, и далее - прием, группирующий под общий показатель степени i два только что возникших множителя. Последний прием при суммировании действительно бывает нужен - например, для выявления геометрической прогрессии либо каких-либо других стандартных сумм с "показательными" подвыражениями, а значит, рассматриваемый фильтр в этих ситуациях тоже придется вводить - без него решатель будет зацикливаться.

Для обрыва серийного решения возвращаемся в отладчик ЛОСа (нажатие "ф") и нажимаем Ctr-з, Esc. Затем возвращаемся в просмотр нашего приема и нажимаем "Б" для восстановления его исходного вида.

3. Прием находим в разделе ("Элементарная алгебра", "Логарифмы", "Решение уравнений", "Потенцирование логарифмических уравнений"). Прием имеет в качестве возможных уровней срабатывания значения от 1 до 5. Он применяется для преобразования условия задачи на описание либо посылки задачи на исследование. Вообще, посылки задачи на исследование, содержащие неизвестные, часто преобразуются так же, как условия задачи на описание, хотя полного совпадения здесь нет. Теорема приема выделяет логарифмический множитель слагаемого левой части уравнения, причем этот множитель должен содержать неизвестную - либо в основании логарифма, либо в выражении под логарифмом. Если решается система уравнений (число неизвестных более одной), то уровень срабатывания 2 отбрасывается, иначе - отбрасывается уровень 3. С учетом ограничений на уровень срабатывания, определяемых дальнейшими фильтрами, это означает, что для систем уравнений прием не спешит с потенцированием - здесь имеются возможности решить задачу и другими средствами, в то время как для одного уравнения его необходимость более вероятна.

Далее идет фильтр, учитывающий наличие комментария "логарифм". Такой комментарий к задаче вводится приемом, выполняющим логарифмирование показательного уравнения. Чтобы понять это, можно было бы, например, воспользоваться оператором "текприем", занеся в него условие проверки наличия указателя "замечание(логарифм)" и просмотрев все приемы, имеющие данный указатель. Ясно, что если уже был предпринят шаг перехода от показательных уравнений к логарифмическим, то обратный переход нужно как-то заблокировать. Но даже и здесь, когда потенцирование очевидным образом приводит к упрощению ситуации, его все же следует разрешать. Фильтр допускает потенцирование уравнения при наличии комментария "логарифм" лишь в тех случаях, когда левая часть уравнения представляет собой линейную комбинацию логарифмов, у которой все основания и коэффициенты известны - тогда после потенцирования неизвестные не будут входить в показатели степени.

Далее располагается фильтр, разрешающий срабатывание данного приема в двух случаях:

а) Под логарифмом находится степенное выражение с неизвестным показателем. Тогда исключение логарифма может привести к обычному показательному уравнению. Уровень срабатывания 1 выбран здесь потому, что маловероятно

решение такого "двухэтажного" по логарифмическим и показательным конструкциям уравнения другими средствами.

б) Каждое неизвестное слагаемое остатка d левой части уравнения, имеющее общую неизвестную с выделенным логарифмом, является произведением логарифма по основанию a на известный коэффициент. При этом уровень срабатывания 1 разрешается лишь для системы уравнений, в которой другие уравнения не имеют ни логарифма выражения b, ни степени с основанием b. Если бы такое уравнение нашлось, то имелась бы некоторая вероятность решить систему без потенцирования - рассмотрением линейных комбинаций уравнений и логарифмированием других уравнений.

Далее идет фильтр, запрещающий наличие неизвестных логарифмов внутри выражения c - здесь либо возможно упрощение логарифмических выражений до потенцирования, либо потенцирование приводит к столь же сложной для решения уравнения ситуации, что и исходная.

Два последних фильтра связаны с задачами, возникающими в математическом анализе и дифференциальных уравнениях; их пока рассматривать не будем. Чтобы распознавать такие фильтры, заметим только, что обычно в них встречаются символы "связка", "нормИнтеграл", "диффильтр".

Чтобы найти задачу, в которой данный прием срабатывает, выходим в оглавление задачника, выделяем раздел "Элементарная алгебра", нажимаем Shift-2 для перехода в оглавление приемов, входим в просмотр приема, нажимаем из этого просмотра "A" (кир.), и далее нажимаем Shift-3 (после небольшой паузы, в течение которой происходит просмотр архива задачника). Тогда снова возникает оглавление задачника, в котором нажимаем "ф" для перехода к буферу. В буфере обнаруживаем список всех задач раздела "Элементарная алгебра", в которых срабатывает рассматриваемый прием. Этот список находится в разделе, заголовок которого воспроизводит заголовок раздела приема в базе приемов. Выбираем, например, первую задачу списка, нажимаем "г" для запуска ее решения через оглавление приемов, входим в просмотр приема и нажимаем Enter. Первое срабатывание приема отображается на экране, и далее устанавливается режим трассировки по шагам решения задачи. По завершении упражнения расчищаем буфер задачника - в произвольном месте его оглавления нажимаем "О" (кир.).

4. Искомый прием находится в разделе ("Элементарная алгебра", "Степени", "Решение уравнений", "Уравнение с радикалами", "Возведение в квадрат уравнения с одним неизвестным радикалом").

   Он используется для возведения в квадрат уравнения, имеющего в левой части ровно одно слагаемое, среди множителей которого есть неизвестные радикалы четной степени. Известно, что возведение в квадрат позволяет уменьшать число слагаемых с неизвестными радикалами лишь для малого числа таковых. Если их нельзя перегруппировать так, что в каждой части будет не более двух радикалов, то возведение в квадрат нецелесообразно. Поэтому в решателе отдельно рассмотрены все 4 случая для возможного числа слагаемых с радикалами - от 1 до 4. В случае нескольких радикалов в приемы вводится учет особых обстоятельств, позволяющих рационально сгруппировать радикалы перед возведением в квадрат. Уровень срабатывания приемов достаточно высок - начиная

с 5. Это сделано для того, чтобы перед возведением в квадрат был, хотя бы бегло, рассмотрен блок анализа задачи и, по мере возможности, использованы более простые средства решения. Например, в случае системы уравнений более эффективным может оказаться использование для исключения радикала линейных комбинаций уравнений; в случае одной неизвестной уравнение может быть решено как квадратное относительно радикала, и т.п.

Первые фильтры рассматриваемого приема фиксируют общий контекст срабатывания: уровень 5; уравнение является условием задачи на описание; выражение под радикалом содержит неизвестные; правая часть уравнения известна. Далее идет фильтр, требующий, чтобы среди остальных слагаемых левой части не имелось ни одного, чьим множителем был бы радикал четной степени. Фактически в фильтре сформулировано немного более общее условие - отсутствие показателя степени, коэффициент знаменателя которого делился бы на 2.

Далее расположено несколько странное, на первый взгляд, условие - если уравнение имеет более одной неизвестной, а число слагаемых его левой части более одного, то не должно иметься другого уравнения, в левой части которого находится не менее двух слагаемых с неизвестными радикалами. Оно возникло при рассмотрении задачи, где после возведения в квадрат второго уравнения появился такой же радикал, как и в первом, и далее для исключения неизвестного радикала удалось использовать линейную комбинацию уравнений. Эта ситуация подсказала, что разумнее начинать возведение в квадрат с уравнений, имеющих более одного радикала. Разумеется, такой фильтр можно было бы сделать более точно учитывающим возможность получения в разных уравнениях "подобных" членов с неизвестными радикалами после возведения в квадрат, но и в приведенной выше упрощенной версии он оказался пока вполне достаточным.

Следующий фильтр запрещает наличие в оставшихся слагаемых левой части неизвестных логарифмов при отсутствии неизвестного логарифма под радикалом. Он возник из рассмотрения уравнений, которые вообще элементарными средствами не решаются. Такие уравнения часто появляются, например, при отыскании корней производной. Рассматриваемый фильтр ускоряет выдачу отказа в этих ситуациях, позволяя решателю переключиться во внешней задаче на другие пути решения.

Наконец, последние фильтры, судя по встречающемуся в них символу "связка", нужны при рассмотрении дифференциальных уравнений, и мы их пропускаем.

Подбираем уравнение для проверки полезности указанного в упражнении фильтра согласно приведенным выше объяснениям. Например, можно расмотреть систему $\sqrt{2x+3}\sqrt{2y+1}+y=45/8; \sqrt{2y+1}+\sqrt{2x+3}-x=5$. Отключая рассматриваемый фильтр, убеждаемся, что система не решается, в то время как при использовании фильтра находим корни $x=-3/2, y=5+5/8$.

5. Раздел приема находим, следуя вдоль пути ("Элементарная алгебра", "Плюс", "Решение уравнений", "Системы уравнений"). Прием срабатывает на втором уровне и предназначен для преобразования уравнения - условия задачи на описание путем перехода к линейной комбинации его с другим уравнением. Применяется в ситуациях, когда задача имеет более одной неизвестной. Фильтр "не(цель(редакция))" блокирует попытки применения его на том этапе, когда происходит завершающее редактирование найденного ответа.

Выражение a идентифицируется как произведение всех неизвестных множителей некоторого слагаемого второго уравнения, причем оно же, домноженное на некоторый известный коэффициент h, встречается и среди слагаемых первого (преобразуемого) уравнения. Левые части уравнений не должны иметь дробных слагаемых - так как для устранения таких слагаемых имеются другие приемы.

Фильтр "или(числонеизвестных(...)...)" требует, чтобы результирующее уравнение либо имело единственную неизвестную, либо, после разложения на множители его левой части f, распалось на несколько уравнений, соединенных связкой "или", либо было линейным по всем своим неизвестным, либо его левая часть имела меньшее число слагаемых с неизвестными дробными степенями, чем преобразуемое уравнение. Любая из этих ситуаций является некоторым эвристическим "достижением", ради которого и выполняется преобразование.

Фильтр "не(входит(независит цели))" относится к ситуациям, в которых значения неизвестных не должны зависеть от явно указанных "варьируемых" параметров задачи. Обычно в этих ситуациях значения неизвестных подбираются так, чтобы при подстановке их пропадала зависимость условий от варьируемых параметров (например, содержащее эти параметры выражение было бы домножено на 0). Поэтому надобность в данном приеме, ориентированном на обычные системы уравнений, отпадает.

Следующий фильтр - "или(равно(число(неизвестная(...)...)...)...)" - ускоряющий. Его проверка предшествует вычислению левой части f нового уравнения, и здесь происходит предварительная оценка преобразования. Он анализирует выражения b, d, которые составлены из слагаемых левых частей уравнений, остающихся в линейной комбинации. Будем называть их "остальными слагаемыми". Фильтр разрешает продолжение действий в следующих случаях: общее число неизвестных в остальных слагаемых равно 1 (тогда и f будет иметь единственную неизвестную); остальные слагаемые линейны относительно входящих в них неизвестных (тогда и f будет линейно); остальные слагаемые содержат члены, пропорциональные уничтожаемым в линейной комбинации выделенным слагаемым (тогда эти члены тоже пропадут в линейной комбинации); выражение a имеет своим множителем неизвестную дробную степень, а слагаемые выражения b - не имеют таких множителей (тогда в линейной комбинации станет одним слагаемым с дробными степенями меньше).

Далее отсекается случай, когда до и после преобразования уравнение имеет ровно одну неизвестную. Без этого легко можно было бы привести пример с зацикливанием на данном приеме.

Фильтр "не(Входит(известно цели))" - ускоряющий; он отсекает преобразования для задач, ответ которых должен быть выражен через специально заданное множество параметров. В этом случае решение происходит только за счет вывода следствий в блоке анализа.

Заметим, что все перечисленные выше фильтры приема возникали в процессе проработки потока задач и аккумулировали в себе лишь те особые случаи, которые этим потоком были подсказаны. Никакого общего критического анализа их вне данного процесса не предпринималось - это лишь сырой материал. По-видимому, наиболее естественными рамками, в которых такой анализ стоило бы предпринять, является разработка процедур автоматического синтеза приемов. Этому будет посвящена третья книга данной монографии. Однако, даже и

с такими фильтрами, решатель, в целом, оказывается достаточно работоспособен.

В качестве примера системы уравнений, на которой прием должен сработать, рассмотрим какую-либо систему с двумя неизвестными, дающую линейную комбинацию с единственной неизвестной. Например, пусть это будут уравнения $2x^2 + 3y^2 = 8, 3x^2 + 4y = 1$. Заметим, что если бы члены с y оба были линейными, то сработал бы другой прием для получения линейной комбинации, имеющий меньший уровень срабатывания.

6. Введем в задачник задачу на упрощение в о.д.з. выражения

$$\sum_{i=1}^{n} \sin(3n+1)^2.$$

Напомним, что степень тригонометрической операции ставится формульным редактором не сразу после ее обозначения, а лишь после обозначения ее аргумента (чтобы получить степень аргумента, нужно взять ее в скобки). Далее нажатием "р" (кир.) запускаем трассировку по шагам решения. Первым же шагом выполняется суммирование, о чем и сообщает поясняющий текст. Нажимаем "б" и входим в просмотр приема, выполнившего преобразование. Затем последовательно нажимаем End (возвращение в просмотр текущего преобразования), Esc (возвращение в исходный текст задачи), End (возвращение в главное меню) и "г" (вход в оглавление базы приемов). В результате окажемся на концевом разделе, содержащем сработавший прием суммирования. Возвращаясь от него "обратным ходом" к корню оглавления приемов, определяем цепочку надразделов приема: ("Элементарная алгебра", "Комбинаторные функции", "Суммавсех", "Тригонометрические суммы", "Суммирование квадратов синусов либо косинусов").

7. Прием находится в разделе ("Элементарная геометрия", "Фигуры", "Окружность", "Описанная около фигуры окружность", "Треугольник", "Теорема синусов"). Переходим в оглавление задачника, входим в раздел "Элементарная геометрия", нажимаем Shift-2 и, переходя в просмотр последних четырех приемов на теорему синусов, каждый раз нажимаем "А" (кир.). Затем возвращаемся в задачник, нажимая Shift-3. Переходим в буфер задачника и предпринимаем анализ избыточности срабатываний приемов - последовательно для каждого подраздела буфера, используя для запуска автоматического анализа клавишу "й". Для первого приема имеем серию из 49 задач, причем в 14 из них его срабатывание оказывается избыточным. Второй прием сработал в единственной задаче, да и там его срабатывание оказалось избыточно. Видимо, на момент создания приема он применялся в какой-то другой задаче, решение которой впоследствии пошло по другому руслу. Третий прием сработал тоже на единственной задаче, но здесь уже он оказался нужен. Наконец, четвертый прием срабатывает в 12 задачах, и в 4 из них - избыточен. Попытки оптимизации этих приемов предоставляем читателю.

8. Приемы расположены в разделе ("Элементарная геометрия", "Фигуры", "Касательная", "Касательная и секущая", "Длина касательной и длины отрезков секущей"). Последние три приема отличаются друг от друга только уровнями

срабатывания и фильтрами. Каждый из них выписывает соотношение равенства квадрата длины касательной произведению длин отрезков секущей.

Первый прием срабатывает при малых значениях текущего уровня - 3 либо 7. Условия срабатывания одинаковы для каждого из этих уровней; если на третьем уровне прием не сработает, то по достижении седьмого уровня могут появиться новые посылки, при которых его срабатывание уже станет возможным. Если бы не был предусмотрен дополнительный уровень срабатывания, то нужно было бы специально заботиться о переключении внимания - уменьшении до 3 веса посылки "касательная(...)", на которой инициализируется применение приема.

Ограничения на срабатывание необременительны - они сводятся к тому, чтобы некоторое количество расстояний, относящихся к рассматриваемым объектам, уже были явно указаны в задаче. Один из фильтров требует, чтобы в задаче уже встречались либо длина касательной, либо обе длины отрезков секущей, либо чтобы секущая проходила через центр окружности. Другой фильтр требует, чтобы уже была рассмотрена длина хотя бы одного из трех отрезков, выделяемых на секущей. Последние два фильтра усиливают эти ограничения, если результирующее соотношение имеет более одного расстояния, не выраженного через числовые параметры.

Второй прием срабатывает на уровне 9. Для его срабатывания нужно, чтобы хотя бы два из трех выделяемых на секущей отрезков были либо вычислены, либо достаточно тесно связаны с неизвестными (в стандартном для планиметрии смысле, определяемом фильтром "неизв(...)"). Кроме того, должно быть выделено расстояние от какого-либо конца рассматриваемого отрезка касательной до другой точки на этой же касательной.

Уровень срабатывания третьего приема равен 13. Для его срабатывания нужно, чтобы длина отрезка касательной была сязана с неизвестными, а число проходящих через рассматриваемую вне окружности точку секущих было не более 2.

В том же разделе есть еще четыре приема, выписывающих то же самое соотношение. Каждый раз, когда при решении задачи в нем возникала потребность, причем имелась какая-то специфическая особенность, отсутствовавшая в предыдущих ситуациях, - вводился дополнительный прием. Взятые вместе, все эти приемы составляют некоторую эвристическую аппроксимацию множества случаев, в которых данное соотношение оказывается полезным, разумеется, пока чрезвычайно сырую. Фильтры геометрических приемов, как правило, прослеживают возможность срабатывания каких-то других приемов, следующих за срабатыванием данного. Чтобы они не отсекали нужных срабатываний, такой учет последействий должен быть достаточно полным, а чтобы не происходило бесполезных срабатываний, он должен быть продолжаем на возможно большую глубину. Пока учет последействий в приемах достаточно эпизодический, и ценность его состоит главным образом в том, что он дает большое количество примеров для дальнейшего анализа.

Для определения задач, в которых срабатывание приема избыточно, поступаем так же, как в предыдущем упражнении. Первый прием срабатывает в 22 задачах, причем в трех из них - избыточным образом; второй прием срабатывает в

трех задачах, и во всех из них нужен; третий прием срабатывает в единственной задаче и для нее необходим.

9. Прием находится в разделе ("Элементарная алгебра", "Тригонометрия", "Тангенс", "Уравнения", "Переход к тангенсу половинного аргумента"). Прием срабатывает на шестом уровне, то есть после того, как все более простые попытки решения уравнения оказались неудачными. Некоторых пояснений требует вид теоремы приема. Левая часть эквивалентности - преобразуемое уравнение $a = b$; указатель "контекст(позиция(x7 x1)вид(x7 косинус(x4)))" определяет идентификацию неизвестного тригонометрического аргумента d по косинусу этого аргумента, встречающемуся в уравнении. Нормализатор "половинныйугол" выражает встречающиеся в левой части уравнения синусы, косинусы и тангенсы через тангенс половинного аргумента (тангенс выражается через половинный угол лишь тогда, когда этот половинный угол уже возник в выражении). Фильтры приемы требуют, чтобы кроме косинуса d уравнение имело либо синус d, либо тангенс половинного аргумента. Кроме того, проверяется, что каждая неизвестная тригонометрическая операция в уравнении совпадает либо с синусом d, либо с косинусом d, либо с тангенсом d, либо с тангенсом половинного аргумента e. Чтобы отсечь случаи чрезмерного возрастания сложности уравнения при переходе к половинному аргументу, не разрешаются степени неизвестных тригонометрических операций, отличные от второй и третьей. Наконец, прием блокируется при рассмотрении дифференциальных уравнений.

   В качестве задачи для тестирования приема будем искать уравнение вида $\sin x + \cos x + \operatorname{tg} x = a$. Величину a подберем так, чтобы уравнение решалось: проследим действия решателя до появления уравнения $(a+1)y^4 - 2y^2 + 4y = a - 1$ относительно вспомогательной неизвестной y, после чего положим $y = 1/2$ и определим $a = 41/15$. При таком значении a решатель находит все корни. Уравнение имеет две их серии, и для одной из них применяется формула Кардано.

10. Прием находится в разделе ("Элементарная алгебра", "Плюс", "Решение уравнений", "Системы уравнений", "Выражение неизвестной X из уравнения $X + A = B$"). Он преобразует уравнение $b + c = a$, где b - неизвестная; c - выражение с неизвестными, не содержащее b; a не содержит неизвестных, к виду $b = a - c$. Такое преобразование немедленно вызывает подстановку в остальные условия задачи выражения $a - c$ вместо b, т.е. фактически равносильно исключению этой неизвестной.

    Указано три различных уровня срабатывания - 0,1 и 5. Фактически уровни 0,1 - альтернативные: в случаи задачи на описание допустим только уровень 1, в случае задачи на исследование - уровень 0. Такое разделение уровней по типу задачи, видимо, объясняется тем, что в какой-то задаче на исследование (при анализе объединенного списка посылок и условий внешней задачи на описание) понадобилось ускорить выражение неизвестной из данного уравнения, чтобы не возникло состязание с другим приемом, срабатывающим на первом уровне. Подробности можно уточнить, скорректировав данный фильтр так, чтобы и в случае задач на исследование прием срабатывал на уровне 1, и выполнив прокрутку по задачнику для поиска тех задач, которые отреагируют на данное изменение негативным образом.

Срабатывание приема на малых уровнях допускается в двух случаях: либо $a = 0$, а число неизвестных равно 2, либо уравнение линейно относительно всех входящих в него неизвестных. В прочих случаях (и только для задачи на описание) уровень срабатывания равен 5. Кроме того, для срабатывания приема необходимо выполнение одного из следующих условий: уравнение имеет ровно две неизвестные; задача - на описание и имеет не более одного нелинейного уравнения; задача - на исследование, $a = 0$, а уравнение содержит ровно 3 неизвестных.

Если неизвестная b - целочисленная, а число неизвестных в c более двух, то для задачи на описание прием блокируется: целочисленные уравнения обычно решаются другими средствами. Наконец, последний фильтр учитывает ограничения на зависимость ответа от заданных переменных - в этой ситуации также используются другие средства решения.

Чтобы определить последствия удаления указанного в упражнении фильтра, проведем прокрутку по разделам задачника ("Элементарная алгебра", "Решение уравнений", "Системы алгебраических уравнений 1,2,3,4") с предварительным удалением фильтра. Так как здесь область применения приема не сужена, а расширена, определение списка задач, в которых прием срабатывал, не поможет. Придется прибегнуть к более трудоемкой полной прокрутке по избранным разделам. Обычная практика при развитии решателя - периодические (раз в несколько дней) полные прокрутки, занимающие около 2 часов. Есть не очень сильно развитые пока средства сужения класса задач, для которых предпринимается прокрутка. Они основаны на учете всех логических символов, встречавшихся при решении задачи; если для срабатывания приема необходимо наличие какого-то символа, а при решении он не возникал, то задачу можно не рассматривать. Однако, для простейших приемов, типа рассматриваемого, такое отсечение малоэффективно.

При удалении фильтра прокрутка первых двух разделов для систем уравнений, казалось бы, показывает его избыточность: несколько задач даже "ускоряются". Однако, на 44-й задаче третьего раздела неожиданно появляется сообщение "переполнение буфера текстов". Нажимая "ф", входим в отладчик ЛОСа, поднимаемся до уровня кадра программы "исследовать" и просматриваем значение переменной x1 - текущей задачи на исследование (удобно сделать это нажатием K1). Видно, что в посылках задачи присутствует введенное данным приемом явное выражение неизвестной z через x, y, а возникшие после исключения z уравнения чрезмерно громоздки. Переполнение (получение слишком длинного набора) произошло при регистрации в задаче на исследование результатов вывода анализатором "сложениеуравнений" линейных комбинаций имеющихся уравнений. Такой результат неудивителен: при решении систем нелинейных уравнений выражение одной неизвестной через другие нужно применять очень осторожно. На ограничение этих действий и был направлен устраненный фильтр.

13.2.5 Предварительные упражнения на создание нового приема

Приведем несколько несложных упражнений на самостоятельную разработку приемов. Иногда эти упражнения будут дублировать уже существующие в системе прие-

мы. Рекомендуется сначала создать свою версию, не прибегая к подсказкам решателя, и протестировать ее на какой-либо специально введенной для этого задаче. Если при тестировании выяснится, что старые приемы решателя опережают новый прием и не дают ему сработать, то следует либо взять уровень срабатывания этого приема меньшим, чем у старых приемов, либо временно отключить старые приемы. Разумеется, приводимых здесь примеров недостаточно, чтобы научиться программировать на ГЕНОЛОГе. Технике такого программирования посвящен весь второй том данной монографии, в котором подробным образом будут рассматриваться разделы базы приемов решателя.

1. Создать прием, упрощающий тангенс утроенного арктангенса.

2. Создать прием для упрощения радикала второй степени, под которым расположен полный квадрат суммы с другим радикалом. То же самое - для радикалов третьей степени.

3. Создать прием, решающий простейшее уравнение для секанса: $\sec x = a$.

4. Создать прием, находящий пересечение двух арифметических прогрессий с общим начальным членом, положительные разности которых относятся как целые числа.

5. Создать прием для суммирования кубов синусов элементов арифметической прогрессии.

6. Создать прием, связывающий площадь вписанного в окружность четырехугольника с длинами его сторон и полупериметром.

7. Создать прием, основанный на теореме о том, что точки пересечения высот, медиан и срединных перпендикуляров в треугольнике лежат на одной прямой. Придумать планиметрическую задачу на вычисление, которая могла бы быть решена решателем только после добавления данного приема.

8. Создать прием, переформулирующий через коэффициенты квадратного трехчлена условие, что заданная величина x больше его корней. Придумать задачу, в которой этот прием мог бы понадобиться.

9. Создать прием, дающий разложение тангенса в ряд с использованием чисел Бернулли.

10. Создать прием, позволяющий вычислять длину кривой, заданной уравнением $y = y(x)$ в прямоугольной системе координат.

11. Создать прием, позволяющий переходить от прямоугольных координат множества точек в трехмерном пространстве к сферическим.

Указания

1. Разумеется, начинается синтез приема с того, что определяется его теорема. Обычно это связано с решением несложной задачи, приводящей к нужной теореме, либо с обобщением какого-то известного утверждения путем ввода в него дополнительных параметров, учитывающих типичные контексты его применения в задачах. В нашем случае нужно вывести формулу тангенса тройного

арктангенса и получить теорему $\mathrm{tg}(3\,\mathrm{arctg}\,a) = (3a - a^3)/(1 - 3a^2)$. Так как это - теорема приема, то условие на область допустимых значений (отличие знаменателя от 0) можно опустить - при правильной работе других приемов оно должно усматриваться из контекста преобразуемого выражения.

Далее выбираем раздел, в котором будет размещен прием. В нашем случае естественно пойти по цепочке подразделов ("Элементарная алгебра", "Тригонометрия", "Тангенс", "Тангенс выражения, содержащего обратные тригонометрические функции"). Вводим для приема новый концевой раздел (нажатием "к"), сопроводив его, например, текстом: "Тангенс утроенного арктангенса". Далее набираем теорему приема и вводим его заголовок - "второйтерм". Так как прием выполняет преобразование, которое, по-видимому, практически всегда будет полезным, уровень срабатывания его выбираем небольшим - например, равным 3 (все-таки, здесь лучше немного подождать - вдруг на уровнях от 0 до 2 этот тангенс тройного арктангенса будет устранен как-то иначе, например, умножен на 0). Других фильтров не вводим. Указатели приему также не требуются. Сохранив (F4) набранную заготовку приема, нажимаем "н" для получения подсказок от генератора приемов о возможных дополнительных элементах описания приема. Сначала возникает указатель "логсимвол(3 умножение)", который говорит компилятору, что тройка здесь должна идентифицироваться как непосредственный операнд произведения. Однако, в нашем случае компилятор и сам это понимает, так что указатель можно не вводить. Далее идут подсказки о вводе нормализаторов общей стандартизации - их принимаем, нажимая каждый раз клавишу Insert. По окончании появления подсказок (возникает голубая линия) прием снова нужно сохранить. Так как он, по существу, уже готов, то вместо F4 сразу нажимаем F3. Наконец, тестируем работу приема. Например, вводим выражение $\mathrm{tg}(3\,\mathrm{arctg}(1/5)$ и на первом же шаге трассировки получаем $37/55$.

Заметим, что попытки протестировать прием на каком-либо значении, для которого арктангенс известен, например, для тангенса $\pi/12$, сразу приведут к срабатыванию старых приемов, и чтобы в этом случае протестировать новый прием, их придется найти и временно отключить. Если эта работа будет проделана, то мы обнаружим, что наш прием вводит сравнительно громоздкое выражение с радикалом, которое вызывает длинную цепочку упрощающих его простых преобразований. Чтобы избежать этого, можно добавить нормализаторы "видумножение" для обработки числителя и знаменателя - после этого пример будет решаться примерно вдвое быстрее.

2. Будем рассматривать простейший случай, когда под радикалом находится выражение $a + b\sqrt{c}$, представляющее собой полный квадрат другого выражения $p + q\sqrt{c}$, причем p, q не имеют радикалов. Возводя в квадрат и рассматривая систему уравнений для неизвестных p, q при известных a, b, c, получаем: $q^2 = (a + \sqrt{a^2 - cb^2})/2c \ \vee \ q^2 = (a - \sqrt{a^2 - cb^2})/2c$; $p = b/2q$. Это подсказывает следующую схему вычислений, выполняемых приемом. Сначала предпринимается попытка разложить на множители выражение $a^2 - cb^2$ и проверить, что результат является полным квадратом некоторого выражения r. Затем рассматриваются выражения $a + r, a - r$ и предпринимается попытка представить какое-либо из них в виде произведения $2c$ на полный квадрат некоторого выражения - оно и будет равно q. Наконец, по b и q определяется p. Теорема приема, соответствующая такой последовательности действий, имеет вид:

$\forall_{abcpqr}(r^2 = a^2 - cb^2 \ \&\ (a + r = 2cq^2 \vee a - r = 2cq^2) \ \&\ b = 2pq \to \sqrt{a + b\sqrt{c}}) = |p + q\sqrt{c}|)$

После набора этой теоремы вводим остальные компоненты описания приема: заголовок - "второйтерм"; фильтр для уровня срабатывания "уровень(0)" и проверяющий отсутствие радикалов в коэффициентах a, b фильтр "не(контекст(список(x5 a b)позиция(x6 x5)вид(x6 степень(x7 дробь(1 2)))))"; указатели "единица(1 b p q)", "заменазнака(минус b p)". Нулевой уровень срабатывания здесь введен, чтобы при тестировании не срабатывали другие приемы, уже имеющиеся в решателе. Впрочем, их уровень срабатывания также маленький - лучше сразу усмотреть возможность избавиться от двойного радикала, чем выполнять с ним какие-то действия. Выражения $a+r, a-r$ могут не иметь заголовка "умножение", а быть числовыми константами. В этом случае идентификация их с произведением $2cq^2$ будет заключаться в делении на число $2c$ и проверке того, что частное является полным квадратом. Чтобы предусмотреть эту ситуацию, добавляем указатель "пересечениесписков(фикс(2 1 2)фикс(2 2 2))", отменяющий проверку наличия у данных выражений заголовка "умножение".

Далее вводим указатель "идентификатор(1 2 3)" и расставляем указатели общей стандартизации. Наконец, добавляем указатели нормализации "видумножение" для разложения на множители выражений $a^2 - cb^2$, $a + r$, $a - r$.

Тестируем прием на какой-либо задаче, например, $\sqrt{2\sqrt{2} + 3}$.

Создание аналогичного приема для радикалов третьей степени, а также возможные обобщения приведенного выше приема оставляем читателю.

3. Теорему приема нетрудно получить, перейдя от уравнения $\sec x = a$ к уравнению $\cos x = 1/a$:

$\forall_{ax}(\sec x = a \leftrightarrow (0 \le 1 + 1/a \ \&\ 0 \le 1 - 1/a \ \&\ (\exists_n(\text{целое}(n) \ \&\ x = \arccos(1/a) + 2\pi n) \ \vee \ \exists_n(\text{целое}(n) \ \&\ x = -\arccos(1/a) + 2\pi n))))$.

Вместо приведенных здесь двух неравенств для a можно было бы использовать неравенство для модуля $1 \le |a|$. Однако, следует помнить, что эти неравенства будут обеспечивать сопровождение по области допустимых значений для арккосинуса. Так как проверки здесь будут относится к выражению $1/a$, то переход к модулю существенно их затруднит, и лучше его не вводить. С другой стороны, от знаменателей в неравенстве можно избавиться, после чего теорема приема примет вид:

$\forall_{ax}(\sec x = a \leftrightarrow (0 \le a(a + 1) \ \&\ 0 \le a(a - 1) \ \&\ (\exists_n(\text{целое}(n) \ \&\ x = \arccos(1/a) + 2\pi n) \ \vee \ \exists_n(\text{целое}(n) \ \&\ x = -\arccos(1/a) + 2\pi n))))$.

Уровень срабатывания приема выбираем равным 2 (как и для прочих тригонометрических операций). Контекст срабатывания задается фильтрами "условие", "тип(описать)", "корень", "известно(a)", "не(известно(x))". Добавляем также имеющийся у других аналогичных приемов фильтр "конец(не(фильтрсерии(a)))", который блокирует данный переход из общих соображений. Например, если уже получено параметрическое описание для какой-либо неизвестной, встречающейся в x, то вместо применения данного приема выгоднее подставить в выражение x явное значение этой неизвестной. Указатель приема вводим единственный - "примечание(серия)". Он сопровождает преобразованное условие комментарием "серия", поясняющим, что встречающиеся в нем кванторы

существования дают параметрическое описание и нет смысла пытаться устранить их, получив явное разрешение подкванторного утверждения относительно связанной переменной.

В заключение расставляем нормализаторы общей стандартизации. При этом нормализатор "нормарккосинус" сопровождаем указателем "посылки(и(целое(n)меньшеилиравно(0 $a(a+1)$)меньшеилиравно(0 $a(a-1)$)))".

4. Множество членов арифметической прогрессии будем записывать в виде $\mathrm{set}_x(\exists_i($целое$(i)$ & $0 \le i$ & $x = a + bi))$. Если есть две прогрессии, с положительными разностями b и c, отношение которых - рациональное число, то для определения такого отношения m/n будем использовать в теореме приема процедуру сокращения дробного выражения b/c. Чтобы убедиться в несократимости дроби m/n, добавляем проверку взаимной простоты чисел m и n - тогда результирующая прогрессия будет иметь разность bn. Таким образом, получаем теорему приема:

   $\forall_{abcmn}(b/c = m/n$ & натуральное(m) & натуральное(n) & взаимнопросты(m,n) $\to \mathrm{set}_x(\exists_i($целое$(i)$ & $0 \le i$ & $x = a + bi)) \cap \mathrm{set}_x(\exists_j($целое$(j)$ & $0 \le j$ & $x = a + cj)) = \mathrm{set}_x(\exists_i($целое$(i)$ & $0 \le i$ & $x = a + bni)))$.

   Единственный фильтр приема - "уровень(1)", так как преобразование представляется полезнмы в любых ситуациях. Заголовок приема - "второйтерм". Указатели - "идентификатор(1)", "блокпроверок(2 3)", "единица(0 a)", "единица(1 b c n)". Нормализаторы приема - обычные, в том числе нормализатор "нормдробь" для частного b/c. В принципе, можно было бы несколько усилить прием, введя также нормализатор "видумножение" для разложения на множители выражений b, c перед попыткой сокращения дроби.

   Проверяем созданный прием на каком-либо простом примере; оказывается, что после его срабатывания применяется другой прием, преобразующий результат к виду "арифмпрогрессия(a bn)". Если бы существование последнего приема было известно заранее, то можно было бы прием для пересечения формулировать сразу с использованием обозначения "арифмпрогрессия".

5. В решателе имеются приемы, позволяющие вычислять сумму синусов либо косинусов элементов арифметической прогрессии. Поэтому все, что требуется для вычисления суммы кубов синусов либо косинусов - это преобразование их к функциям кратного аргумента, так как дальше сработают указанные выше приемы. Это соображение позволяет нам создать даже более общий прием - для вычисления сумм проивзольных натуральных степеней синусов и косинусов. Теорема такого приема будет, по существу, вырожденной - в ней должно содержаться только обращение к нормализатору "стандплюс" - он выполняет не только раскрывание скобок, но и переход к кратному тригонометрическому аргументу, если при обращении ввести комментарий "видумножение". Записывается данное обращение так:

$$\forall_{abmnkp}((\sin(a+bi))^k = p(i) \to \sum_{i=m}^{n}(\sin(a+bi))^k = \sum_{i=m}^{n} p(i)).$$

   Здесь $p(i)$ - вспомогательное обозначение для результата перехода к кратному аргументу. Заголовок приема - "второйтерм". Уровень срабатывания его выберем равным единице. Добавим фильтры "натуральное(k)" (показатель степени есть десятичная запись натурального числа) и "меньше$(k$ 6)" (отсекаем

слишком большие показатели степени, так как пока неясно, нужно ли будет для них применять данный переход). Вводим стандартный набор указателей - "идентификатор(1)", "отображение(p)", "единица(0 x1)", "заменазнака(минус x2)", "единица(1 x2)". Левую часть антецедента теоремы снабжаем нормализаторами "стандплюс(замечание(видумножение))", "посылки(и(целое(i) меньшеилиравно(m i) меньшеилиравно(i n)))". Указатель нормализации "посылки(...)" введен из-за того, что преобразуемое выражение содержит переменную i, определенную лишь в контексте суммирования, и нормализатору может понадобиться дополнительная информация о значении этой переменной.

6. Формулу, связывающую площадь вписанного четырехугольника с длинами его сторон a, b, c, d и полупериметром p, нетрудно вывести или взять из учебника: $S = \sqrt{(p-a)(p-b)(p-c)(p-d)}$. Здесь имеются пять параметров - S, a, b, c, d. Из общих соображений, выписывать соотношение для них имеет смысл, если достаточно большая их часть известна либо представляет интерес для нахождения значений неизвестных. Желательно, чтобы хотя бы один из параметров при этом был неизвестен - иначе соотношение, в лучшем случае, может понадобиться лишь для усмотрения противоречивости описания чертежа. Вообще-то, новые приемы вводятся в решатель только в том случае, когда они понадобились для решения какой-то конкретной задачи. Здесь мы отступаем от этого правила - и это может сказаться на качестве приема. Если нет примера, подтверждающего полезность приема, то не исключено, что он вообще будет избыточен. Поэтому, доведя упражнение до конца, попробуем найти задачу, которая без данного приема решаться системой не будет.

   Прежде чем вводить теорему приема, нарисуем геометрическим редактором чертеж - проведем окружность EF и изобразим вписанный в нее четырехугольник $ABCD$. Теорема приема, кроме приведенного выше соотношения, должна содержать также описание геометрической ситуации:

   $\forall_{ABCDEFabcdpS}$(окружность(EF)описана около фигура($ABCD$) & актив(S(фигура($ABCD$))) & $p = (l(AB)+l(BC)+l(CD)+l(AD))/2 \rightarrow S$(фигура($ABCD$)) = $\sqrt{(p-l(AB))(p-l(BC))(p-l(CD))(p-l(AD))}$).

   Второй антецедент означает, что в задаче уже рассматривается площадь четырехугольника - в этом случае автоматически вводится явная ссылка на площадь, представляющая собой фиктивную посылку вида "актив(площадь(...))". Третий антецедент вводит вспомогательное обозначение p для полупериметра, который четырежды упоминается в консеквенте теоремы.

   Прием имеет заголовок "вывод". Уровень его срабатывания выберем средним - для геометрии в качестве такого уровня годится, например, шестой. Соответственно, имеем фильтры "уровень(6)", "тип(исследовать)".

   Переходим к основному фильтру, который будет рассматривать список связываемых выводимым равенством параметров. Начнем этот фильтр с перечисления значений данных параметров: "контекст(равно(x5 набор(терм(расстояние(AB)) терм(расстояние(BC)) терм(расстояние(CD)) терм(расстояние(AD)) терм(площадь(фигура(набор($ABCD$))))))... Так как мы перечисляем в наборе x5 термы, определяемые теоремными выражениями, то каждое из них заключаем внутри записи "терм(...)". В действительности, нас будут интересовать не выражения "расстояние(...)", "площадь(...)", а значения этих выражений, опре-

деляемые на текущий момент из контекста задачи. Чтобы получить такие значения, будем обрабатывать выражения нормализаторами "нормрасстояние" и "нормплощадь".

После того, как набор x5 значений параметров определился, продолжаем запись фильтра: ...входит(x6 x5) неизв(x6) меньше(число(входит(x7 x5) не(известно(x7)) не(неизв(x7)))2) меньше(1 число(входит(x7 x5) известно(x7)))". Здесь требуется, чтобы хотя бы один параметр был не известен и связан по цепочке соотношений с неизвестными внешней задачи на описание; чтобы имелось не более одного неизвестного параметра, не связанного таким образом с неизвестными внешней задачи; чтобы не менее двух параметров были известны. Разумеется, это - лишь нулевая версия возможного фильтра, и ее стоит проверить путем прокрутки по задачнику. Если окажется, что в каких-то задачах срабатывание приема является вредным, то фильтр придется усиливать. Заметим, что пока в область допустимых срабатываний попадают все ситуации прямого определения единственного неизвестного параметра через известные остальные параметры.

Единственный указатель приема - "идентификатор(3)"; нормализаторы берем все, которые подсказывает система автоматического пополнения описания приема.

Для тестирования приема вводим простейшую задачу, чертеж которой в точности совпадает с условием теоремы - четырехугольник, вписанный в окружность и имеющий известные длины своих сторон, например, 3,5,7,4. Требуется в задаче найти площадь четырехугольника. Нажимая Ctr-ч, получаем достаточно точно воспроизводящий значения длин сторон чертеж. При трассировке убеждаемся в том, что прием срабатывает. При его отключении (на момент написания данного примера) решатель задачи не решает.

7. Начинаем создание приема с построения геометрическим редактором чертежа. Рисуем треугольник ABC, проводим высоты AD и BE, медианы BF и AG, а также срединные перпендикуляры FQ и GQ, продолженные до их точки пересечения Q. Вводим также точки P, R пересечения высоты и медиан соответственно. Затем переходим к набору теоремы:

$\forall_{ABCDEFGPQR}(\triangle(ABC)$ & $D \in$ прямая(BC) & $E \in$ прямая(AC) & прямая(AD) $\perp$прямая(BC) & прямая$(BE)\perp$прямая(AC) & $G \in$ прямая(BC) & $l(BG) =$ $l(CG)$ & $F \in$ прямая(AC) & $l(AF) = l(CF)$ & прямая$(FQ)\perp$прямая(AC) & прямая$(GQ)\perp$прямая(BC) & $P \in$ прямая(AD) & $P \in$ прямая(BE) & $R \in$ прямая(AG) & $R \in$ прямая$(BF) \rightarrow Q \in$ прямая$(PR))$.

Заголовок приема - "вывод"; уровень срабатывания выбираем не очень большим, так как утверждение о принадлежности трех точек одной прямой может оказаться существенным для всего дальнейшего ходя решения. Например, можно взять его равным 4. Обычно такой произвольный выбор уровня срабатывания корректируется при регулировочной прокрутке приема на задачах. Вводим фильтр "или(тип(доказать)тип(исследовать))". Каких-то особых соображений об ограничениях на срабатывание данного приема не возникает, и других фильтров пока не вводим. Для ввода указателей и нормализаторов

используем автоматическое пополнение описания приема - нажимаем "н" и соглашаемся с появляющимися предложениями.

Для тестирования приема введем задачу, чертеж которой совпадает с чертежом теоремы; зададим какие-то конкретные длины сторон треугольника, и потребуем найти отношение длин отрезков PR и RQ. При срабатывании приема решатель сравнительно быстро находит ответ; при его отключении ответ не выдается.

8. Условие на x запишем в таком виде: $\forall_y(ay^2 + by + c = 0 \rightarrow y < x)$. Заметим, что оно вовсе не означает наличие у трехчлена корней; если корни есть, то они должны быть меньше x, а если их нет, то никаких дополнительных ограничений не требуется. В этой ситуации, как легко видеть, условие эквивалентно следующей альтернативе: либо дискриминант неотрицателен, значение трехчлена в точке x имеет тот же знак, что и коэффициент a, а сама точка x расположена правее точки $-b/2a$ экстремума трехчлена, либо дискриминант отрицателен. Это дает следующую теорему приема:

   $\forall_{abcx}(\neg(a = 0) \rightarrow (\forall_y(ay^2 + by + c = 0 \rightarrow y < x) \leftrightarrow (0 \leq b^2 - 4ac \ \& \ 0 <$
   $a(ax^2 + bx + c) \ \& \ -b/2a < x) \vee (b^2 - 4ac < 0))))$.

   Заголовок приема - "второйтерм". Уровень срабатывания его выберем не очень большим - устранение квантора может существенно упростить задачу, и откладывать его не имеет смысла. Пусть, например, этот уровень будет равен 2. Для ввода указателей и нормализаторов прибегаем к подсказкам системы автоматического доопределения приема; со всеми этими подсказками соглашаемся. Чтобы протестировать прием, вводим какую-либо несложную задачу, например, задачу с неизвестной b и условием $\forall_x(3x^2 + bx + 5 = 0 \rightarrow x < 4)$. Запускаем трассировку по шагам решения, и обнаруживаем, что наш прием не срабатывает, а вместо него применяется другой прием, предпринимающий попытку явного разрешения подкванторного утверждения относительно переменной кванторной приставки (т.е. относительно x). На этом пути решатель, разумеется, получает ответ. Однако, такое решение, все же, более трудоемко, чем предлагаемое нашим приемом. Чтобы "опередить" старый прием, понижаем уровень срабатывания нового приема до 1. Снова запускаем трассировку, и опять оказывается, что прием не сработал. При более внимательном рассмотрении ситуации обнаруживаем, что решатель, учитывая условия на область допустимых значений, ввел под квантор, кроме условия равенства трехчлена нулю, также утверждение "число(x)". Такого рода пополнение антецедентов кванторных импликаций условиями на тип значений переменных - следствие используемой в решателе общей стандартизации, изменять которую сейчас нежелательно. Поэтому приходится изменить сам прием - ввести в него под квантор общности дополнительный антецедент "число(y)". Лишь после этого трассировка покажет срабатывание приема, а при выдаче ответа будет отмечено небольшое ускорение по сравнению с запуском без нового приема.

   Этот пример демонстрирует типичную ситуацию при обучении решателя - как правило, новый прием обязательно приводит к каким-то неожиданностям, и лишь после некоторого цикла коррекций удается получить желаемый результат. К такому явлению при обучении системы придется привыкать. Не следует воспринимать его как что-то негативное, свидетельствующее либо о малом опыте учителя системы, либо о принципиально неправильном подходе к ее обучению.

Это - совершенно нормальное явление, своего рода неизбежное "трение" процесса обучения. Создать и откомпилировать прием - это лишь малая часть труда, который приходится в него вкладывать. Основная работа состоит в том, чтобы после этого преодолеть все те неожиданные барьеры, которые выявляются при тестировании. В общем-то, это неудивительно - логические процессы сложны и разнообразны, а изучение их только начинается. Для преодоления трудоемкости дрессировки решателей необходимо иметь предусмотрительность, обеспечить которую, по-видимому, сможет лишь постепенное формирование, пусть хотя бы и эмпирическим путем, и пусть хотя бы и нематематической, но все же теории логических процессов.

9. Для разложения в ряд Тейлора решатель использует нормализатор "рядтейлора". Ему передается входной комментарий (рядтейлора x b), указывающий переменную x, для которой выполняется разложение, и точку b, в которой оно выполняется. Число Бернулли с индексом i обозначается в решателе "числобернулли(i)" (см. раздел "Элементарная алгебра", "Комбинаторные функции"). Разложение в степенной ряд выражения $\mathrm{tg}\,x$ сразу обобщаем до разложения выражения вида $tg((a(x+c)^k)/d)$. Теорема приема получается следующая:

   $\forall_{abcdxk}$(натуральное(k) & $c=-b \to \mathrm{tg}((a(x+c)^k)/d) =$

   $\sum_{i=1}^{\infty}((-1)^{i-1}2^{2i}(2^{2i}-1)$числобернулли$(2i)a^{2i-1}(x+c)^{(2i-1)k})/((2i)!d^{2i-1})$.

   Здесь b - точка разложения, передаваемая приему через комментарий к нормализатору.

   Заголовок приема - "замена(второйтерм рядтейлора)". Вводим фильтры приема "не(входит(x a))", "не(входит(x c))", "не(входит(x d))", "не(входит(x k))", необходимые для того, чтобы бесконечная сумма представляла собой степенной ряд. Комментарий, определяющий x и b, извлекаем указателем "вход(рядтейлора x b)". Далее вводим указатели "блокпроверок(1)", "идентификатор(2)", "единица(0 c)", "единица(1 a d k)", "заменазнака(минус a)" и расставляем нормализаторы общей стандартизации. Заметим, что прием служит для получения бесконечной суммы; если нужно применить локальную формулу Тейлора и ограничиться заданным числом членов, то используется нормализатор "формулатейлора". Попробуйте самостоятельно ввести аналогичный прием для последнего нормализатора, отключив имеющийся в нем прием разложения тангенса через разложения синуса и косинуса. Нахождение чисел Бернулли осуществляется нормализатором общей стандартизации "нормчислобернулли".

10. Будем считать, что кривая P задается выражением "точки(A K)", определяющим множество точек плоскости, координаты которых в системе координат K образуют множество A. На тот случай, если условие принадлежности пары чисел (x,y) множеству A не имеет вида, явно разрешенного относительно y, предпримем в приеме попытку такого разрешения. Для полученной в результате зависимости $y=f(x)$ и определившегося при разрешении отрезка $[a,b]$ далее используем известную формулу длины кривой. Таким образом, имеем следующую теорему приема:

   $\forall_{abfKPAB}$(прямкоорд(K) & $P=$точки(A,K) & $((x,y)\in A) = (y=f(x)$ & $B(x))$

   & $B(x) = (a\le x$ & $x\le b$ & число$(x)) \to$ длина$(P) = \int\limits_a^b \sqrt{1+\left(\frac{df(x)}{dx}\right)^2}dx)$.

Второй антецедент теоремы идентифицирует определяющее кривую выражение "точки(A,K)". Либо изначально кривая задана таким выражением, либо в контексте имеется равенство, дающее это выражение для P. Третий антецедент нужен для получения явной зависимости $y = f(x)$. Он выражает эквивалентность двух утверждений - утверждения о принадлежности "произвольной" пары (x, y) множеству координат точек кривой A, и результата $y = f(x)$ & $B(x)$ явного разрешения этого условия относительно неизвестной y. Такие эквивалентности в посылках теорем приемов записываются в виде равенств - для удобства компиляции. После разрешения условия на координаты точки относительно y остаются какие-то условия $B(x)$ на независимую переменную. Чтобы получить отрезок интегрирования, эти условия необходимо явно разрешить относительно x. Это и делает четвертый антецедент теоремы.

Заголовок приема - "второйтерм". Уровень срабатывания его выбираем сравнительно небольшим. Так как в решателе уже имеется прием для вычисления длины кривой, срабатывающий на третьем уровне, лучше положить уровень срабатывания нашего приема равным 2.

Кроме стандартных указателей "идентификатор(2 3 4)", "отображение(f B)", вводим указатели "новаяпеременная(x)" и "новаяпеременная(y)", объясняющие компилятору, что переменные x, y он может выбрать произвольным образом (но без совпадения их с уже имеющимися в задаче). При разрешении условия $B(x)$ относительно x вместо нестрогих неравенств могут появиться строгие. Чтобы обобщить прием и на эти случаи, добавляем указатели "вариант(фикс(4 2 1)меньше)", "вариант(фикс(4 2 2)меньше)". Они разрешают появляться строгим неравенствам вместо нестрогих в задании промежутка для x, не изменяя прочих действий приема.

Расставляем нормализаторы приема. Во-первых, для идентификации выражения "точки(A,K)" снабжаем переменную P (изначально идентифицируемую с кривой из выражения "длина(P)") нормализатором "смточки". Это - очень простой вспомогательный нормализатор, заменяющий P на выражение "точки(...)", если в контексте идентификации встречается равенство "P =точки(...)". Далее, для обработки левой части третьего антецедента вводим обращение к вспомогательной задаче на описание, разрешающей эту часть относительно y. Оно имеет следующий вид: "задача(6 тип(описать)полный явное буфер прямойответ упростить цель(неизвестная(y)))". Цель "буфер" добавлена для того, чтобы на некоторое время запомнить результат решения данной задачи и при повторных попытках ее решения, предприниаемых другими приемами, сразу извлекать из него готовый результат. Надобность в этом может возникнуть, если создать прием, предпринимающий попытку разрешения уравнения кривой относительно y для усмотрения распадения кривой на несколько ветвей и разбиения задачи на подзадачи вычисления длин отдельных ветвей.

Для обработки левой части четвертого антецедента вводим аналогичный указатель нормализации: "задача(5 тип(описать)полный явное буфер прямойответ упростить цель(неизвестная(x)))".

Наконец, для вычисления интеграла сопровождаем его указателем нормализации "задача(6 упростить)"; радикал под интегралом - указателем "задача(4 упростить одз)"; производную - указателем "задача(5 упростить одз)". Такое принудительное выделение этапов вычисления во многих случаях существенно

ускоряет получение результата.

11. Пусть в прямоугольной системе координат K множество точек A задается соотношением коорд$(A,K) = set_{xyz}(f(x,y,z) = 0$ & число(x) & число(y) & число$(z))$. Тогда для перехода к сферическим координатам можно записать следующую теорему приема:

    $\forall_{AKf}$(прямкоорд(K) & коорд$(A,K) = set_{xyz}(f(x,y,z) = 0$ & число(x) & число(y) & число$(z)) \rightarrow$ сферкоорд$(A,K) = set_{abc}(f(a\cos c\cos b, a\cos c\sin b, a\sin c) = 0$ & $0 \le a$ & $-\pi \le 2c$ & $2c \le \pi$ & $-\pi \le b$ & $b \le \pi))$.

    Так как решатель обычно устраняет знаменатели в уравнениях и неравенствах с дробями, то неравенства для широты домножены на 2 уже в теореме.

    Заголовок приема - "вывод". Поводом для его срабатывания пусть будет упоминание в посылках задачи выражения "сферкоорд(A K)". В этом случае уровень срабатывания можно сделать небольшим, например, равным 3. Лучше проявить предусмотрительность на тот случай, если соотношение для прямоугольных координат множества A появится в задаче несколько позднее, и добавить еще одну попытку применения приема на более высоком уровне - например, на шестом. В результате появится фильтр "уровень(3 6)". Далее добавляем стандартные фильтры "посылка", "тип(исследовать)". Чтобы избежать повторных применений приема, нужно ввести еще какой-то ограничитель. Пусть это будет фильтр, проверяющий отсутствие в посылках равенства, дающего выражение для сферических координат множества A через описатель "класс" - "не(контекст(посылка(x1)вид(x1 равно(сферкоорд(AK)x2))заголовок(x2 класс)))".

    Далее вводим указатели "отображение(f)" и "контрольвывода(сферкоорд(A K))". Последний из них и обязывает компилятор начать программу приема с усмотрения в задаче выражения "сферкоорд(A K)". Чтобы решатель не поменял местами левую и правую части равенства (при прочих равных условиях влево переносится более длинное выражение), добавляем указатель "примечание(ориентацияравенства)". Он вводит комментарий "ориентацияравенства" к новой посылке, блокирующий на последующих шагах перестановку частей равенства. Далее, вводим указатель "примечание(нормкоорд)". Он сопровождает посылку комментарием "нормкоорд", блокирующим попытки исключения описателя "класс" путем явного разрешения условия принадлежности этому классу. Такие попытки для множеств, заданных уравнениями, обычно бесполезны.

    Чтобы упростить уравнение для новых координат, сопровождаем утверждение под описателем "класс" нормализатором "задача(6 тип(описать) полный прямойответ явное упростить цель(неизвестная(a b c)))". Это накладывает достаточно сильные требования на уравнение множества - прием пытается получить явное соотношение, выражающее одну из сферических координат через две другие. Чтобы избежать слишком долгого времени его применения, вводим в заключение ограничитель трудоемкости - указатель "лимит(10000000)".

13.2.6 Создание общелогических приемов

Приведем ряд упражнений, в которых требуется реализовать на ГЕНОЛОГе различные общелогические приемы. В основном, эти приемы уже реализованы на ЛОСе, так что перед проверкой работоспособности созданного приема следует отключить

старую его версию. Она легко находится через оглавление программ. Для отключения приема можно поместить в начало его программы какой-либо тождественно ложный оператор, например, оператор "равно(0 1)". Разумеется, после тестирования нового приема эту вставку следует удалить, а также удалить новый прием.

1. Создать прием, исключающий повторяющиеся операнды конъюнкции;
2. Создать прием, отбрасывающий такой член дизъюнкции, который получается из другого ее члена добавлением конъюнкции некоторых утверждений;
3. Создать прием, реализующий логическое упрощение $(A \,\&\, B) \vee (A \,\&\, \neg B) \to A$;
4. Создать прием, исключающий повторяющиеся антецеденты кванторной импликации;
5. Создать прием, устраняющий квантор существования в антецеденте кванторной импликации;
6. Создать прием, исключающий связанную переменную x кванторной импликации, если консеквент этой импликации имеет вид $\neg(x = t)$, где t не содержит x;
7. Создать пакетный нормализатор "нормэквивалентно" для приведения к стандартному виду эквивалентностей. Занести в него несколько приемов общей стандартизации эквивалентности, перечисленных в соответствующем разделе оглавления программ;
8. Создать прием, преобразующий выражение "префикс(a префикс(b c))" в "конкатенация(набор(a b)c)";
9. Создать прием, сводящий задачу на доказательство кванторной импликации с антецедентами $A_1, \ldots, A_n$ и консеквентом B к задаче на доказательство утверждения B из списка посылок, пополненного указанными антецедентами.

Указания

1. Входим в оглавление базы приемов; выбираем раздел "Логические приемы" - "Общие приемы" - "Конъюнкция", и вводим в нем новый концевой пункт. Входим в это пункт, нажимаем "Ctr-ф", и приступаем к вводу теоремы приема. Чтобы устранять повторяющиеся конъюнктивные члены, достаточно теоремы $\forall_x(x \,\&\, x \leftrightarrow x)$. При компиляции преобразований замены для ассоциативно-коммутативных операций создается программа, извлекающая часть необходимых для идентификации операндов и сохраняющая без изменения прочие операнды. Поэтому наш прием будет применим к конъюнкциям любого числа утверждений. После ввода теоремы под ней появляется горизонтальная линия. Нажимаем Enter, и вводим текстовым редактором заголовок приема - "второйтерм". Далее появляется третье окно. В нем размещаем фильтр "уровень(0)", определяющий уровень срабатывания приема. В случае приемов сканирования задачи указание в третьем окне уровня срабатывания является необходимым. Четвертое окно игнорируем - нажимаем "Esc". Под ним появляется голубая линия, означающая, что ввод приема завершен. Необходимо сразу же сохранить прием. Если нет уверенности в том, что прием уже готов к компиляции, лучше

сохранить прием с помощью клавиши F4 - без компиляции. Это - обычная практика. Однако, в нашем случае прием крайне прост. Он не требует даже нормализаторов. Поэтому нажимаем сразу F3. Компиляция завершается успешно, о чем свидетельствует черная нижняя линия. Теперь можно посмотреть программу приема, нажав клавишу "Home". Входим в просмотр программы и нажимаем клавишу "курсор вверх". Она переводит к фрагменту программы, содержащему контрольную точку "прием(2)". Читая текст программы после контрольной точки, обнаруживаем, что здесь размещается старая версия приема, только что реализованного на ГЕНОЛОГе. Помещаем перед оператором "заменавхождения(...)" вставку "равно(0 1)", чтобы заблокировать работу старой версии. Теперь протестируем работу нового приема. Выйдем в задачник, выберем какой-либо его раздел, и создадим, например, задачу на описание с единственной неизвестной x и условием $(x = 0 \ \& \ x = 0) \ \vee \ x = 1$. Запустим процесс решения этой задачи клавишей "р". На первом же шаге сработает новый прием, заменяющий условие на $x = 0 \ \vee \ x = 1$. В заключение возвратимся к просмотру старой версии программы приема и уберем заглушку "равно(0 1)". Затем перейдем в базу приемов, удалим нажатием Ctr-Del новый прием, и удалим созданный для этого приема концевой пункт оглавления.

2. Теорема приема имеет вид $\forall_{ab}(a \ \vee \ (a \ \& \ b) \leftrightarrow a)$. При просмотре программы приема убеждаемся, что кроме выявления двух дизъюнктивных членов вида a, $a \ \& \ b$, она идентифицирует также случай, когда a - дизъюнкция нескольких членов, одновременно являющаяся конъюнктивным множителем текущего члена.

3. Теорема приема имеет вид $\forall_{AB}((A \ \& \ B) \ \vee \ (A \ \& \ \neg B) \leftrightarrow A)$.

4. Теорема приема имеет вид $\forall_{ABC}(\forall_x(A(x) \ \& \ A(x) \ \& \ B(x) \rightarrow C(x)) \leftrightarrow \forall_x(A(x) \ \& \ B(x) \rightarrow C(x)))$. Заметим, что антецеденты и консеквент необходимо представлять в виде функциональных переменных $A(x)$, $B(x)$, $C(x)$. Простые переменные A, B, C будут идентифицироваться как утверждения, не содержащие переменных связывающей приставки x. Заголовок приема - "второйтерм", единственный фильтр - "уровень(0)". Однако, в четвертом окне уже появляются элементы, необходимые для компиляции. Прежде всего, вводится указатель "кортежпеременных(х23)". Он означает, что теоремная переменная x будет идентифицироваться не с единственной переменной, а со связывающей приставкой произвольной длины. Так как выражения $A(x)$, $B(x)$, $C(x)$ нужно идентифицировать не с термами вида "значение(...)", а с произвольными термами, вводится указатель "отображение(х26 х27 х28)". Рекомендуется блокировать попытки компилятора рассматривать идентифицируемые кванторы с точностью до отрицания. Для этого вводим указатель "внешнийквантор(фикс(0 1))". Наконец, нужно объяснить компилятору, из каких соображений антецеденты разбиваются на три группы - два раза $A(x)$ и один раз $B(x)$. Нам нужно рассматривать $A(x)$ как произвольные единичные антецеденты. Поэтому вводим указатель "антецедент(х26 фикс(0 1))".

   Чтобы научиться аккуратно обращаться на ГЕНОЛОГе с функциональными переменными и кванторами, нужен некоторый опыт. Рекомендуется в затруднительных случаях находить аналогичные приемы. Иногда может понадобиться даже расширение ГЕНОЛОГа и развитие компилятора. Впрочем, приемы, требующие функциональных переменных, сравнительно редки.

5. Для исключения квантора существования, являющегося антецедентом кванторной импликации, можно использовать теорему вида: $\forall_{ABC}(\forall_x(\exists_y(A(x,y))\ \&\ B(x) \to C(x)) \leftrightarrow \forall_{xy}(A(x,y)\ \&\ B(x) \to C(x)))$. Чтобы x, y идентифицировались со связывающими приставками произвольной длины, вводим указатели "кортежпеременных(x23)", "кортежпеременных(x24)". Функциональные переменные A, B, C выделяем указателем "отображение(x26 x27 x28)". Наконец, указатели "внешнийквантор(фикс(0 1))". "внешнийквантор(фикс(0 1 3))" добавляем, чтобы кванторы общности и существования идентифицировались без попыток перехода к их отрицаниям.

6. Если консеквент кванторной импликации имеет вид $\neg(x = t)$, где x - переменная связывающей приставки, а выражение t не содержит x, то можно перенести в антецеденты равенство $x = t$, а консеквентом сделать отрицание одного из антецедентов. После этого переменная x легко устраняется. Теорему приема, выполняющего данные преобразования, можно записать в следующем виде: $\forall_{ABt}(\forall_{xy}(A(x,y)\ \&\ B(x,y) \to \neg(x = t(y))) \leftrightarrow \forall_y(A(t(y),y) \to \neg B(t(y),y)))$. Чтобы компилятор идентифицировал $B(x,y)$ как произвольный антецедент импликации, вводим указатель "антецедент(x27 фикс(0 1))". Если желательно выбрать какой-либо специальный антецедент, то достаточно ввести фильтр, отбирающий такой антецедент. Например, чтобы, по возможности, исключить появление отрицания в консеквенте, будем при наличии хотя бы одного отрицания в антецедентах брать в качестве $B(x,y)$ только антецедент с отрицанием. Соответствующий фильтр имеет вид "или(заголовок(фикс(0 1 5)не)не(контекст(антецедент(фикс(0 1)x4)заголовок(x4 не))))". Указатель "кортежпеременных(x24)" позволяет рассматривать произвольные списки y связанных переменных, дополняющие x. Указатель "внешнийквантор(фикс(0 1))" сопровождает идентифицируемый квантор общности; указатель "отображение(x19 x26 x27)" выделяет функциональные переменные A, B, t. Чтобы убрать возможное двойное отрицание в утверждении $\neg B(t(y),y)$, сопровождаем его нормализатором "нормлог". Заметим, что прием будет обрабатывать и вырожденные случаи - отсутствие антецедентов $A(x,y)$, отличных от $B(x,y)$, а также случай пустого списка y. Вместо кванторной импликации в последнем случае будет создана дизъюнкция.

7. Чтобы создать новый нормализатор "нормэквивалентно", используем вспомогательный интерфейс. Нажимаем клавишу Ctr-п. Так как нам нужен нормализатор общей стандартизации, название его можно пока не вводить. Оно будет создано автоматически. Достаточно выбрать окно "Нормализатор" и нажать левую кнопку мыши. Далее нажимаем левую кнопку мыши в окне "Нормализатор общей станартизации ...". Появляется курсор текстового редактора. Вводим название того логического символа, для которого предназначен нормализатор. В нашем случае это символ "эквивалентно". После нажатия Enter, завершающего ввод символа, появляются также число уровней срабатывания (по умолчанию 3) и указатели на наличие списка посылок и корневой режим обработки терма. Нажимаем Enter и возвращаемся в оглавление базы приемов, где обнаруживаем подраздел для нового нормализатора "нормэквивалентно". Здесь уже имеются приемы справочников, характеризующих нормализатор, а также корневой фрагмент программы нормализатора (пункт "Переключатель уровня").

Рассмотрим первые два простейших приема нормализатора, предоставляя читателю продолжить его развитие самостоятельно. Эти приемы имеют теоремы $\forall_x((x \leftrightarrow x) \leftrightarrow$ истина$), \forall_x((x \leftrightarrow \neg x) \leftrightarrow$ ложь$)$. Заметим, что консеквент теоремы приема имеет вид эквивалентности и заменяемая часть тоже имеет вид эквивалентности. Поэтому заменяемую часть необходимо заключить в скобки (иначе проблемы появятся уже на этапе работы с формульным редактором). Кроме того, заменяюшая часть указанных двух теорем представляет собой логическую константу, которую следует указать явно ("истина" вводится двойным нажатием "t", "ложь" - двойным нажатием "f". Если бы в первой теореме логическая константа отсутствовала, а консеквент имел вид $x \leftrightarrow x$), то компилятор создал бы программу, заменяющую текущий преобразуемый терм на самого себя.

8. Теорема приема имеет вид $\forall_{abc}($префикс$(a,$ префикс$(b,c)) = ((a,b);c))$. Точка с запятой обозначает здесь конкатенацию наборов. Хотя после ввода текста теоремы пара (a,b) будет перерисована без скобок, ее нужно вводить заключенной в скобки. Если не вводить никаких указателей, то прием откомпилирован не будет. Причиной этого послужит невозможность выбора в теореме точки привязки. Ведь по умолчанию выражение "префикс(A B)" идентифицируется с задающим набор термом достаточно произвольного вида - либо "набор(...)", либо "префикс(...)", либо "конкатенация(...)". Нет никакого логического символа, который обязательно встречался бы в каждом из указанных случаев и который можно было бы выбрать в качестве символа привязки. Поэтому вводим указатели "нормнабор(фикс(0 1))" и "нормнабор(фикс(0 1 2))", означающие, что символ "префикс" идентифицируется буквально. С ними компиляция приема становится возможной.

9. В этом примере мы встречаемся с ситуацией, когда перед выполнением преобразования необходимо выполнить проверку некоторого утверждения B относительно заданных посылок A. Ее можно было бы организовать, снабдив теорему приема антецедентом типа $\forall_x(A(x) \to B(x))$ и обратившись к вспомогательной задаче на его доказательство. Однако, удобнее применять другой способ - ввести в антецеденты теоремы только $B(x)$, а посылки $A(x)$ передавать вспомогательной задаче через специальные указатели. Тогда теорема приема, без учета сопровождающих указателей, приобретает несколько странный вид: $\forall_{AB}(B(x) \to \forall_x(A(x) \to B(x)))$. Вводим заголовок приема "второйтерм", а также уточняющие контекст фильтры: "уровень(3)"; "тип(доказать)"; "условие"; "корень". Для обработки антецедента вводим указатель "следствие(1)", определяющий обращение к вспомогательной задаче на доказательство. Максимальный уровень обращения совпадает с максимальным уровнем текущей задачи. Чтобы передать этой задаче в качестве дополнительных посылок конъюнкцию антецедентов $A(x)$, используем указатель "занесениепосылки(1 значение(x26 x23))". Добавляем стандартные указатели "кортежпеременных(x23)", "отображение(x26 x27)"и "внешнийквантор(фикс(0))".

13.2.7 Создание приемов для теоретико-множественных операций и отношений

Упражнения по програмированию на ГЕНОЛОГе обычно приводятся в виде списка задач, для которых недостаточно имеющихся в системе приемов. Однако, по мере

развития решателя, потребность в таких приемах может возникнуть, они будут занесены в базу приемов, и приводимые здесь упражнения станут утраченными. Чтобы избежать такой ситуации, в интерфейсе системы предусмотрен специальный ослабленный "учебный" режим запуска задач. В этом режиме блокируется срабатывание приемов ГЕНОЛОГа, имеющих указатель "ученик". По мере возможности, таким указателем будут снабжаться все приемы, воздействующие как-либо на процессы решения приводимых в книге учебных задач. Запуск решения задачи в учебном режиме обеспечивается нажатием клавиши "щ" (без трассировки) либо "Щ" (с трассировкой). Если таким образом запускается решение задачи, ранее хранившейся в задачнике, то данные о ней (ответ и время решения) будут изменены. Чтобы восстановить исходный их вид, достаточно повторно запустить решение задачи нажатием клавиши "о".

Во всех упражнениях этой книги, не оговаривая каждый раз особо, запускаем решатель только нажатием клавиши "щ" либо "Щ". Если в решателе обнаруживаются приемы с указателем "ученик", о которых в соответствующих разделах книги ничего не говорилось, то они были созданы уже после написания данных разделов.

1. Найти первую задачу подраздела задачника "Теория множеств - Задачи на преобразование" и просмотреть процесс ее решения по шагам, восстанавливая все применяемые преобразования. В том числе, определить все преобразования нормализаторов общей стандартизации (в общем режиме трассировки "по срабатываниям приемов" они на экран не выводятся).

2. Найти в разделе "Теория множеств - Задачи на описание" задачу номер 15 и просмотреть процесс ее решения по шагам, восстанавливая все применяемые преобразования.

3. Найти в разделе "Теория множеств - Задачи на доказательство" задачу номер 21 и просмотреть процесс ее решения по шагам, восстанавливая все логические переходы.

4. Ввести приемы, позволяющие решать задачи на преобразование выражений алгебры множеств к виду объединения пересечений. Использовать их для приведения к указанному виду выражения $(a \cup e \cup (b \cap c)) \cap (c \cup d \setminus b \cup f)$.

5. Применить решатель для упрощения выражения, полученного в предыдущем примере после "раскрывания скобок". Убедиться в том, что результат оказывается сложнее исходного выражения предыдущего примера, и добавить приемы, необходимые для восстановления этого выражения.

6. Добавить приемы, необходимые для решения задачи на нахождение всех таких множеств A, B, что выполнено условие $\exists_{xy}(A \setminus x \subseteq B \cap y \;\&\; B \setminus x \subseteq A \cap y)$.

7. Добавить приемы, необходимые для решения задачи на нахождение всех таких множеств A, B, C, что выполнено условие $\exists_x(\forall_y(C \subseteq y \to A \cup x \subseteq B \cap y))$.

8. Добавить приемы, необходимые для решения задачи на упрощение выражения $(C \cup D) \setminus A$ при выполнении посылок $A \subseteq B, \forall_x(A \subseteq x \;\&\; x \subseteq B \to C \setminus x = D \setminus x)$.

9. Добавить приемы, необходимые для решения задачи на нахождение примера таких значений x, y, z, что $x \subseteq y, y \subseteq z, \neg(y \subseteq x), \neg(z \subseteq y)$.

10. Найти еще какую-нибудь задачу на рассмотренные понятия алгебры множеств, не решаемую системой, и добавить необходимые для ее решения приемы.

Указания

1. Запускаем решение задачи с трассировкой "по срабатываниям приемов" (клавиша "р"). Первое преобразование заключается в устранении вложенных пересечений. Следующий переход - замена выражения $(a \cup b \setminus d) \cap (a \setminus (b \cup c)) \cap (d \setminus (a \cap b))$ на выражение $(a \cap d) \setminus (b \cup c)$. Прием, выполняющий данную замену, основан на теореме $\forall_{abc}(c \cap (b \setminus a) = (b \cap c) \setminus a)$. Для просмотра этой теоремы используем клавишу "б"; для возвращения к текущему кадру трассировки - клавишу "End". Сразу понятно, что основная часть содержательных преобразований была выполнена нормализаторами, к которым обращался прием. Это, безусловно, ускорило вычисления, но создало трудности в пошаговом просмотре решения.

   Если бы прием пользовался какими-либо крупными нормализаторами, формат которых содержит элемент "контрольнормализации", то можно было бы обратиться из текущего кадра трассировки к детальному просмотру цепочки их преобразований. Однако, в нашем случае использовались только нормализаторы общей стандартизации. На момент просмотра текущего кадра обращения к ним уже состоялись. Поэтому нужно повторить трассировку решения задачи с самого начала и постараться войти в пошаговый анализ работы нормализаторов рассматриваемого приема через отладчик ЛОСа. Выполняем следующие действия. Возвращаемся к просмотру задачи (Esc) и входим в оглавление приемов для выбора точки прерывания (клавиша "г"). Оказываемся в оглавлении базы приемов, у которого выделен пункт "Пересечение с разностью", содержащий нужный нам прием. Нажимаем "курсор вправо" и входим в просмотр данного приема. Для выбора точки прерывания перед его реализацией и запуска решения нажимаем "Ctr-Enter". Почти сразу появляется кадр отладчика ЛОСа, соответствующий начальным операторам приема. Из программы приема видно, что обращения к нормализаторам расположены внутри оператора "заменавхождения(...)". Поэтому выбираем точку прерывания перед реализацией данного оператора: нажимаем "курсор вниз"; с помощью клавиши "курсор вправо" выделяем синим цветом оператор "равно(x10 набор(x8))", предшествующий оператору "заменавхождения", и нажимаем "Enter". Отладчик переходит в кадр, непосредственно предшествующий обращениям к нормализаторам. Убеждаемся в том, что преобразуется нужный нам терм, выведя его на экран, например, путем просмотра значения переменной x2 - текущего вхождения в задачу. Из программы видно, что прежде всего будет применяться нормализатор "нормпересечение". Поэтому устанавливаем прерывание при обращении к нему: нажимаем клавищу "л"; вводим текстовым редактором символ "нормпересечение" и нажимаем "Enter". Появляется стартовый кадр работы нормализатора. Чтобы просмотреть цепочку изменений преобразуемого выражения, нажимаем клавиши "7" (появляется буква "з" и курсор текстового редактора), "1"(номер той переменной, изменения которой нас интересуют), и "Enter". Таким образом создана установка на просмотр изменений переменной x1, и каждое следующее нажатие клавиши "Enter" будет приводить к кадру, указывающему очередное изменение. Нажимаем эту клавишу. На экране появляется текст "Условие $a \cap (a \cup b \setminus d) \cap (d \setminus (a \cap b))$ заменяется на $(d \setminus (a \cap b)) \cap a$ (Безусловное поглощение)". Клавиша "б" позволяет посмотреть описание примененного приема.

Это преобразование уже действительно одношаговое: второй член пересечения поглощается первым. Снова нажимаем "Enter". Появляется очередной кадр: "Условие $(d \setminus (a \cap b)) \cap a$ заменяется на $(a \cap d) \setminus b$ (Пересечение с разностью)". Хотя это преобразование и несложное, оно уже не одношаговое.

Если бы мы захотели разложить его на составные шаги, то пришлось бы еще раз повторить трассировку, и на предыдущем кадре цепочки преобразований нормализатора установить прерывание перед входом в данный прием. Впрочем, есть и другие способы выйти на нужные точки преобразований с помощью отладчика ЛОСа. Можно вставить в программу представляющего интерес приема оператор "трассировка(стоп 0)" чтобы отслеживать все моменты его применения; если нужно отслеживать лишь часть таких моментов, то вставить оператор "или(A трассировка(стоп 0))", где A - оператор, ложный для представляющих интерес случаев, и т.п.

Продолжая начатую выше трассировку, нажимаем "Enter" и попадаем в кадр "Ответ: $(a \cap d) \setminus b$". Чтобы просмотреть теперь работу другого нормализатора, нажимаем "ф" и попадаем снова в отладчик ЛОСа. С помощью нажатия клавиши "PageUp" возвращаемся в просмотр программы приема сканирования задачи. Из нее усматривается, что далее будет применяться нормализатор "нормразность". Как и выше, входим в его стартовый кадр, устанавливаем трассировку по шагам изменения переменной x1 и прослеживаем остающиеся преобразования. В конечном счете, оказываемся в кадре трассировки задачи "по срабатываниям приемов", с которого и начали описанную цепочку действий.

Заметим, что обычно не возникает потребность в прослеживании вычислений, выполняемых нормализаторами общей стандартизации. Лишь в редких предметных областях, типа алгебры множеств, такие нормализаторы могут выполнять сколь-нибудь неочевидные действия. Поэтому специальный интерфейс слежения за их действиями и не был создан.

2. Задача имеет условия $a \cup x \setminus y = b \cap x$, $x \setminus a = x \cap y$ и неизвестные x, y. Запускаем процесс ее решения нажатием клавиши "р". Сразу же срабатывает прием, предпринимающий попытку разрешить оба уравнения сначала относительно y. Это - общелогический прием, реализованный на ЛОСе. Сопровождающий срабатывание текст "решение задачи на описание с несколькими неизвестными путем выражения одной из неизвестных через остальные" попросту копирует название концевого пункта оглавления программ, соответствующего ближайшей внешней контрольной точке "прием(...)" текущего оператора программы. Изменяя название пункта оглавления, будем изменять и сопровождающий текст.

   Следующее нажатие "Enter" приводит к кадру, заменяющему первое уравнение системы на конъюнкцию утверждений $a \subseteq b$, $a \subseteq x$, $x \setminus b \subseteq y$, непересек$(y, (b \cap x) \setminus a)$. Они явно разрешают первое уравнение относительно y. Примененный прием заменил равенство на два включения, а каждое включение обработал нормализатором "уравнсодержится". В отличие от предыдущего примера, нет необходимости повторно запускать решение задачи, чтобы посмотреть шаги работы данных нормализаторов. Прокручивая текущий кадр вниз, находим окна, соответствующие обращениям к нормализаторам. Выбираем, например, первое из них и располагаем так, чтобы верхняя горизонтальная линия окна находилась в верхней части экрана, причем над ней не оставалась прорисованной часть предыдущего окна. Для этого удобно применять клавиши "Ctr-курсор вверх" и

"Ctr-курсор вниз". В окне содержится текст "Применить нормализатор "уравнсодержится" к выражению: $a \cup x \setminus y \subseteq b \cap x$". Для повторения действий нормализатора нажимаем "Enter". Первый шаг состоит в развертке условия включения объединения. Оно преобразуется в конъюнкцию $a \subseteq b \,\&\, a \subseteq x \,\&\, x \setminus b \subseteq y$. Впрочем, здесь снова имелись обращения к нормализаторам. Окна этих обращений размещены снизу, и можно просмотреть цепочки их преобразований так же, как для текущего нормализатора. Глубина вложенности таких просмотров не ограничена. Если нужно оборвать просмотр на текущем уровне и вернуться к предыдущему уровню, нажимаем "End". Иначе возвращение произойдет автоматически, по нажатии "Enter" при получении ответа на текущем уровне.

Возвращаемся к прерванному просмотру основной траектории преобразований задачи. Следующее нажатие "Enter" дает разрешение относительно y второго уравнения. После цепочки несложных преобразований далее возникает кадр с сопровождающим текстом "Учет неизвестных, которые временно рассматривались как известные". Прокруткой выводим на экран расположенный над ним кадр, описывающий текущее состояние задачи: $x \setminus a \cup x \setminus b \subseteq y$, непересек$(y, b \cap x), a \subseteq x, a \subseteq b$. Видно, что относительно y имеем явное описание. Возникший кадр вызван срабатыванием приема усмотрения ответа. Этот прием обратился к процедуре "редакторответа", а она обнаружила в комментариях информацию о том, что переменная x представляет собой "отложенную" неизвестную. Поэтому была введена вспомогательная задача разрешения относительно x двух не зависящих от y утверждений $a \subseteq x, a \subseteq b$, к которым добавлено условие существования значений y, удовлетворяющих найденному явному описанию. Последнее сохраняется отдельно и будет впоследствии добавлено к ответу данной вспомогательной задачи.

Нажимаем "Enter" и входим в решение задачи с неизвестной x. Сразу срабатывает прием, устраняющий квантор существования. Он заменяется на утверждение непересек$(x \setminus a \cup x \setminus b, b \cap x)$. Далее идет цепочка несложных преобразований, приводящих к следующему ответу для x:

$a \subseteq x, a \subseteq b,$ неперсесек$(x, b \setminus a)$.

Наконец, возникает кадр обращения к вспомогательной задаче на редактирование объединенного ответа. В этом ответе имеется включение $a \subseteq b$, отсутствовавшее на момент разрешения уравнений относительно y. Теперь оно дает возможность упростить условие $x \setminus a \cup x \setminus b \subseteq y$, которое заменяется на $x \setminus a \subseteq y$. Затем выдается окончательный ответ.

3. Задача на доказательство имеет своими посылками утверждения $a \cap b = \emptyset, c \subseteq b, c \subseteq d, b \subseteq e, d \cap e \cap f \subseteq a$. Условием ее является утверждение $c \cap f = \emptyset$. Запускаем решение нажатием "р". Прежде всего, посылка $a \cap b = \emptyset$ заменяется на "непересек(a, b)", а условие - на "непересек(c, f)". Следующий кадр - расшифровка условия непересечения: оно заменяется на $\forall_g (g \in c \to \neg(g \in f))$. Еще одно нажатие "Enter" - и срабатывает прием, усматривающий ложность принадлежности $g \in f$. В результате условие задачи заменяется на $\forall_g (g \in c \to \neg$ложь$)$. Прием использовал проверочный оператор "усмнепринадлежит". Так как после действий этого оператора задача, по существу, оказывается решенной, рассмотрим их подробнее. Прокручиваем изображение до тех пор, пока не появится окно "Проверить утверждение $\neg(g \in f)$. Это и есть обращение к проверочному оператору. Добиваемся расположения верхней горизонтальной линии окна

в верхней части экрана и нажимаем "Enter". В отличие от работы нормализатора, представляющей собой цепочку шагов, работа проверочного оператора сводится к единственному шагу - срабатыванию приема, усматривающего истинность проверяемого утверждения. Поэтому текущий кадр уже дает ответ на вопрос о том, как была усмотрена истинность утверждения $\neg(g \in f)$. Текст "использование включения множеств" во втором сверху окне является ссылкой на примененный прием. Нажимаем клавишу "б" и находим теорему приема: $\forall_{abcd}(a \cap b \subseteq c \ \& \ d \in a \ \& \ \neg(d \in c) \rightarrow \neg(d \in b))$. Для возвращения нажимаем "End". Чтобы понять, как эта теорема была применена, нужно вывести на экран список посылок проверочного оператора. Выделяем правой кнопкой мыши верхнее окно, содержащее текст "Повторное применение процедуры", и нажимаем клавишу "п". Появляется искомый список, предшествующий проверяемому утверждению. Из него видно, что было использовано включение $d \cap e \cap f \subseteq a$. Окна обращений к вспомогательным проверочным операторам показывают, что использовались также утверждения $g \in d \cap e, \neg(g \in a)$. Можно войти в просмотр проверки этих утверждений, и т.д. Таким образом, трассировка проверочных операторов выполняется в обратном порядке - от заключения к посылкам. Рекомендуется довести ее в данном примере до конца самостоятельно.

4. Решатель имеет нормализатор "стандобъединение", выполняющий требуемые преобразования. Однако, пока нет целевой установки задач на преобразование выражений к виду объединения пересечений, и нет приема, который при наличии такой установки обращался бы к нормализатору "стандобъединение". Мы должны ввести то и другое. Прежде всего, выбираем логический символ для цели задачи. Например, можно взять в качестве такого символа название нормализатора "стандобъединение". Предварительно проверяем, что он еще не использован в целевых установках. Для этого входим в редактор ЛОСа через символ "стандобъединение" и нажимаем F3. Попадаем в окно справочной информации по символу, которое оказывается пустым. Сохраняем в нем краткий текст об использовании символа в качестве цели задач на преобразование.

   Теперь нужно создать прием, обращающийся к оператору "стандобъединение" в задачах на преобразование, имеющих одноименную цель. Сразу же возникает вопрос о символе, при появлении которого в задаче должна выполняться попытка применения приема. Можно было бы отказаться от фиксации этого символа вообще - в произвольной задаче на преобразование выполнять попытку преобразований, если имеется цель "стандобъединение". Однако, утяжелять без особой необходимости группу общих приемов нежелательно - лучше создать несколько приемов, связанных с различными символами частной предметной области, и инициировать попытки применения приема только для задач этой области. В нашем случае придется создать приемы, связанные с теоретико-множественными операциями "пересечение", "объединение" и "разность". Они устроены почти одинаково, так что можно ввести какой-либо один, а прочие получать из его копий небольшими коррекциями. Такая практика при программировании на ГЕНОЛОГе применяется сравнительно часто.

   Входим в оглавление базы приемов и выбираем подходящий раздел - пусть, например, это будет подраздел "объединение" раздела "Теория множеств". Создаем новый концевой пункт - например, "Обращение к нормализатору СТАНДОБЪЕДИНЕНИЕ". Входим в этот пункт и начинаем ввод теоремы приема.

Нажимаем "Ctr-ф", после чего возникает курсор формульного редактора. Вводим теорему приема (например, для символа "пересечение"): $\forall_{abc}(c = a \cap b \rightarrow a \cap b = c)$. Антецедент теоремы означает присвоение переменной c результата применения нормализатора "стандобъединение" к выражению $a \cap b$. Обращение к этому нормализатору будет введено ниже. В приниципе, можно было бы не использовать вспомогательной переменной c и ввести теорему $\forall_{ab}(a \cap b = a \cap b)$, снабдив ее тем же самым обращением к нормализатору. Однако, такой стиль программирования на ГЕНОЛОГе нежелателен - вспомогательные переменные для результатов промежуточных преобразований упрощают ссылки на них из фильтров приема.

По завершении ввода формулы нажимаем "Enter". Под теоремой прорисовывается горизонтальная черта. Теперь нужно ввести заголовок приема. Так как речь идет о приеме замены "слева направо", то нужный заголовок - "второйтерм". Чтобы ввести его, нажимаем снова "Enter", и далее пользуемся текстовым редактором.

Еще одно нажатие "Enter", и появляется третье окно. В нем нужно ввести фильтры приема. Обязательно нужно указать уровень срабатывания приема. Например, вводим текст "уровень(2)". Далее указываем целевой контекст: "тип(преобразовать)", "цель(стандобъединение)", "корень". Последний фильтр разумно ввести, чтобы прием применялся лишь ко всему терму, а не к его многочисленным подтермам. Нажимаем "Enter" и переходим к набору содержимого четвертого окна.

Выделяем антецедент теоремы указателем "идентификатор(1)" и нажимаем "Enter". Появляется голубая горизонтальная черта. Она означает, что введенный прием пока не сохранен. Его нужно немедленно сохранить, нажав F4. Иначе, при случайном выходе из текущего кадра (например, в оглавление базы приемов), вся проделанная работа будет утеряна. После того, как прием сохранен, нижняя черта становится красной. Это значит, что прием пока не откомпилирован. Перед компиляцией нужно еще ввести обращение к нормализатору. Выделяем в антецеденте подтерм $a \cap b$ - мышью либо клавишами курсора, либо и тем, и другим. Затем нажимаем "Enter". Под красной чертой появляется курсор текстового редактора. Вводим название нормализатора "стандобъединение" и нажимаем "Enter". Чтобы отменить выделение подтерма, нажимаем "пробел". Теперь прием можно откомпилировать, нажав клавишу F3. Одновременно будет сохранено изменение приема - добавление к нему нормализатора. Если здесь компилировать не через F3, а через F5, то изменение будет утеряно. Рекомендуется ознакомиться с результатом компиляции - нажать "Home" для перехода в просмотр программы ЛОСа, а затем вернуться в ГЕНОЛОГ через "End".

Теперь переходим к проверке работы созданного приема. Входим в задачник и вводим указанную в упражнении задачу. Целевую установку ее создаем с помощью текстового редактора: нажимаем "Ctr-ц" и далее вводим текст "преобразовать стандобъединение одз". Условие $(a \cup e \cup (b \cap c)) \cap (c \cup d \setminus b \cup f)$ вводим обычным образом. Применяя трассировку "по приемам", на первом же шаге обнаруживаем срабатывание созданного нами приема. Рекомендуется провести трассировку по шагам работы нормализатора "стандобъединение", войдя в нее через отладчик ЛОСа. Для этого придется заново запустить решение задачи.

Если бы мы ввели в целевую установку задачи также символ "упростить", то обнаружили бы зацикливание: после каждого раскрывания скобок снова выполняется обратная "свертка" выражения. Рекомендуется немного усовершенствовать прием, чтобы в таких ситуациях (впрочем, уже по постановке задачи противоречивых) зацикливание не возникало.

Вернемся к трассировке решения нашей задачи. После следующего нажатия "Enter" обнаруживаем, что требуемый вид выражения нарушен: произошло обращение к процедуре завершающего редактирования ответа, которая решила вынести часть переменных за скобки. Это противоречит цели "стандобъединение", и нужно заблокировать обращение к указанной процедуре. Чтобы найти ее программу, не выходя из трассировки нажимаем "б". На экране появляется концевой пункт оглавления программ "Завершающее редактирование ответа". Он будет сохранен в оглавлении программ как "текущий пункт". Чтобы войти в редактирование программы этого пункта, возвращаемся к кадру трассировки (клавиша "End"); обрываем трассировку ("Esc") и выходим на главное меню ("End"). Далее входим в оглавление программ ("л") и снова оказываемся на пункте "Завершающее редактирование ответа". Нажимаем "курсор вправо", чтобы выйти в просмотр программы. Сбрасываем режим выделения операторов нажатием клавиши "о", и для редактирования программы нажимаем "р". В начале программы уже имеется цепочка операторов вида "не(цель(х1 ...))". Добавляем к ним оператор "не(цель(х1 стандобъединение))". При повторной трассировке решения задачи убеждаемся, что требуемый вид ответа не нарушен.

5. Чтобы создать задачу на упрощение полученного выше выражения, выполняем следующие операции: выделяем левой кнопкой мыши ответ предыдущей задачи; высвобождаем на экране место для новой задачи и нажимаем "з" (появляется верхняя линия новой задачи); вводим цель "Упростить в о.д.з."; выделяем новую задачу правой кнопкой мыши; нажимаем "Insert" для копирования выделенного ответа в качестве условия новой задачи. Запускаем решение задачи нажатием "о" и получаем ответ $a \cap (c \cup f) \cup c \cap (b \cup e) \cup e \cap f \cup (d \cap (a \cup e)) \setminus b$. Очевидно, он сложнее исходного выражения. Нужно подобрать одно или несколько простых тождеств, которые могли бы продолжить упрощение, и создать их приемы. Для первых трех членов указанного выше объединения можно усмотреть возможность следующей перегруппировки: $(a \cup e) \cap (c \cup f) \cup b \cap c$. Это подсказывает такое тождество: $\forall_{abcde}((a \cap (b \cup c)) \cup (b \cap (d \cup e)) \cup c \cap d = ((a \cup d) \cap (b \cup c)) \cup c \cap d)$. Рекомендуем читателю создать соответствующий прием и посмотреть, к чему приводит его срабатывание.

6. Вводим задачу на описание, имеющую неизвестные A, B и единственное условие $\exists_{xy}(A \setminus x \subseteq B \cap y \ \& \ B \setminus x \subseteq A \cap y)$. Начинаем трассировку ее решения. В некоторый момент срабатывает прием, предпринимающий попытку явного разрешения подкванторного утверждения относительно x, y. Продолжаем трассировку этой попытки. Сначала список неизвестных сужается до единственной неизвестной y; после разрешения относительно y - сопровождающие утверждения разрешаются относительно x. В заключение предпринимается редактирование объединенного ответа $A \setminus x \cup B \setminus x \subseteq y, A \setminus B \cup B \setminus A \subseteq x$. Заметим, что обе неизвестные x, y здесь - несущественные, так как имеется цель (параметры xy). Поэтому они могли бы быть устранены, однако приемов для этого

нет - решатель ограничивается переходом к симметрической разности $A \triangle B$ и выдает ответ $A \triangle B \subseteq x, x - \text{set}, y - \text{set}, (A \cup B) \setminus x \subseteq y$. Далее происходит возвращение к исходному условию существования, которое теперь приобретает вид $\exists_{xy}(A \triangle B \subseteq x, x - \text{set}, y - \text{set}, (A \cup B) \setminus x \subseteq y)$. Здесь опять появляется очевидная возможность исключить сначала y, а затем x. Однако, приема для этого нет, и в некоторый момент решатель переходит к неявной зависимости от y, заменяя включение $(A \cup B) \setminus x \subseteq y$ на $A \cup B \subseteq x \cup y$. Преобразования такого рода продолжаются; в конце концов существование требуемых значений x, y становится неочевидным, и решатель выдает "отказ".

Приведенная траектория позволяла решателю в двух точках добиться успеха. В первый раз - при редактировании ответа вспомогательной задачи мог сработать прием, исключающий несущественную неизвестную X, для которой имеются лишь условия вида $P \subseteq X$, $X - \text{set}$. Второй раз - при рассмотрении квантора существования можно было из тех же соображений отбросить связанные переменные. Хотя для данной задачи достаточно было бы лишь одного из таких двух приемов, однако каждый из них мог бы быть полезен в других ситуациях независимо от другого. Первый прием основан на теореме $\forall_a(a - \text{set} \rightarrow \exists_x(a \subseteq x \ \& \ x - \text{set}))$. Заголовок приема - "связка". Фильтры - "уровень(0)", "тип(описать)", "не(входит(x23 x1))". Второй прием основан на теореме $\forall_{AB}(\exists_{xy}(x - \text{set} \ \& \ A(y) \subseteq x \ \& \ B(y)) \leftrightarrow \exists_y(B(y)))$. Он имеет заголовок "второйтерм"; фильтр "уровень(0)" и указатели "отображение(x26 x27)", "кортежпеременных(x24)". Последний указатель допускает пустоту списка y, так что прием будет работать и для единственной переменной x. Рекомендуем самостоятельно создать оба приема и проверить, что они позволяют решить задачу.

7. Создаем задачу на описание, имеющую условие $\exists_x(\forall_y(C \subseteq y \rightarrow A \cup x \subseteq B \cap y))$ и неизвестные A, B, C. Начинаем трассировку решения этой задачи. После ряда общелогических преобразований в условии появляется подутверждение $\forall_y(y - \text{set} \rightarrow \neg(C \subseteq y))$. Оно сохраняется достаточно долго, чтобы понять, что решатель не имеет приема, усматривающего его ложность. Отсутствие такого приема доводит дело даже до разбора случаев. Поэтому первый шаг проработки задачи ясен - нужно ввести прием, основанный на теореме $\forall_A(\neg(\forall_x(x - \text{set} \rightarrow \neg(A \subseteq x))))$. Прием имеет заголовок "второйтерм"; уровень срабатывания его выбираем равным 0. Введя прием и откомпилировав его, повторяем трассировку задачи. После цепочки преобразований возникают три условия: $A \subseteq B$, $\exists_x(\forall_y(y - \text{set} \ \& \ C \subseteq y \rightarrow x \subseteq y) \ \& \ x - \text{set} \ \& \ x \subseteq B)$, $\forall_y(y - \text{set} \ \& \ C \subseteq y \rightarrow A \subseteq y)$. Здесь срабатывает прием, обращающийся к вспомогательной задаче для разрешения относительно x утверждения под квантором существования. Она имеет условия $\forall_y(y - \text{set} \ \& \ C \subseteq y \rightarrow x \subseteq y)$ и $x \subseteq B$. Попытка разрешить относительно y утверждения под квантором общности ничего не дает - после ряда преобразований этот квантор возникает опять. На некотором уровне всплывает попытка решать задачу накоплением необходимых условий, пока они не сложатся в достаточные. Она тоже безуспешна. С другой стороны, рассматриваемая кванторная импликация, очевидно, эквивалентна условию включения множества x в множество C. Эта эквивалентность представляется настолько простой, что для нее стоит ввести отдельный прием. Он имеет теорему $\forall_{AB}(A - \text{set} \ \& \ B - \text{set} \rightarrow \forall_x(x - \text{set} \ \& \ A \subseteq x \rightarrow B \subseteq x) \leftrightarrow B \subseteq A)$, заголовок приема - "второйтерм". Антецеденты обрабатываются проверочны-

ми операторами; уровень срабатывания выбираем равным 1. После ввода и компиляции приема еще раз повторяем трассировку задачи. Теперь уже она доводится до ответа: $A - \text{set}, B - \text{set}, A \subseteq B \cap C, C - \text{set}$.

Рекомендуем далее отключить последний прием и попробовать добавить приемы, с помощью которых решатель смог бы реализовать попытку накопления необходимых условий, пока они не сложатся в достаточные.

8. Вводим задачу на преобразование, имеющую посылки $A \subseteq B$, $\forall_x(A \subseteq x \ \& \ x \subseteq B \rightarrow C \setminus x = D \setminus x)$ и условие $(C \cup D) \setminus A)$. Очевидно, что для ее решения нужно сначала использовать кванторную импликацию и вывести равенство $C \setminus A = D \setminus A$, а затем усмотреть возможность применить данное равенство к преобразуемому выражению. Пока решатель не делает ни того, ни другого. Как мы знаем, имеется общелогический прием, предпринимающий попытку использовать кванторную импликацию для вывода следствий. Однако, для его срабатывания требуется, чтобы выводимое утверждение не содержало новых выражений. Поэтому сначала обеспечим вывод вспомогательных посылок "актив($C \setminus A$)", "актив($D \setminus A$)", указывающих, что данные выражения представляют интерес. Теорема приема имеет вид $\forall_{AB}$(актив($A \setminus B$)); заголовок приема - "вывод". Чтобы активизировать прием при обнаружении в условии задачи выражения $(A \cup C) \setminus B$, вводим указатель "контрольвывода(разность(объединение(х26 х28)х27))". Указываем в фильтрах контекст срабатывания: "уровень(2)", "тип(преобразовать)", "цель(упростить)". Далее указываем, что задача имеет кванторную посылку, консеквент которой содержит разность, пересекающуюся по переменным с преобразуемым выражением: "контекст(посылка(х3)заголовок(х3 длялюбого)последнийсимвол(х3 равно) подчинено(х4 последнийоперанд(х3))символ(х4 разность)пересекаются(параметры(х4)параметры(теквхожд)))".

После компиляции приема убеждаемся в выводе необходимых утверждений. Заметим, что эти утверждения на экране не прорисовываются. Чтобы их увидеть, нужно перейти к показу полного списка посылок нажатием клавиши "ы". Впрочем, косвенной информацией о их появлении служит вывод следствия $C \setminus A = D \setminus A$.

Следующий шаг - использование выведенного равенства - можно обеспечивать различными средствами. Ограничимся простейшим - создадим прием с теоремой $\forall_{ABCDE}(A \setminus B = C \ \& \ E = C \cup D \setminus B \rightarrow (A \cup D) \setminus B = E)$. Он усматривает посылку $A \setminus B = C$ и предпринимает попытку применить нормализаторы общей стандартизации к выражению, полученному из $(A \cup D) \setminus B$ заменой "неявного" подвыражения $A \setminus B$ на C. Если результат E получился короче исходного выражения, то реализуется замена. Заголовок приема - "второйтерм"; фильтры - "уровень(3)", "короче(х30 фикс(0 1))"; указатель - "идентификатор(2)". Выражения $D \setminus B$ и $C \cup (D \setminus B)$ обрабатываются соответствующими нормализаторами общей стандартизации.

Конечно, предложенный "учебный" вариант проработки предложенной задачи не претендует на особую эффективность и общность. Можно предложить читателю развить в решателе общий аппарат для решения задач указанного типа - на упрощение выражений алгебры множеств при наличии в посылках дополнительных соотношений. Здесь был бы полезен специальный нормализатор, выделяющий в преобразуемом выражении вхождения заданного подвыражения,

извлеченного из внешнего контекста.

9. Вводим задачу на описание, имеющую неизвестные x, y, z и условия $x \subseteq y$, $y \subseteq z$, $\neg(y \subseteq x)$, $\neg(z \subseteq y)$. Цели ее определяем с помощью пункта "Найти пример значений неизвестных" оглавления целевых установок. При трассировке решения обнаруживается, что происходит последовательный ввод параметрических описаний, преобразующих задачу к новым неизвестным. Например, условие $x \subseteq y$ и неизвестная y исключаются с помощью параметрического описания, представляющего y как $x \cup a$; аналогичным образом исключаются условие $x \subseteq z$ и неизвестная z, и т.д. По ходу дела предпринимается разбор случаев. Доведя трассировку до получения первого отказа на один из подслучаев, обнаруживаем задачу с условиями $\neg(f = c)$, $\neg(f \in x)$, $\neg(f \in d)$, $\neg(c \in x)$, $g \subseteq g \cup x \cup \{c, f\}$. Видно, что для условий непринадлежности параметрические описания введены не были, и это подсказывает ввод соответствующего приема. Его теорема имеет вид $\forall_{bc}(c - \text{set} \rightarrow \neg(b \in c) \leftrightarrow \exists_a(a - \text{set} \ \& \ c = a \setminus \{b\}))$; заголовок приема - "параметризация". Фильтры приема вводим по аналогии, например, с приемом, дающим параметрическое описание условия принадлежности: "уровень(5)", "условие", "тип(описать)", "или(цель(пример) цель(параметризация)) неизвестная(x3) не(входит(x3 x2))". Антецедент выделяем указателем "блокпроверок(1)".

   После создания указанного приема повторяем трассировку до получения первого отказа. Оказывается, что на этот раз не решена задача подбора примера таких c, g, что $\neg(c = g)$. Вводим прием с теоремой $\forall_{ab}(a = 0 \ \& \ b = 1 \ \& \ \rightarrow \neg(a = b))$ и заголовком "подборзначений". Фильтры приема определяем так: "уровень(1) условие тип(описать) цель(пример) неизвестная(x1) неизвестная(x2) не(контекст(новоеусловие(x3) или(входит(x1 x3) входит(x2 x3))))". Указатель "подборзначений(1 2)" показывает, что оба антецедента используются для замещения в задаче условия $\neg(a = b)$. Указатель "новый" стоит ввести для того, чтобы отслеживать момент исключения прочих условий задачи, содержащих переменные a, b.

   После создания указанного приема решатель доводит задачу до ответа: $z = \{0, 1\}, y = \{1\}, x = \emptyset$.

13.2.8 Создание приемов для прочих понятий алгебры множеств

1. Ввести приемы, необходимые для решения относительно x, y уравнения $x \times y = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$.

2. Ввести новый тип целевой установки задач на преобразование: "представить теоретико-множественное выражение в виде прямого произведения". Научить систему решать задачи на представление в виде прямого произведения перечня пар - например, $\{(1, 1), (1, 2), (2, 1), (2, 2)\}$.

3. Ввести приемы, необходимые для разрешения относительно неизвестной x условия $\forall_y(y \in a \ \& \ x \in y \rightarrow \forall_z(z \in b \rightarrow x \in z))$.

4. Ввести задачу на нахождение всех разбиений множества $\{1, 2, 3\}$. Создать приемы, необходимые для ее решения.

5. Ввести приемы, необходимые для решения уравнения:

   слойсемейства$(\{\{1,2\},\{1,3\},\{2,4\}\},\{1,2,3,4\},x)=\{1,2\}$.

6. Ввести прием, заменяющий условие $(a_1,\ldots,a_n)\in B_1\times\ldots B_n$ на условие $a_1\in B_1$ & $\ldots$ & $a_n\in B_n$ для произвольного натурального $n\geq 2$.

Указания

1. Вводим уравнение и убеждаемся, что система его не решает. В разделе "Прямоепроизведение" базы приемов находим приемы, связанные с решением уравнений, одна из частей которых имеет вид прямого произведения. Они обращаются к оператору "видпрямоепроизведение", предпринимающему попытку представить в виде прямого произведения и другую часть. Включаем трассировку и выходим на момент обращения к оператору "видпрямоепроизведение", Таким образом убеждаемся, что система в данной задаче действительно к нему обращается. Далее остается лишь пополнить оператор "видпрямоепроизведение", чтобы он мог разложить на множители выражение $\{(1,1),(1,2),(2,1),(2,2)\}$. Заметим, что для разложения в прямое произведение множества пар имеется очень простой способ - нужно найти первую и вторую проекции, перемножить их и сравнить с исходным множеством. Реализуем этот способ на ГЕНОЛОГе. Пусть рассматривается множество пар A. Во-первых, нам нужно найти обе его проекции. Введем для этого антецеденты $a=$ Проекция$(A,1)$, $b=$ Проекция$(A,2)$. Если A задано конечным списком, то обращение к нормализатору "нормПроекция" позволит вычислить a и b. Так как включение $A\subseteq a\times b$ заведомо выполнено, достаточно проверить обратное включение, введя в теорему приема антецедент $a\times b\subseteq A$. Применим к левой части включения нормализатор "нормпрямоепроизведение(замечание(нормперечень))", который преобразует произведение двух конечных списков в список пар. Сам антецедент выделим указателем "блокпроверок". Тогда он будет обрабатываться проверочным оператором "усмсодержится". Наконец, в качестве консеквента теоремы приема возьмем равенство $A=a\times b$. Окончательно, теорема приема будет иметь вид:

   $\forall_{Aab}(a=$ Проекция$(A,1)$ & $b=$ Проекция$(A,2)$ & $A\subseteq a\times b\rightarrow A=a\times b)$.

   Заголовок приема - "замена(второйтерм видпрямоепроизведение)". Введя и откомпилировав прием, проверям, что уравнение системой решается. Созданный прием сохраняем, так как он понадобится уже в следующей задаче.

2. Прежде всего, нужно выбрать логический символ, который будет кодировать указанную цель задачи на преобразование. Например, пусть это будет символ "прямоепроизведение". Проверяем по связанной с символом справочной информации, что он ранее не был использован как элемент целевой установки. Теперь нам нужно пополнить меню целевых установок системы. Входим в какую-либо из задач задачника и вызываем данное меню для редактирования (клавиша Shift-ц). Входим в подраздел "преобразовать выражение", вводим новый концевой пункт "разложить в прямое произведение", нажимаем "курсор вправо" и "Enter", после чего вводим текстовым редактором кодовый символ "прямоепроизведение". При выборе данного пункта оглавления система будет обращаться к справочнику "смцель", который должен организовать диалог ввода параметров целевой установки. Чтобы получить образец приемов такого справочника,

выбираем какой-либо другой концевой пункт оглавления и находим его кодовый символ. Например, пусть это будут пункт "разложить перестановку в произведение независимых циклов" и его кодовый символ "циклперест". Находим прием справочника "смцель" для символа "циклперест" и копируем его в программу символа "прямоепроизведение". В этой копии нужно заменить ссылку на текст, который будет выдаваться на экран при выборе целевой установки. Такой текст (т.е. "представить теоретико-множественное выражение в виде прямого произведения") нужно создать во 2-м инф.блоке. Входим в подраздел "ресурсы и установки" главного меню; нажимаем "и" и вводим номер "2" нужного информационного блока. После нажатия "Enter" оказываемся в корневом каталоге блока. Мы должны выбрать какой-либо логический символ S, такой, что из корневого каталога нет цепочки переходов по символам S, "слово". Пытаемся угадать S. Например, пусть сначала будет S = "прямоепроизведение". Нажимаем "курсор вниз" и вводим данный символ. Оказываемся в некотором ранее созданном указателе - списке. Перехода по символу "слово" из него нет, так что выбранное S нам подходит. Нажимаем "Insert" и вводим символ "слово". Сверху появляется меню, в котором выбираем пункт "Ч.-б. текст". Наконец, вводим текстовым редактором шаблон "Представить теоретико-множественное выражение в виде прямого произведения". Набор завершаем нажатием "Enter", и далее нажимаем "End" для возвращения в главное меню.

Роль ссылки на введенный текстовый шаблон играет символ "прямоепроизведение". Возвращаемся в скопированную программу справочника "смцель" и заменяем в операторе "смцель(набор(Выч)x1 x2 x3)" символ "Выч", который представлял собой ссылку на текстовый шаблон исходной программы, на символ "прямоепроизведение". В новой программе нужно также скорректировать список целей вводимой задачи - вместо старого комплекта "упростить, одз, циклперест" нужен комплект "упростить, одз, прямоепроизведение". На этом коррекция справочника "смцель" завершена. Можно попробовать воспользоваться оглавлением целей и ввести новую задачу. Например, пусть это будет задача на разложение выражения $\{(1,1),(1,2),(2,1),(2,2)\}$. Однако, если после набора задачи выйти в оглавление задачника, а затем снова вернуться в просмотр задачи, то окажется, что текстовый шаблон изменился - теперь он имеет вид "Упростить выражение". Нажатием "Ctr-ц" можно убедиться, что целевая установка введена верно, и причину рассогласования следует искать в программах интерфейса задачника. Возвращаемся в главное меню, входим в оглавление программ и выбираем раздел "Интерфес просмотра и редактирования задач" - "Процедура ЭЛЕМЕНТЫЗАДАЧИ" - "Стандартные комплекты целей задач на преобразование". В этой точке находится программа, которая усматривает один из стандартных комплектов целей задачи и формирует для его прорисовки соответствующую текст-формульную строку. Чтобы долго не искать нужный нам случай, будем действовать по аналогии. Нажимаем F4 и вводим символ "циклперест" - элемент целевой установки, который мы выше использовали как образец. Появляется фрагмент программы "или(цель(x1 разложитьнамножители)цель(x1 циклперест)...)". Добавляем в эту дизъюнкцию оператор "цель(x1 прямоепроизведение)". Чтобы определить кодовый символ x7 нужного нам текстового шаблона, разветвляем последний пункт цепочки вложенных условных выражений в правой части присвоения: заменяем символ "видеоключ" на терм "вариант(цель(x1 прямоепроизведение)прямоепроизведение видеоключ)". Те-

перь можно вернуться в просмотр нашей задачи и убедиться, что текстовый шаблон прорисован верно.

Чтобы при решении задачи с целью "прямоепроизведение" произошло обращение к нормализатору "видпрямоепроизведение", нужно ввести специальный прием. К сожалению, здесь нет никакого конкретного логического символа, появление которого в преобразуемом выражении инициировало бы попытку применения данного приема. Поэтому возможны два пути: либо создать на ЛОСе общий прием задач типа "преобразовать", который будет срабатывать при обнаружении цели "прямоепроизведение", либо выделить несколько типичных символов - заголовков выражений, к которым будет применяться прием, и создать для них различные приемы на ГЕНОЛОГе.

Реализуем второй способ. В нашем случае заголовком преобразуемого выражения служит символ "перечень". Поэтому берем следующую теорему приема: $\forall_{ab}(a = \{;b\} \rightarrow \{;b\} = a)$. Антецедент выделяем указателем "идентификатор(1)", к правой его части применяем нормализатор "видпрямоепроизведение". Вводим фильтр "заголовок(x1 прямоепроизведение)", контролирующий успех попытки разложения.

Согласно предыдущему упражнению, система теперь должна решить введенную задачу. Однако, после запуска процесса решения оказывается, что она зациклилась. Это обнаруживаем по затянувшейся паузе: нажимаем "Break" и видим, что после получения прямого произведения $\{1,2\} \times \{1,2\}$ применяется прием "элиминация прямого произведения перечней", выполняющий обратное преобразование. Чтобы данный прием учитывал нашу новую целевую установку, вводим в него дополнительный фильтр: "или(не(тип(преобразовать)) не(цель(прямоепроизведение)) посылка не(корень))". Снова запускаем процесс решения, и получаем ответ.

3. Вводим условие задачи и убеждаемся, что система ее не решает. Чтобы определить, на каком этапе возникают затруднения, начинаем трассировку. Система преобразует исходное утверждение к виду дизъюнкции $\forall_y(y - \text{set} \ \& \ y \in a \rightarrow \neg(x \in y)) \vee \forall_z(z \in b \rightarrow (z - \text{set} \ \& \ x \in z))$. Утверждение $z-\text{set}$ в консеквенте второго дизъюнктивного члена введено системой для сопровождения по о.д.з. По этой причине преобразование второй кванторной импликации в конъюнкцию импликаций блокируется. На следующем шаге срабатывает прием, преобразующий первую кванторную импликацию к виду отрицания принадлежности объединению семейства множеств:

$$\neg(x \in \bigcup_{y,y-set,y\in a} y).$$

Следующий шаг - разбор случаев по дизъюнкции. Рассмотрение первого подслучая сразу приводит к получению частичного ответа. Однако, при рассмотрении второго подслучая система не может перейти к условию принадлежности неизвестной пересечению семейства. После нескольких безуспешных попыток применить прочие приемы здесь выдается отказ. Таким образом, наша задача проясняется: нужно создать прием, переформулирующий кванторную импликацию $\forall_z(z \in b \rightarrow (z - \text{set} \ \& \ x \in z))$ в терминах принадлежности пересечению семейства. Прежде всего, попробуем найти в базе приемов близкие приемы. Переходим в раздел "Теория множеств" - "Семействамножеств" - "Перечесениевсех" - "Кванторная свертка в условие принадлежности пересечению семейства

множеств". Оказывается, что там уже есть прием, выполняющий аналогичное преобразование. Он применяется к утверждениям вида $\forall_x(f(x) \to a \in g(x))$. Однако, применение его требует усмотрения непустоты множества таких x, что истинно $f(x)$. В нашем случае непустота не гарантирована, и требуется более общий прием. Чтобы учесть возможность появления в консеквенте заменяемой импликации дополнительных условий, нужных для сопровождения по о.д.з., будем считать, что она имеет вид $\forall_x(A(x) \to a \in f(x)\ \&\ B(x))$. Тогда получаем следующую теорему приема:

$\forall_{ABfa}(\forall_x(A(x) \to a \in f(x)\ \&\ B(x)) \leftrightarrow \forall_x(A(x) \to B(x))\ \&\ (\neg(\exists_x(A(x)))\ \vee \exists_x(A(x))\ \&\ a \in \bigcap_{x,A(x)} f(x)))$.

Утверждение $\exists_x(A(x))$ добавлено к условию принадлежности пересечению семейства как сопровождающее по о.д.з.

В фильтрах приема указываем, что выражение, идентифицируемое с a, содержит неизвестные, а прочие подвыражения - не содержат. Переменные A, B, f выделяем указателем "отображение", переменную x - указателем "кортежпеременных". Чтобы пополнить список нормализаторов приема, используем клавишу "н". Будут последовательно появляться предложения, которые можно выбирать либо отвергать нажатием клавиш Insert, Delete. Важно при этом следить за левой частью меню, размещенного в верхней части экрана. Если там записано "Новый указатель", то речь идет о добавлении к описанию приема каких-либо элементов. Если же появляются слова "Старый указатель", то процедура предлагает убрать или сохранить те элементы приема, которые не могут быть воспроизведены текущей версией генератора приемов. Обычно такие предложения отвергаются, и лучше сразу нажать Esc для обрыва последовательности предложений. Ранее отобранные либо удаленные элементы будут учтены в текущей просматриваемой версии приема, но пока не зарегистрированы в файлах. Чтобы их закрепить, нужно нажать, например, F3 (с перекомпиляцией) либо F4 (без перекомпиляции).

После того, как прием создан, снова запускаем процесс решения и убеждаемся, что ответ получен.

4. Задача на нахождение разбиений имеет единственное условие "разбиение($\{1, 2, 3\}, x$)" и неизвестную x. Чтобы перечислить разбиения, будем использовать рекурсию с отбрасыванием первого элемента перечня. Теорема приема, определяющая данную рекурсию, имеет следующий вид:

$\forall_{abA}(\text{разбиение}(\{;b\}, y) = A(y) \to \text{разбиение}(\{a;b\}, x) \leftrightarrow \exists_y(A(y)\ \&\ (x = \{\{a\}\} \cup y\ \vee\ \exists_z(z \in y\ \&\ x = (y \setminus \{z\}) \cup \{z \cup \{a\}\}))))$.

В исходном условии "разбиение($\{a;b\}, x$)" отбрасывается первый элемент списка a и решается вспомогательная задача с неизвестной y, имеющая условие "разбиение($\{;b\}, y$)". Обращение к ее решению происходит при обработке антецедента теоремы, выделенного указателем "идентификатор". Левая часть антецедента помечена нормализатором "задача(4 тип(описать) полный явное прямойответ упростить цель(неизвестная(x24)))"; ответ задачи присваивается функциональной переменной $A(y)$. Заменяющий терм строится на основе утверждения $A(y)$, означающего, что y - некоторое разбиение подсписка $\{;b\}$. В нем рассматриваются два случая: либо x получено добавлением к y нового одноэлементного класса a, либо получено из y добавлением элемента a к одному из

старых классов. Последнее выражено как результат исключения из y класса z и добавления класса $z \cup \{a\}$. Прием имеет фильтры "урвень(2)", "условие", "тип(описать)", "не(известно(x23))" и указатели "идентификатор(1)", "отображение(x26)", "новаяпеременная(x24)".

После создания приема и проверки того, что задача решена, рекомендуем рассмотреть задачу на перечисление всех разбиений множества $\{1,2,3,4\}$. Хотя ответ и будет получен, он потребует дополнительных упрощений: система не сумеет вычитать друг из друга конечные списки, составленные из конечных списков. Создайте прием проверочного оператора "усмнепринадлежит", который позволил бы упростить данный ответ. Аналогичную работу проделайте для пятиэлементного множества.

5. Чтобы определить значение неизвестной x, достаточно найти, скольким множествам указанного семейства принадлежит какой-либо элемент результирующего множества, например, 1. Поэтому вводим следующую теорему приема:

   $\forall_{abcdxmn}$(слойсемейства$(\lambda_i(a(i), i \in \{1,\ldots,n\}), b, x) = \{c; d\}$ & $m = \sum_{i=1}^{n}$(1 при $c \in a(i)$, иначе 0) $\rightarrow x = m$)

   Прием будет определять дополнительное условие задачи $x = m$, поэтому заголовок его - "выводусловия". Фильтры - "уровень(3)", "тип(описать)", "условие", "не(известно(x23))", "известно(фикс(1 1 1))", "известно(x2)", "известно(x3)". Первый антецедент идентифицируется с рассматриваемым уравнением. Подтерм $\lambda_i(a(i), i \in \{1,\ldots,n\})$ выделен указателем "развертка", т.е. семейство множеств должно быть явно задано термом "набор(...)". Функциональная переменная a выделена указателем "отображение". При идентификации находится набор термов $a(1),\ldots,a(n)$, используемый компилятором для построения других выделенных указателем "развертка" термов с переменной a. Второй антецедент выделен указателем "идентификатор", причем входящая в него конечная сумма выделена указателем "развертка". Она обрабатывается нормализатором "нормплюс"; подвыражение "вариант(...)" обрабатывается нормализатором "нормвариант"; подутверждение $c \in a(i)$ - обрабатывается нормализатором "нормусм". Эти нормализаторы обеспечивают в стандартных случаях непосредственное вычисление количества элементов семейства a, которым принадлежит c. Компилируем прием; вводим условие задачи и убеждаемся, что она системой решается. Однако, если попробовать провести трассировку по шагам решения, то выяснится, что после нахождения условия $x = 2$ проверка равенства для слоя семейства на экран не выводится. Рекомендуется использовать режим трассировки "Ручной выбор входа в подпроцесс" для просмотра того, как будет выполняться проверка. Она оказывается чрезмерно сложной, и попытка добиться существенного ее упрощения оставляется читателю в качестве самостоятельного дополнительного упражнения.

6. Теорема приема имеет вид:

$$\forall_{aBn}(\lambda_i(a(i), i \in \{1,\ldots,n\}) \in \prod_{i=1}^{n} B(i) \leftrightarrow \forall_i(i \in \{1,\ldots,n\} \rightarrow a(i) \in B(i))).$$

   Знак произведения соответствует символу "Прямоепроизведение" во внутренней (текстовой) записи данной теоремы. Его можно вводить нажатием клавиши "Ctr-*". Указатель "развертка(фикс(0 1 1)фикс(0 1 2)фикс(0 2))" определяет

идентификацию выражения $\lambda_i(a(i), i \in \{1, \ldots, n\})$ с конечным набором, идентификацию выражения $\prod_{i=1}^{n} B(i)$ с прямым произведением n множеств, а также построение терма $\forall_i(i \in \{1, \ldots, n\} \to a(i) \in B(i))$ в виде конъюнкции принадлежностей. Чтобы предотвратить вырожденный случай одноэлементного набора, вводим фильтр "меньше(1 x14)".

13.2.9 Создание приемов для простейших свойств функций

1. Ввести приемы, позволяющие решить задачу на описание, в которой требуется привести пример взаимно-однозначного отображения f, определенного на множестве $\{1, 2\}$ и принимающего значения на множестве $\{3, 4\}$.

2. Ввести приемы, позволяющие найти функцию f, сужение которой на множество $\{1, 2\}$ определяется таблицей $\{1 \to 3, 2 \to 4\}$, сужение на множество $\{3, 4\}$ - таблицей $\{3 \to 1, 4 \to 2\}$, причем область определения равна $\{1, 2, 3, 4\}$

3. Ввести приемы для решения задачи на описание, имеющей условия "перестановка(f, $\{1, 2, 3\}$)", $f(1) = 2$, $f(3) = 1$ и неизвестную f.

4. Ввести приемы для решения задачи на описание, имеющей условие "выборка((1, 3, 2, 4, 1, 5), x) = (1, 1, 5)" и неизвестную x.

5. Ввести приемы, позволяющие найти набор "наложение((1, 2, 3), (4, 5), (1, 3, 4))".

6. Ввести приемы, позволяющие решить уравнение "Вставка((1, 2, 3, 4),x,y) = (1, 5, 2, 3, 4)".

7. Ввести прием, позволяющий найти набор "кортежпар((1, 2, 3), (4, 5, 6))".

8. Ввести приемы, позволяющие найти последовательность a, удовлетворяющую условиям: "последовательность(a $\mathbf{N}$)", "подпоследовательность($\lambda_i(i, i -$ натуральное), a)", "подпоследовательность($\lambda_i(i^2, i -$ натуральное), a)".

9. Ввести приемы, позволяющие перейти к табличному заданию функции "индикатор($\{1, 2, 3\}, \{1, 3\}, 1$)".

10. Ввести приемы, позволяющие найти явное задание для функции "функстепень($\lambda_i(i^2, i -$ натуральное), 3)".

Указания

1. Прежде всего, вводим условия задачи на описание: "взаимнооднозначно(f)", "Dom(f) = $\{1, 2\}$", "Val(f) = $\{3, 4\}$". Выбираем в оглавлении целевых установок пункт "Найти пример значений неизвестных" и вводим неизвестную f. Убеждаемся, что система не решает данную задачу.

   Если это не так, т.е. при обучении системы, уже после написания данного раздела, пришлось ввести приемы, обеспечивающие решение, то рекомендуется найти такие приемы путем пошаговой трассировки и временно отключить. Это замечание, не повторяя его каждый раз, относим ко всем приводимым в книге упражнениям. Впрочем, по мере возможности, предоставляемое упражнениями поле для самостоятельного творчества читателя будет сохраняться.

В нашей задаче требуется найти пример взаимно-однозначного отображения одного конечного множества на другое. При этом оба множества заданы путем явного перечисления своих элементов. Последний случай настолько типичен, что для него стоит создать отдельный прием. Оставляя для читателя возможность самостоятельно развить технику установления взаимно-однозначного соответствия между двумя конечными множествами, заданными другими средствами, ограничимся этим приемом. Так как речь идет о подборе примера, теорема приема должна представлять собой кванторную импликацию, консеквентом которой будет конъюнкция реализуемых условий. В нашем случае это условия взаимной однозначности f и равенства областей определения и значений f двум конкретным перечням $\{;a\}$, $\{;b\}$. Антецедентами теоремы приема естественно сделать утверждения $l(a) = n$, $l(b) = n$, указывающие на совпадение длин наборов a, b и вводящие вспомогательное обозначение n, а также утверждения о попарном различии элементов списков a, b. Наконец, в антецеденты заносим равенство, определяющее f с помощью таблицы, где элементу $a(i)$ сопоставлен элемент $b(i)$. Окончательно, теорема приема принимает следующий вид:

$\forall_{abfn}(l(a) = n \ \& \ l(b) = n \ \&$ взаимнооднозначно$(a) \ \&$ взаимнооднозначно(b)
$\& \ f =$ таблица$\{;\lambda_i(a(i) \ \rightarrow \ b(i), i \in \{1, \ldots, n\})\} \rightarrow$ взаимнооднозначно(f)
$\& \ \mathrm{Dom}(f) = \{;a\} \ \& \ \mathrm{Val}(f) = \{;b\})$

Прием имеет заголовок "подборзначений". Указатель "подборзначений(5)" выделяет тот антецедент, который будет заменять конъюнкцию утверждений консеквента. Указатели "идентификатор(1 2)", "блокпроверок(3 4)" задают способ обработки прочих антецедентов. Чтобы переменной n было присвоено численное значение общей длины наборов a, b, выражения $l(a)$, $l(b)$ обрабатываются нормализатором "нормдлинанабора". Указатель "развертка(фикс(5 2 1 1))" необходим, чтобы получить явное представление перечня таблицы через выражение "набор(...)".

Выбираем для приема какой-либо средний уровень срабатывания, например, 4. Прочие фильтры его - "тип(описать)", "условие", "неизвестная(f)", "или(цель(пример)не(цель(полный)))", "заголовок(x1 набор)", "заголовок(x2 набор)". Последние два гарантируют, что значением n будет натуральное число.

После создания приема убеждаемся, что задача решается.

2. Вводим задачу на описание, имеющую условия "сужение(f,$\{1,2\}$) = $\{1 \ \rightarrow \ 3, 2 \ \rightarrow \ 4\}$", "сужение($f$,$\{3,4\}$) = $\{3 \ \rightarrow \ 1, 4 \ \rightarrow \ 2\}$", "Dom($f$) = $\{1,2,3,4\}$". В отличие от предыдущего примера, нам нужно найти все значения неизвестной f, для которых истинны условия. Убеждаемся, что система задачу не решает.

   Естественным приемом для решения уравнений вида "сужение(f, A)= g", где A, g известны, а f - неизвестная, является представление f как доопределения известной функции g на дополнении A до области определения f. Это доопределение можно представить в виде "таблица($\{g, h\}$)"; h - функция, имеющая своей областью определения указанное дополнение. Таким образом, приходим к следующей теореме приема:

   $\forall_{fgA}(A \subseteq \mathrm{Dom}(f) \ \& \ \mathrm{Dom}(g) = A \rightarrow$ сужение$(f, A) = g \leftrightarrow \exists_h(h -$функция $\& \ f =$ таблица$\{g, h\} \ \& \ \mathrm{Dom}(h) = \mathrm{Dom}(f) \setminus A))$

Так как замена эквивалентная, прием имеет заголовок "второйтерм". Чтобы решатель исключил квантор существования, перейдя к вспомогательной неизвестной h, вводим указатель "примечание(серия)". Для уточнения контекста срабатывания вводим фильтры: "уровень(5)"; "тип(описать)"; "условие"; "корень"; "неизвестная(x6)"; "известно(x26)"; "известно(x7)". Чтобы отсечь вырожденный случай, когда области определения функций f, g совпадают, добавляем фильтр "не(равно(x26 терм(область(x6))))". Первый антецедент выделяем указателем "блокпроверок", второй - указателем "идентификатор". Наконец, создаем нормализаторы. В первую очередь, здесь следует обработать нормализатором "нормобласть" выражения для областей определения f, g.

После создания приема проводим трассировку решения задачи и убеждаемся, что ответ получается. В действительности при рассмотрении данного примера обнаружилось отсутствие ряда других мелких, но необходимых приемов. На данном этапе обучения это неудивительно. Указанные приемы были введены в решатель и оставлены в нем, так как могут быть полезны во многих других случаях. Ситуация, когда основной прием, введенный для решения задачи, оказывается недостаточен, и нужно вводить серию простых сопровождающих приемов, достаточно типична. Читатель должен быть к этому готов, и для большей наглядности мы приведем список тех приемов, которые в нашем примере понадобилось создавать дополнительно.

Прежде всего, нужен прием, усматривающий вырожденное сужение:

$\forall_{fA}(A = \mathrm{Dom}(f) \to \text{сужение}(f, A) = f)$

Прием относится к оператору "нормсужение", хотя его можно было бы продублировать, оформив также как прием сканирования задачи.

Далее понадобился прием, определяющий область определения функции, заданной "одноточечной" таблицей:

$\forall_{ab}(\mathrm{Dom}(a \ \to \ b) = \{a\})$

Этот прием отнесен к нормализатору "нормобласть".

Понадобился прием, отбрасывающий при сужении таблицы на некоторое множество те ее элементы, которые выходят за рамки данного множества:

$\forall_{abcA}(\neg(a \in A) \to \text{сужение}(\text{таблица}\{a \ \to \ b; c\}, A) = \text{сужение}(\text{таблица}\{; c\}, A))$

Этот прием нужен в двух вариантах - как прием сканирования задачи, инициирующий обработку сужения, и как прием нормализатора "нормсужение", используемый при рекурсивных обращениях (заменяющий терм сам обрабатывается нормализатором "нормсужение").

Наконец, понадобилось скорректировать веса ранее созданных приемов пакета "нормсужение", чтобы новые приемы этого пакета срабатывали первоочередным образом.

3. Вводим условия задачи на описание: "перестановка(f $\{1,2,3\}$)", "$f(1) = 2$", "$f(3) = 1$". Чтобы решатель мог начать самостоятельные действия по ее решению, нужно перейти к параметрическому представлению f в виде "таблица($\{1 \ \to \ b(1), 2 \ \to \ b(2), 3 \ \to \ b(3)\}$)". Здесь $b(1), b(2), b(3)$ - три новые неизвестные. Теорема приема, обеспечивающего данный переход, имеет вид:

   $\forall_{afn}(\text{перестановка}(f, \{; \lambda_i(a(i), i \in \{1, \ldots, n\})\})$ & взаимнооднозначно$(a) \to \exists_b(f = \text{таблица}\{; \lambda_i(a(i) \ \to \ b(i), i \in \{1, \ldots, n\})\}))$

Прием имеет заголовок "выводусловия". Первый антецедент идентифицируется с условием "перестановка($f, \{1, 2, 3\}$)"; второй - обрабатывается проверочным оператором. Указатель "переменные(x2 x14)" определяет выбор для b набора новых переменных, имеющего ту же длину, что и набор a. Указатель "примечание(серия)" заставит решатель рассматривать новое условие как параметрическое описание для неизвестной f. В результате квантор существования по b будет устранен, и переменные b станут новыми неизвестными. Прочие указатели приема - обычные: "блокпроверок(2)", "отображение(x1)", "развертка(фикс(1 2 1) фикс(0 2 2 1 1))". Фильтры тоже не содержат каких-либо особенностей: "уровень(4)", "тип(описать)", "условие", "неизвестная(x6)", "известно(фикс(1 2))".

После компиляции приема запускаем пошаговую трассировку. Решатель использует равенства для $f(1), f(3)$, чтобы найти значения неизвестных $b(1), b(3)$. Затем он сводит утверждение вида "перестановка(таблица(...)A)" к равенствам для области определения и области значений таблицы. Из этих равенств извлекается значение неизвестной $b(3)$.

4. Вводим условие задачи на описание: "выборка$((1, 3, 2, 4, 1, 5), x) = (1, 1, 5)$". Очевидно, значением неизвестной x должно служить множество элементов набора в правой части уравнения. При этом необходима проверка:

   $\forall_{abx}(\{; b\} \subseteq \{; a\} \rightarrow$ выборка$(a, x) = b \leftrightarrow x = \{; b\}$ & выборка$(a, \{; b\}) = b)$

   В этой теореме антецедент не является логически необходимым; он играет роль дополнительного фильтра. Впрочем, его наличие означает, что решателю будет нужен (но не для рассматриваемой задачи) еще один прием - специально для случая, когда включение нарушается. Второй конъюнктивный член консеквента обеспечивает проверку. Прием имеет заголовок "второйтерм" и фильтры "уровень(2)", "тип(описать)", "условие", "известно(x1)", "известно(x2)", "не(известно(x23))". Указатель "блокпроверок(1)" определяет обработку антецедента проверочным оператором.

5. Вводим условие "наложение$((1, 2, 3), (4, 5), (1, 3, 4))$" задачи на преобразование. Выбираем стандартную целевую установку "упростить в о.д.з.". Чтобы находить наложение двух наборов по заданному списку номеров позиций, которые в результате займет первый набор, создадим новый пакетный нормализатор "нормналожение". Заметим, что хотя третий аргумент операции "наложение(...)" - набор, порядок его элементов несущественен.

   Входим в подраздел символа "наложение" оглавления базы приемов и используем интерфейс ввода нового пакетного нормализатора. Нажимаем "Ctr-п" и выбираем левой кнопкой мыши прямоугольник "Нормализатор". Затем нажимаем "с" и вводим название символа "наложение". По завершении ввода нажимаем Enter дважды: первое нажатие завершает редактирование, второе - инициализирует пакет. Далее будем заполнять пакет приемами. Вычисление результирующего набора определим по рекурсии. Начнем с завершающего шага, когда один из наборов пуст. Создаем в оглавлении после автоматически созданного пункта "Переключатель уровня" новый пункт "Пустой набор". В нем размещаем следующие два приема:

   $\forall_{ab}$(наложение$(a,$ пустослово$, b) = a)$

   $\forall_{a}$(наложение(пустослово$, a,$ пустослово$) = a)$

Как обычно, при инициализации пакетного нормализатора соблюдаем аккуратность после компиляции первого приема: его немедленная перекомпиляция приведет к порче пакета. Для восстановления пакета понадобится удалить ЛОС-программу данного приема, а также приема "Переключатель уровня" (клавиша F7), после чего снова их откомпилировать, начиная с переключателя уровня. Если в нормализаторе откомпилированы хоты бы два приема, допустима перекомпиляция любого из них.

Переходим к рассмотрению шага рекурсии. При вычислении значения "наложение(a b c)" рассматриваем два случая: либо набор c содержит единицу, и тогда в начало результирующего набора помещается первый элемент набора a, либо он не содержит единицы, и тогда берется первый элемент набора b. В обоих случаях перед рекурсивным обращением c корректируется: единица, если она есть, отбрасывается, а все прочие элементы уменьшаются на 1. Таким образом приходим к следующим двум теоремам приемов:

$\forall_{abcden}(1 \in \{;d\}\ \&\ \{;e\} = \{;d\}\backslash\{1\}\ \&\ l(e) = n\ \&\ p = \text{наложение}(b, c, \lambda_i(e(i)-1, i \in \{1, \ldots, n\})) \rightarrow \text{наложение}(\text{префикс}(a, b), c, d) = \text{префикс}(a, p))$

$\forall_{abcden}(\neg(1 \in \{;d\})\ \&\ l(d) = n\ \&\ p = \text{наложение}(a, c, \lambda_i(d(i)-1, i \in \{1, \ldots, n\})) \rightarrow \text{наложение}(a, \text{префикс}(b, c), d) = \text{префикс}(b, p))$

Антецеденты $p = \ldots$ выделены указателем "идентификатор"; они реализуют рекурсивное обращение (правый части обрабатываются нормализатором "нормналожение"). Первый прием имеет фильтры "заголовок(x4 набор)", "заголовок(x5 набор)", второй - фильтр "заголовок(x4 набор)". Они введены, чтобы компилятор мог правильно сформировать выделенные указателем "развертка(...)" описатели $\lambda_i(\ldots)$. Заметим, что в первом приеме введен вспомогательный набор e, полученный отбрасыванием единицы из третьего операнда "наложения", и лишь затем выполняется вычитание единицы из его элементов. Существенной деталью является то, что для проверки вхождения единицы в набор d предпринимается рассмотрение созданного на базе этого набора перечня $\{;d\}$ (множества элементов набора).

Однако, первый прием имеет небольшой изъян - если набор d был одноэлементным, то после отбрасывания из его элементов единицы возникнет пустое множество, и его не удастся идентифицировать как $\{;e\}$: последний терм имеет своим заголовком символ "перечень". Поэтому придется для данного случая создать еще один прием:

$\forall_{ab}(\text{наложение}((a), b, (1)) = \text{префикс}(a, b))$

При вводе одноэлементных наборов (a),(1) могут возникнуть трудности: формульный редактор не поймет, что скобки означают одноэлементные наборы, и проигнорирует их. Потому либо нужно будет использовать специальную клавишу Ctr-н, либо набрать прием текстовым редактором.

После того, как пакет создан, нужно еще ввести прием, обращающийся к нему:

$\forall_{abcd}(a = \text{наложение}(b, c, d) \rightarrow \text{наложение}(b, c, d) = a)$

Прием имеет заголовок "второйтерм". Антецедент выделен указателем "идентификатор", причем правая часть антецедента обрабатывается нормализатором "нормналожение". Фильтр "не(заголовок(x1 наложение))" контролирует невырожденность обращения.

6. Вводим задачу на описание с условием "Вставка$((1,2,3,4),x,y)=(1,5,2,3,4))$"; x,y - неизвестные, причем требуется найти все возможные их значения. Теорема приема должна сравнивать два набора и находить такой номер позиции k, до которой наборы совпадают, а после - элементы первого набора совпадают с элементами второго, имеющими на единицу большие номера. Если при этом на k - й позиции элементы различны, то место вставки определяется однозначно. Таким образом, приходим к следующему приему:

   $\forall_{abkmnxy}(m=n+1\ \&\ k\in\{1,\dots,n\}\ \&\ \neg(a(k)=b(k))\ \&\ \forall_i(i\in\{1,\dots,k-1\}\to a(i)=b(i))\ \&\ \forall_i(i\in\{k,\dots,n\}\to a(i)=b(i+1))\to(\text{Вставка}(\lambda_i(a(i),i\in\{1,\dots,n\}),x,y)=\lambda_j(b(j),j\in\{1,\dots,m\}))\leftrightarrow(x=k\ \&\ y=b(k)))$

   Прежде всего, идентифицируются наборы a, b. Соответствующие им описатели "отображение" в левой части эквивалентсноти выделены указателями "развертка". Сами переменные a,b выделены указателем "отображение". После идентификации наборов определяются натуральные константы m,n. Первый антецедент, выделенный указателем "программа", проверяет, что второй набор на единицу длиннее первого. Второй антецедент тоже выделен указателем "программа" - он перечисляет номера k сравниваемых позиций. Третий антецедент выделен указателем "легковидеть". Так как тип элементов наборов априори неизвестен, для установления различия элементов используем вспомогательную задач на доказательство. Четвертый и пятый антецеденты обеспечивают циклы сравнения наборов до и после позиции k. Они выделены указателем "идентификатор". Прием имеет заголовок "второйтерм". Фильтры возьмем, например, следующие: "уровень(2)", "тип(описать)", "условие".

7. Вводим условие задачи на преобразование "кортежпар$((1,2,3),(4,5,6))$". Очевидно, необходима следующая теорема приема:

   $\forall_{abn}(\text{кортежпар}(\lambda_i(a(i),i\in\{1,\dots,n\}),\lambda_j(b(j),j\in\{1,\dots,n\}))=\lambda_i((a(i),b(i)),i\in\{1,\dots,n\}))$

   Указатель "развертка(фикс(0 1 1)фикс(0 1 2)фикс(0 2))" определяет работу с конечными наборами. Переменные a,b выделены указателем "отображение".

8. Вводим задачу на описание с условиями "последовательность(a,N)", "подпоследовательность$(\lambda_i(i,i-\text{натуральное}),a)$", "подпоследовательность$(\lambda_i(i^2,i-\text{натуральное}),a)$". В ней нужно найти пример значения неизвестной a. Создадим прием, дающий параметрическое описание для последовательности, имеющей заданную подпоследовательность. Простейший способ - разместить элементы этой подпоследовательности на четных позициях, а на нечетных - элементы новой последовательности, играющей роль параметра. Теорема приема имеет вид:

   $\forall_{xaA}(\exists_c(\text{последовательность}(c,A)\ \&\ x=\lambda_j((a(j/2)\text{ при }j-\text{even},\text{иначе }c((j+1)/2)),j-\text{натуральное}))\to\text{последовательность}(x,A)\ \&\ \text{подпоследовательность}(\lambda_i(a(i),i-\text{натуральное}),x))$

   Заголовок приема - "подборзначений"; он реализует попытку неэквивалентной замены условий. Указатель "подборзначений(1)" означает, что роль заменяющего утверждения играет антецедент. Указатель "комментарий(1 серия)" передает комментарий "серия"к заменяющему утверждению во вспомогательную задачу. Таким образом, это утверждение будет рассматриваться как параметрическое описание с параметром c. Функциональная переменная a выделена ука-

зателем "отображение". Прием имеет фильтры "уровень(4)", "тип(описать)", "цель(пример)", "неизвестная(x23)", "известно(фикс(0 2 1))", "контекст(новоеусловие(x2)не(равно(x2 фикс(0 1)))входит(x23 x2))". Последний из фильтров означает, что имеется еще какое-то условие на неизвестную последовательность. Иначе можно было бы просто отождествить ее с известной подпоследовательностью.

После срабатывания введенного приема и исключения неизвестной a, явно выраженной через новую неизвестную последовательность, возникнет задача, имеющая условие:

подпоследовательность($\lambda_i(i^2, i-$натуральное), $\lambda_n((n/2$ при $n-$even, иначе $b((n+1)/2)), n-$натуральное)).

Здесь b - новая неизвестная последовательность. Нам нужен еще один прием, который будет связывать указанную подпоследовательность элементов i^2 с b. Этот переход неэквивалентный, так что прием снова имеет заголовок "подборзнчений". Его теорема имеет вид:

$\forall_{cpqA}$(последовательность(q, A) & подпоследовательность$(c, q) \rightarrow$ подпоследовательность$(c, \lambda_n((p(n)$ при $n-$ even, иначе $q((n+1)/2)), n-$ натуральное)))

Указатель "подборзначений(2)" выделяет заменяющий антецедент. Прочие указатели и фильтры - стандартные. Хотя данный прием и может показаться слишком искусственным, он всего лишь учитывает специальный вид параметрического задания неизвестных последоватеьностей, определяемый предыдущим приемом.

Для завершения решения задачи нам понадобится еще один прием - отождествление неизвестной последовательности x с известной подпоследовательностью, если прочих условий на x нет. Теорема приема такова:

$\forall_{axA}$(последовательность(a, A) & $x = a \rightarrow$ последовательность(x, A) & подпоследоватеьность(a, x))

Заголовок приема - "подборзначений". Указатели "блокпроверок(1)", "подборзначений(2)" задают обработку антецедентов. Фильтр "не(контекст(новоеусловие(x2)не(равно(x2 фикс(0 1)))входит(x23 x2)))" обеспечивает отсутствие прочих условий с неизвестной x. Оставшиеся фильтры - "уровень(2)", "тип(описать)", "цель(пример)", "неизвестная(x23)", "известно(x1)".

9. Вводим задачу на упрощение выражения "индикатор($\{1, 2, 3\}, \{1, 3\}, 1$)". Переход к табличному заданию такого выражения будем считать (по крайней мере, на момент рассмотрения данной задачи) преобразованием общей стандартизации. Возможно, впоследствии возникнут какие-то разумные ограничения этого перехода. Теорема приема имеет вид:

$\forall_{aAnb}$(индикатор$(\{; \lambda_i(a(i), i \in \{1, \ldots, n\})\}, A, b) =$ таблица$\{; \lambda_i(a(i) \rightarrow (b$ при $a(i) \in A,$ иначе $0), i \in \{1, \ldots, n\})\})$

Указатель "развертка(фикс(0 1 1 1)фикс(0 2 1 1))" определяет идентификацию и формирование описателей "отображение" как конечных наборов. Условное выражение в заменяющем терме обрабатывается нормализатором "нормвариант", а его подутверждение $a(i) \in A$ - нормализатором "нормпринадлежит".

10. Вводим задачу на преобразование выражения "функстепень($\lambda_i(i^2, i-$натуральное$), 3)$". Заметим, что "функстепень" вводится и прорисовывается формульным редактором как обычная степень, причем перед началом решения задачи происходит автоматическое преобразование ее в "функстепень". Для рекурсивного определения n - кратного последовательного применения функции к заданному аргументу создаем следующие два приема:

    $\forall_{fgABn}(\forall_x(A(x) \to A(f(x))) \ \& \ (\lambda_x(f(x), A(x)))^{n-1} = \lambda_x(g(x), B(x)) \to (\lambda_x(f(x), A(x)))^n = \lambda_x(f(g(x)), A(x)))$

    $\forall_{fA}((\lambda_x(f(x), A(x)))^1 = \lambda_x(f(x), A(x)))$

    Здесь тоже "функстепень" вводится как обычная степень, а при компиляции приема модифицируется. Первый антецедент первой теоремы приема выделен указателем "следствие". Он проверяет, что область значений рассматриваемой функции содержится в ее области определения, и таким образом решает проблему согласования областей определения при последовательных отображениях. Второй антецедент выделен указателем "идентификатор", а левая часть его обрабатывается вспомогательной задачей на упрощение. Здесь и реализуется шаг рекурсии. Фильтр "натуральное(x14)" ограничивает попытки применения приема случаями, когда величина степени - натуральная константа. Рекомендуем самостоятельно реализовать эту же схему рекурсии в виде приемов нормализатора "нормфункстепень" и добавить прием, обращающийся к данному нормализатору. Нормализатор "нормфункстепень" уже имеет приемы, предназначенные для специальных случаев, так что новые приемы следует вводить на более высоких уровнях, чтобы они не мешали срабатыванию уже введенных.

13.2.10 Создание приемов по комбинаторике

Ввести в задачник решателя перечисляемые ниже задачи по комбинаторике, переведя их на понятный ему логический язык. В тех случаях, когда на задачу выдается отказ, создать недостающие приемы и обеспечить ее решение.

1. Пять учеников следует распределить по трем параллельынм классам. Сколькими способами это можно сделать ?

2. Восемь авторов должны написать книгу из шестнадцати глав. Сколькими способами возможно распределение материала между авторами, если два человека напишут по три главы, четыре - по две и два - по одной главе книги ?

3. Семь яблок и три апельсина надо положить в два пакета так, чтобы в каждом пакете был хотя бы один апельсин и чтобы количество фруктов в них было одинаковым. Сколькими способами это можно сделать ?

4. Садовник должен в течение трех дней посадить 10 деревьев. Сколькими способами он может распределить по дням работу, если будет сажать не менее одного дерева в день ? Рассмотреть два случая, в одном из которых деревья не различаются, а в другом - различаются.

5. Сколькими способами можно выбрать 12 человек из 17, если данные двое человек из этих 17 не могут быть выбраны вместе ?

6. Сколькими способами можно выбрать из чисел от 1 до 100 три числа так, чтобы их сумма делилась на 3 ?

7. n предметов расположены в ряд. Сколькими способами можно выбрать из них три предмета так, чтобы не брать никаких двух соседних элементов ?

8. Сколькими способами 6 человек могут выбрать из 6 пар перчаток по правой и левой перчатке так, чтобы ни один не получил пары ?

9. Сколькими способами можно раздать 27 книг лицам A,B,C так, чтобы A и B вместе получили вдвое больше книг, чем C ?

10. Каких чисел от 1 до 10000000 будет больше: в записи которых встречается единица, или тех, в записи которых ее нет ?

Указания

1. Число способов распределения пяти учеников по трем классам равно числу отображений множества $\{1,\dots,5\}$ (номера учеников) в множество $\{1,\dots,3\}$ (номера классов). Вводим задачу на преобразование с целевой установкой "упростить выражение в области допустимых значений". Условием ее служит выражение $\mathrm{card}(\mathrm{set}_f(\text{Отображение}(f,\{1,\dots,5\},\{1,\dots,3\})))$. Запускаем процесс решения и получаем ответ 243. При трассировке обнаруживаем, что сразу сработал прием, вычисляющий мощность класса отображений одного конечного множества в другое.

2. Задачу можно понимать двояко: если заранее известно, какие авторы сколько должны написать, то ответ будет одним; если это распределение допускается варьировать, то ответ другой. Ограничимся рассмотрением первого случая. Здесь требуется найти число таких отображений множества $\{1,\dots,16\}$ в множество $\{1,\dots,8\}$, что мощности прообразов элементов 1 и 2 равны 3; мощности прообразов элементов 3,4,5,6 равны 2; мощности прообразов элементов 7,8 равны 1. Это дает следующее условие задачи на преобразование:

   $\mathrm{card}(\mathrm{set}_f(\text{Отображение}(f,\{1,\dots,16\},\{1,\dots,8\})\ \&\ \forall_i(i\in\{1,2\}\to\mathrm{card}(\text{слой}(f, i))=3)\ \&\ \forall_i(i\in\{3,\dots,6\}\to\mathrm{card}(\text{слой}(f,i))2)\ \&\ \forall_i(i\in\{7,8\}\to\mathrm{card}(\text{слой}(f, i))=1)))$.

   Решатель выдает ответ 3632488000, что равно $(16!)/(2^6\cdot 3^2)$. В принципе,при решении должна была бы применяться стандартная формула для числа отображений с заданными мощностями прообразов. Соответствующий прием легко найти в разделе "Непосредственное определение мощности". Однако, этот прием не имеет указателя "развертка", позволяющего идентифицировать список мощностей прообразов конкретных элементов. Он срабатывает лишь при условии, что мощности прообразов заданы некоторым общим выражением. При трассировке решения усматривается момент, когда условия на мощности прообразов перечисляются по отдельности для каждого элемента. Однако, ввиду отсутствия указанной версии приема, далее начинается трудоемкий процесс рекурсивного упрощения выражения, с рассмотрением каждого условия на мощность прообраза по отдельности. Рекомендуется самостоятельно добавить такую версию и обеспечить решение задачи с ее применением.

3. Формулировку задачи начнем с того, что обозначим множество апельсинов через A, множество яблок через B, и введем посылки $\mathrm{card}(A) = 3$, $\mathrm{card}(B) = 7$, $\mathrm{непересек}(A, B)$. Так как пакеты не различаются, то нужно найти множество неупорядоченных пар C, D подмножеств объединения $A \cup B$: в один из пакетов кладутся элементы C, в другой - элементы D. Дополнительные условия состоят в равенстве мощностей множеств C, D, в пересечении каждого из них с A, а также в том, что эти множества дополняют друг друга до $A \cup B$. Отсюда получаем условие задачи:

   $\mathrm{card}(\mathrm{set}_M(\exists_{CD}(M = \{C, D\}\ \&\ C \subseteq (A \cup B)\ \&\ \mathrm{card}(C) = \mathrm{card}(D)\ \&\ D = (A \cup B) \setminus C\ \&\ \neg(A \cap C = \emptyset)\ \&\ \neg(A \cap D = \emptyset))))$

   Запускаем решатель и убеждаемся, что он задачу не решает. Выдаваемый ответ имеет вид:

   $\mathrm{card}(\mathrm{set}_M(\exists_C(\neg(A \subseteq C)\ \&\ \neg(\mathrm{непересек}(A, C))\ \&\ M = \{C, (A \cup B) \setminus C\}\ \&\ \mathrm{card}((A \cup B) \setminus C) = \mathrm{card}(C)\ \&\ C - \mathrm{set}))).$

   Так как не был выполнен переход от рассмотрения неупорядоченной пары M к рассмотрению упорядоченных пар, то естественно начать с создания приема, выполняющего этот переход. Выше был рассмотрен прием данного типа, у которого неупорядоченная пара относится к двум переменным связывающей приставки квантора существования. Здесь он неприменим. Будем создавать его обобщение на случай пар общего вида $\{p(y), q(y)\}$, где y - связывающая приставка квантора существования. Чтобы можно было перейти к неупорядоченным парам, следует, во-первых, убедиться, что $p(y)$ никогда не совпадает с $q(y)$, а также в том, что при перестановке $p(y)$ и $q(y)$ снова возникает допустимая пара, т.е. существует z, такое, что $p(z) = q(y)$ и $q(z) = p(y)$. Отсюда получаем теорему приема:

   $\forall_{pqQ}(\neg(p(y) = q(y))\ \&\ \exists_z(p(z) = q(y)\ \&\ q(z) = p(y)\ \&\ Q(z)) \to \mathrm{card}(\mathrm{set}_x(\exists_y(x = \{p(y), q(y)\}\ \&\ Q(y)))) = (\mathrm{card}(\mathrm{set}_{uv}(\exists_y(u = p(y)\ \&\ v = q(y)\ \&\ Q(y))))/2))$

   Оба антецедента выделены указателями "следствие"; чтобы соответствующие вспомогательные задачи на доказательство получили дополнительные посылки $Q(y)$, введены указатели "занесениепосылки(1 значение(x41 x24))", "занесениепосылки(2 значение(x41 x24))". Прочие указатели приема - "отображение(x15 x16 x41)", "кортежпеременных(x24)", "внешнийквантор(фикс(0 1 1 2))", "список(фикс(0 1 1 2 2 1 2 1))". Компилируем прием и запускаем решение задачи. Созданного приема оказалось достаточно - решатель выдает ответ 105.

4. Ограничимся случаем, когда деревья не различаются. Тогда нужно найти число троек натуральных чисел x, y, z, составляющих в сумме 10: x деревьев будет посажено в первый день, y - во второй, и z - в третий. Условие задачи имеет следующий вид:

   $\mathrm{card}(\mathrm{set}_{xyz}(x{-}\text{натуральное}\ \&\ y{-}\text{натуральное}\ \&\ z{-}\text{натуральное}\ \&\ x{+}y{+}z = 10))$

   Решатель выдает ответ 36, используя прием для числа сочетаний с повторениями.

5. Обозначим через A множество из 17 человек; B - его подмножество из двух заданных человек. Тогда требуется определить число подмножеств C, не включающих B:

$\mathrm{card}(\mathrm{set}_C(C - \mathrm{set}\ \&\ C \subseteq A\ \&\ \mathrm{card}C = 12\ \&\ \neg(B \subseteq C)))$

Вводим задачу и запускаем решение. Получаем ответ 3185.

6. Требуется найти число трехэлементных подмножеств A множества $\{1, \ldots, 100\}$, сумма которых делится на 3:

$$\mathrm{card}(\mathrm{set}_A(A - \mathrm{set}\ \&\ A \subseteq \{1, \ldots, 100\}\ \&\ \mathrm{card}A = 3\ \&\ 3|\sum_{i, i \in A} i))$$

На эту задачу решатель не находит ответа, и приходится создавать новые приемы. В учебнике по комбинаторике предлагается следующий план решения: тройка чисел представляется как объединение трех подмножеств, соответствующих различным вычетам по модулю 3, и анализируются возможные случаи для мощностей этих подмножеств. Попробуем реализовать его с помощью цепочки приемов. Прежде всего, нужно как-то объяснить переход к рассмотрению подмножеств, определяемых значениями вычета по модулю 3. Этот переход подсказывается тем, что в задаче имеется условие делимости конечной суммы на заданное натуральное число (3). Такую сумму можно заменить суммой произведений значений вычетов j на количества слагаемых, имеющих заданное значение вычета. Теорема приема такова:

$$\forall_{Atm}(m \mid \sum_{i, A(i)} t(i) \leftrightarrow m \mid \sum_{j=1}^{m-1}(j\mathrm{card}(\mathrm{set}_i(A(i)\ \&\ t(i)(\mathrm{mod}\ m) = j))))$$

Вводя фильтр "контекст(подчинено(теквхожд x2)символ(x2 мощность))", ограничиваем действие приема случаями, когда условие делимости расположено внутри выражения для мощности множества. Фильтр "натуральное(x13)" определяет идентификацию m с натуральной константой.

После применения данного приема возникают подвыражения $\mathrm{set}_i(i(\mathrm{mod}3) = 2\ \&\ i \in A)$, $\mathrm{set}_i(i(\mathrm{mod}3) = 1\ \&\ i \in A)$. Так как внешний описатель "класс" относится к перечисляемым множествам A, то удобно вынести из этих подвыражений множество A наружу, представив их как пересечения A с некоторыми не варьируемыми множествами. Для данной цели создаем следующий прием:

$\forall_{ABP}(A \subseteq B \rightarrow \mathrm{set}_i(i \in A\ \&\ P(i)) = A \cap \mathrm{set}_i(i \in B\ \&\ P(i)))$

Включение $A \subseteq B$ непосредственно встречается в контексте замены. Фильтр "контекст(подчинено(теквхожд x2)символ(x2 класс)входит(x26 связприставка(x2)) не(пересекаются(параметры(x27)связприставка(x2))))" указывает на то, что A есть варьируемая переменная внешнего описателя "класс", в то время как выражение B не имеет варьируемых переменных этого описателя.

После вынесения множества A наружу срабатывает уже имеющийся прием, представляющий это множество в виде $a \cup b$, где a - подмножество $\mathrm{set}_i(i(\mathrm{mod}3) - 2 = 0\ \&\ i \in \{1, \ldots, 100\})$; b - подмножество дополнения последнего множества до $\{1, \ldots, 100\}$. Данный прием был приведен выше в подразделе "Параметрическое описание класса" раздела "Класс". Далее, после ряда несложных переходов, условие задачи приобретает следующий вид:

$\mathrm{card}(\mathrm{set}_{ab}(-3+\mathrm{card}a+\mathrm{card}b=0\ \&\ a-\mathrm{set}\ \&\ b-\mathrm{set}\ \&\ a\subseteq\mathrm{set}_i(-2+i(\mathrm{mod}3)=0\ \&\ i\in\{1,\ldots,100\})\ \&\ b\subseteq\mathrm{set}_i(\neg(-2+i(\mathrm{mod}3)=0)\ \&\ i\in\{1,\ldots,100\})\ \&\ 3|(2\mathrm{card}a+\mathrm{card}(b\cap\mathrm{set}_i(-1+i(\mathrm{mod}3)=0\ \&\ i\in\{1,\ldots,100\})))))$

Снова срабатывает прием, подразбивающий b на два подмножества c и d,и реализуется цепочка упрощающих преобразований. В итоге имеем условие следующего вида:

$\mathrm{card}(\mathrm{set}_{cda}(-3+\mathrm{card}a+\mathrm{card}c+\mathrm{card}d=0\ \&\ a-\mathrm{set}\ \&\ c-\mathrm{set}\ \&\ d-\mathrm{set}\ \&\ a\subseteq\mathrm{set}_i(-2+i(\mathrm{mod}3)=0\ \&\ i\in\{1,\ldots,100\})\ \&\ c\subseteq\mathrm{set}_i(-1+i(\mathrm{mod}3)=0\ \&\ i\in\{1,\ldots,100\})\ \&\ d\subseteq\mathrm{set}_i(\neg(-1+i(\mathrm{mod}3)=0)\ \&\ \neg(-2+i(\mathrm{mod}3)=0)\ \&\ i\in\{1,\ldots,100\})\ \&\ 3|(2\mathrm{card}a+\mathrm{card}c)))$

Здесь возникает новая для решателя ситуация: нужно усмотреть, что числа, у которых значения вычета по модулю 3 отличны от 1 и 2, делятся на 3. Это усмотрение реализуем в два шага. Сначала обеспечиваем группировку отрицаний равенства вычета заданным значениям в виде отрицания принадлежности вычета. Инициализацию такой группировки выполняет следующий прием:

$\forall_{abckmn}(m=-a\ \&\ n=-c\ \&\ m\in\{0,\ldots,k-1\}\ \&\ n\in\{0,\ldots,k-1\}\to\neg(a+b(\mathrm{mod}k)=0)\ \&\ \neg(c+b(\mathrm{mod}k)=0)\leftrightarrow\neg(b(\mathrm{mod}k)\in\{m,n\}))$

В нашем случае нет необходимости создавать еще один прием, обеспечивающий слияние условий непринадлежности вычета и отрицания его равенства. Рекомендуется создать его самостоятельно. Указанный выше прием снабжаем фильтром "конец(контекст(подчинено(теквхожд x4)символ(x4 класс)пересекаются(x2 связприставка(x4))))", ограничиваясь пока лишь задачами по комбинаторике. После того, как отрицания условий равенства модуля преобразованы в условие непринадлежности модуля, оказывается необходим прием, усматривающий, что остается единственное возможное значение модуля. Вводим этот прием:

$\forall_{abcn}(l(b)=n-1\ \&\ \{0,\ldots,n-1\}\setminus\{;b\}=\{c\}\to\neg(a(\mathrm{mod}n)\in\{;b\})\leftrightarrow a(\mathrm{mod}n)=c)$

После срабатывания приема получаем такое условие задачи:

$\mathrm{card}(\mathrm{set}_{cda}(-3+\mathrm{card}a+\mathrm{card}c+\mathrm{card}d=0\ \&\ a-\mathrm{set}\ \&\ c-\mathrm{set}\ \&\ d-\mathrm{set}\ \&\ a\subseteq\mathrm{set}_i(-2+i(\mathrm{mod}3)=0\ \&\ i\in\{1,\ldots,100\})\ \&\ c\subseteq\mathrm{set}_i(-1+i(\mathrm{mod}3)=0\ \&\ i\in\{1,\ldots,100\})\ \&\ d\subseteq\mathrm{set}_i(i(\mathrm{mod}3)=0\ \&\ i\in\{1,\ldots,100\})\ \&\ 3|(2\mathrm{card}a+\mathrm{card}c)))$

Так как мощность каждого из множеств a,b,c не превосходит 3, причем фиксация этого значения позволяет декомпозировать условие под описателем на независимые подусловия для a,b,c, то нужен прием, разбирающий случаи для значения мощности. Отнесем его к какому-либо одному из параметров a,b,c. Чтобы избежать усложнения приема обращением к пакету, определяющему верхнюю оценку мощности множества, ограничимся ситуацией нашей задачи, где такая оценка извлекается из равенства суммы мощностей тройке. Получаем следующую теорему приема:

$\forall_{APmn}(\text{конечное}(A)\ \&\ 0\le m(y)\to\mathrm{card}(\mathrm{set}_{xy}(x-\mathrm{set}\ \&\ x\subseteq A\ \&\ \mathrm{card}x+m(y)-n=0\ \&\ P(\mathrm{card}x,y)))=\sum_{i=0}^{n}(C^i_{\mathrm{card}A}\mathrm{card}(\mathrm{set}_y(i+m(y)-n=0\ \&\ P(i,y)))))$

Указатель "развертка(фикс(0 2))" определяет преобразование конечной суммы в обычную. Чтобы выполнить идентификацию выражения $P(\mathrm{card}x,y)$, вводим

указатель "новаргумент(х40 х23 фикс)". Необходим также указатель "кортеж-переменных(х24)". Выражения для мощности под суммой обрабатываем вспомогательными задачами на преобразование. В нашей задаче это обеспечит рекурсивное срабатывание приема для разбора случаев по одному из оставшихся параметров a, b, c. Применение приема приводит к выдаче ответа: 53922.

7. Можно рассматривать задачу как нахождение числа троек целых чисел i, j, k, удовлетворяющих неравенствам $1 \le i, i < j-1, j < k-1, k \le n$. Тогда ее условие имеет следующий вид:

   $\mathrm{card}(\mathrm{set}_{ijk}(1 \le i \,\&\, i < j{-}1 \,\&\, j < k{-}1 \,\&\, k \le n \,\&\, i{-}\text{целое} \,\&\, j{-}\text{целое} \,\&\, k{-}\text{целое}))$

   В посылки задачи заносим утверждение n – натуральное. Решатель выдает в качестве ответа сумму всех значений $(j-2)$ по множеству пар j, k, удовлетворяющих условиям $j \in \{3, \ldots, k-2\}, k$ – целое, $5 \le k, k \le n$. Это подсказывает необходимость создания приема, преобразующего двойную сумму в повторную. Теорема приема, например, может иметь следующий вид:

   $$\forall_{PQtm}(m(j) = \sum_{i,P(i,j)} t(i,j) \to \sum_{i,j,P(i,j)\ \&\ Q(j)} t(i,j) = \sum_{j,Q(j)} m(j))$$

   Указатель "перечень(х41 не(входит(х9 х41)))" определяет идентификацию утвер ждения $Q(j)$ со всеми конъюнктивными членами, не содержащими параметра i. Антецедент выполняет суммирование выражений $t(i,j)$ по i. Для этого используется вспомогательная задача на преобразование, посылки которой пополнены конъюнктивными членами утверждения $Q(j)$, т.е. вводится нормализатор "задача(4 упростить) посылки(значение(х41 х10))". Указатель "элемент(х9)" разрешает идентифицировать переменную i либо с первым, либо со вторым параметрами суммирования. Наконец, фильтр "не(входит(суммавсех значение(х13 х10)))" проверяет успех попытки суммирования по i. После создания приема решатель выдает ответ в следующем виде: "Если $5 \le n$, то $(n-2)(n-3)(n-4)/6$, иначе 0".

8. Выбор левых перчаток представляет собой перестановку множества $\{1, \ldots, 6\}$ номеров пар: на i-м месте расположен номер пары, из которой берет перчатку i-й человек. Аналогичную перестановку имеем для правых перчаток. Соответственно, формулируем условие задачи:

   $\mathrm{card}(\mathrm{set}_{fg}(\text{перестановка}(f, \{1, \ldots, 6\}) \,\&\, \text{перестановка}(g, \{1, \ldots, 6\}) \,\&\, \forall_i(i \in \{1, \ldots, 6\} \to \neg(f(i) = g(i)))))$

   Запускаем процесс решения и обнаруживаем, что система "вязнет" в преобразованиях, не доходя до ответа. Входим в трассировку решения (клавиша "Щ"). Решатель сначала избавляется от квантора общности, заменяя его на шесть отрицаний равенства $\neg(f(i) = g(i))$. Затем начинаются попытки применить прием, представляющий искомую мощность как разность мощности множества с отброшенным отрицанием некоторого равенства и множества, где оно заменено на равенство. Таким образом удается дойти до задачи на вычисление мощности множества $\mathrm{set}_{fg}(f(6) = g(6) \,\&\, \text{перестановка}(f, \{1, \ldots, 6\}) \,\&\, \text{перестановка}(g, \{1, \ldots, 6\}))$. Эту мощность решатель не вычисляет. Так как значение перестановки

f в точке 6 фиксировано, ее можно свести к перестановке меньшего множества. Соответствующий прием имеет вид:

$\forall_{iAPt}(\text{card}(\text{set}_{fg}(\text{перестановка}(f, A(g)) \ \& \ f(i) = t(g) \ \& \ P(f,g))) = \text{card}(\text{set}_{fg}(\text{перестановка}(f, A(g) \setminus \{t(g)\}) \ \& \ P(\text{Вставка}(f, i, t(g)), g))))$

После срабатывания данного приема возникает выражение $\text{card}(\text{set}_{fg}(\text{перестановка}(f, \{1, \ldots, 6\} \setminus \{g(6)\}) \ \& \ \text{перестановка}(g, \{1, \ldots, 6\})))$. Связь переменных f, g пока не исчезла - она сохраняется в выражении для множества значений f. Однако, мощность этого множества не зависит от g. Чтобы выполнить декомпозицию, вводим такой прием:

$\forall_{APn}(\text{card}A(g) = n \to \text{card}(\text{set}_{fg}(\text{перестановка}(f, A(g)) \ \& \ P(g))) = n! \cdot \text{card}(\text{set}_g(P(g))))$

Антецедент выделен указателем "идентификатор"; для вычисления мощности $A(g)$ используется вспомогательная задача. Фильтр "не(входит(x7 x14))" проверяет, что найденная мощность от g не зависит. Добавленных приемов уже достаточно, и после трудоемких вычислений решатель выдает ответ 190800. Заметим, что фактически вычисления реализуют примитивную версию принципа "включения и исключения", сводящуюся к многократному представлению множеств $\text{set}_x(P(x) \ \& \ \neg(Q(x)))$ в виде разности $\text{set}_x P(x)$ и $\text{set}_x(P(x) \ \& \ Q(x))$. Реализация в решателе обычной версии данного принципа предоставляется читателю.

9. Распеределение книг между A, B, C определяется функцией, заданной на множестве $\{1, \ldots, 27\}$ и принимающей значения $\{1, 2, 3\}$. Поэтому условие задачи имеет следующий вид:

   $\text{card}(\text{set}_f(\text{Отображение}(f, \{1, \ldots, 27\}, \{1, \ldots, 3\}) \ \& \ 2\text{card}(\text{слой}(f, 3)) = \text{card}(\text{прообраз}(f, \{1, 2\}))))$.

   При запуске решателя выясняется, что происходит лишь тривиальная арифметическая стандартизация. Выражения для мощностей прообраза элемента 3 и прообраза множества $\{1, 2\}$ сохраняются неизмененными. Так как эти прообразы дополняют друг друга до области определения отображения, то можно ввести прием, выражающий один из них через другой:

   $\forall_{ABfc}(\text{Отображение}(f, C, A) \ \& \ B = A \setminus \{c\} \to \text{прообраз}(f, B) = C \setminus \text{слой}(f, c))$

   Инициализация приема происходит при усмотрении выражения "прообраз(f, B)". Указатель "контекст(позиция(x4 корень)вид(x4 слой(x6 x3)))" определяет следующим шагом идентификации усмотрение в текущем терме задачи подвыражения "слой(f, c)". Второй антецедент, выделенный указателем "идентификатор", проверяет, что множество B есть разность области значений A и одноэлементного множества $\{c\}$. После создания приема решатель определяет мощность прообраза элемента 3, и далее доводит задачу до ответа.

10. Множество чисел в указанном интервале, десятичная запись которых не содержит единицы, равномощно множеству отображений из $\{1, \ldots, 7\}$ в множество цифр $\{0, \ldots, 9\} \setminus \{1\}$, за исключением тождественно нулевого отображения. Поэтому условие задачи можно задать, например, следующим образом:

$$\text{card}(\text{set}_f(\text{Отображение}(f, \{1, \ldots, 7\}, \{0, \ldots, 9\} \setminus \{1\}))) - 1 < \frac{1}{2} \cdot 10000000$$

Тип задачи - "описать"; целевая установка - "проверить истинность утверждения". Ответ на задачу находится решателем сразу.

13.2.11 Создание приемов на числовые множества

Создать приемы, необходимые для решения следующих задач:

1. Упростить выражение $\mathbf{Q} \cap \mathbf{Z}$.
2. При заданном вещественном a найти все такие множества x, для которых a является верхней гранью.
3. Вычислить точную нижнюю грань множества $\text{set}_x(\exists_y(x = \sin(y)\ \&\ y - \text{число}))$.
4. Найти точную верхнюю грань множества $\text{set}_m(\exists_n(n - \text{целое}\ \&\ m = 1 + n - 2n^2))$. Найти наибольший элемент этого множества.
5. Найти все такие вещественные x, для которых множество $\text{set}_n(\exists_m(m - \text{натуральное}\ \&\ n = xm))$ ограничено снизу.
6. Найти внутренность круга $\text{set}_{xy}(x^2 + y^2 \leq 1)$.
7. Найти границу того же круга.
8. Найти замыкание множества $\text{set}_x(\exists_n(n - \text{натуральное}\ \&\ x = 1/n))$.
9. Найти все предельные точки того же множества.
10. Создать проверочный оператор "усмОбласть" для усмотрения связных открытых множеств.

Указания

1. Вводим задачу на преобразование, имеющую условие $\mathbf{Q} \cap \mathbf{Z}$, и запускаем ее решение. Обнаруживаем, что выражение не упрощается. Мы знаем, что имеется прием упрощения пересечения двух множеств, одно из которых включается в другое. Раз он не применяется, значит, отсутствуют средства проверки включения множества целых чисел в множество рациональных чисел. Находим проверочный оператор "усмсодержится". В нем не оказывается никаких приемов для усмотрения подмножеств множества рациональных чисел. Поэтому можно ввести несколько таких приемов, хотя бы для основных случаев - множеств натуральных, целых неотрицательных и целых чисел. Для нашего примера создаем прием с теоремой $\mathbf{Z} \subseteq \mathbf{Q}$ и заголовком "спуск(усмсодержится)". Единственный его фильтр - "уровень(1)". Рекомендуется самостоятельно создать несколько более общие приемы - например, для усмотрения включения в множество рациональных чисел множеств, заданных описателями вида $\text{set}_x(x - P\ \&\ Q(x))$, где P - один из символов "натуральное", "целое", "рациональное".

2. Вводим задачу на описание, имеющую посылку "a – число" и условия "верхняяграань(a,x)", "$x \subseteq \mathbf{R}$". Неизвестной служит переменная x. Эта задача системой не решается. Находим подраздел базы приемов, относящийся к верхним граням числовых множеств. В нем нет никаких приемов, позволяющих разрешать предикат "верхняяграань(a,x)" относительно x. Вводим для этой цели следующий прием: $\forall_{ax}(a-$число $\&\ x \subseteq \mathbf{R} \to$ верхняяграань$(a,x) \leftrightarrow x \subseteq (-\infty, a])$. Напоминаем, что круглая и квадратная скобки при задании числового промежутка вводятся нажатиями клавиш "Ctrl- скобка". Фильтры приема задаем, например, такие: "тип(описать)", "условие", "уровень(2)", не(известно(x23))", "известно(x1)". Оба антецедента выделяем указателями "блокпроверок".

3. Вводим задачу на упрощение выражения $\inf(\mathrm{set}_x(\exists_y(x = \sin(y)\ \&\ y -$ число$)))$ и запускаем решатель. Он никак не изменяет данное выражение. Переходим в раздел базы приемов, относящийся к символу "инф", и анализируем имеющиеся приемы. Оказывается, что подходящих приемов для задач рассматриваемого типа там нет. Имеется несколько возможностей для восполнения пробела. Например, можно было бы создать прием, переформулирующий задачу в терминах образов числовых множеств и воспользоваться приемами нормализатора "норминф". Рассмотрим другой подход - создадим прием, предпринимающий попытку разрешить подкванторное условие относительно связанной переменной квантора существования. Такое разрешение сделает вероятным последующее исключение квантора. Теорема приема имеет вид: $\forall_{PQ}(P(x,y) = Q(x,y) \to \inf(\mathrm{set}_x(\exists_y(P(x,y)))) = \inf(\mathrm{set}_x(\exists_y(Q(x,y)))))$. Указатель "отображение(x40 x41)" делает переменные P, Q функциональными. Первый антецедент, выделенный указателем "идентификатор(1)", обращается к вспомогательной задаче на разрешение условия $P(x,y)$ относительно переменной y. Его левая часть обрабатывается нормализатором "задача(5 тип(описать)полный явное прямойответ упростить цель(неизвестная(x24)))". Уровень срабатывания приема выбираем равным 4, чтобы в простых случаях успевали сработать другие приемы определения точной нижней грани. После компиляции приема решатель выдает ответ "-1".

4. Вводим задачу на упрощение выражения $\sup(\mathrm{set}_m(\exists_n(n -$ целое $\&\ m = 1 + n - 2n^2)))$. Решатель не получает требуемый ответ, даже если создать прием для sup, аналогичный приему для inf из предыдущей задачи. Переходим в раздел базы приемов, соответствующий символу "суп". В нем находим подраздел "числовая последовательность". Выбираем пункт "анализ локальных максимумов и верхнего предела". Здесь имеется прием, определяющий точную верхнюю грань числовой последовательности путем поиска точек локального максимума и вычисления верхнего предела последовательности. Однако, в нашей задаче параметр n не натуральный, а целочисленный. Поэтому придется создать альтернативную версию найденного приема, подходящую для целочисленного параметра. Чтобы ускорить процесс, скопируем сначала старую версию (Ctr-д и F4). Выделим подутверждение теоремы "$n-$натуральное", нажмем клавишу "Ф" и введем альтернативную версию "$n-$целое". Повторим это для каждого из двух вхождений в теорему. Затем выделим подутверждение "$n = 1$" и нажмем "Ctrl-Del" для его удаления. Так же удалим "$2 \le n$". Чтобы рассмотреть верхний предел в минус-бесконечности, добавим антецедент $q = \overline{\lim}_{n\to\infty}\{f(-n)\}$. Этого можно добиться, выделив последний антецедент старой версии, нажав "В" и введя формульным редактором указанный дополнительный антецедент. Нако-

нец, заменим в старой версии теоремы подвыражение $\{p\}$ на $\{p, q\}$. Используя текстовый редактор, пополняем кванторную приставку символом q (x16). Далее корректируем указатель "идентификатор(1 2)", заменяя его на "идентификатор(1 2 3)", и обрабатываем нормализатором "задача(5 упростить)" правую часть равенства для q. Компилируем прием и запускаем решатель. Теперь получается ответ 1.

5. Вводим задачу на описание, имеющую условие "огрснизу(set$_n$($\exists_m$(m − натуральное & $n = mx$)))" и единственную неизвестную x. Запускаем ее решение и обнаруживаем, что решатель "зависает" - не выдает ни ответа, ни отказа. Чтобы выяснить причину, нажимаем "Break", затем - пробел (для выхода на уровень трассировки по срабатываниям приемов), и "Enter" - для начала такой трассировки. Нажимая далее многократно "Enter", легко выявляем зацикливание: сначала происходит исключение квантора существования путем явного разрешения подкванторного условия относительно m, а затем снова предпринимается ввод вспомогательного целочисленного параметра m вместе с квантором существования. Чтобы устранить зацикливание, нужно заблокировать хотя бы одно их этих действий. Видимо, естественно отказаться от попытки исключения только что введенного параметра m. Чтобы найти способ заблокировать эту попытку, переходим к просмотру ЛОС-программы приема "Разрешение подкванторного условия относительно переменных кванторной приставки". Для этого в момент срабатывания приема нажимаем "End", из главного меню входим в оглавление программ, и далее нажимаем "курсор вправо". Анализируем начало программы приема, где расположены различные фильтры, блокирующие его срабатывание. В частности, обнаруживаем фильтр "коммент(x1 связприставка подтерм(x2))", который подсказывает способ блокировки: ввод комментария (связприставка A) для рассматриваемого квантора существования A. Можно просмотреть информацию о символе "связприставка" и узнать, что такой комментарий свидетельствует о ранее предпринимавшейся попытке разрешения A. Далее переходим к оглавлению приемов, и в разделе ("Описатель КЛАСС", "Параметрическое описание класса", "Ввод вспомогательного целочисленного параметра") находим упомянутый выше прием, вводящий целочисленный параметр m. Добавляем к нему указатель "замечание(связприставка фикс(0 2 2))", перекомпилируем (F3) и повторно запускаем решение. Теперь зацикливание пропадает, но ответ не находится.

   Очевидно, нужно создать прием, выполняющий расшифровку условия ограниченности снизу множества, заданного описателем "класс". Вводим прием по следующей теореме:

   $\forall_P$(огрснизу(set$_x$($P(x)$)) $\leftrightarrow \exists_y$($y$ − число & $\forall_x$($P(x) \rightarrow y < x$)))

   Сопровождаем прием фильтрами "уровень(2)", "условие", "тип(описать)", "корень", "не(известно(корень))". Нажимаем "н", и получаем следующие подсказанные генератором приемов указатели: "кортежпеременных(x23)", "отображение(x40)". Компилируем прием, запускаем решение задачи и получаем ответ "x − число, $0 \leq x$".

6. Вводим задачу на преобразование выражения "внутренность(set$_{xy}$($x^2 + y^2 \leq 1$))". Запускаем решение, и обнаруживаем, что утверждение под описателем "класс" разрешено относительно переменных связывающей приставки явным образом, однако внутренность не найдена. Переходим в раздел базы приемов,

связанный с символом "внутренность", для рассмотрения уже введенных ранее приемов. В нормализаторе "нормвнутренность" обнаруживаем прием для определения внутренности множества точек плоскости, заданного системой неравенств, явно разрешенной относительно координат. Берем этот прием (выбираем из нескольких представленных в разделе подходящий для нашего случая, т.е. последний) и копируем его нажатием "Ctr-д". Заменяем заголовок на "второйтерм" и добавляем фильтр "уровень(2)". Компилируем прием и запускаем решатель. Обнаруживаем, что после применения приема начинается цикл совершенно ненужных попыток явно разрешить утверждение под описателем "класс". Анализируя прием, предпринимающий такую попытку, замечаем в нем указатель "ключ(класс теквхожд)", блокирующий срабатывание при наличии комментария (класс A), где A - описатель "класс". Поэтому вводим в нашем новом приеме указатель "замечание(класс фикс(0 2))". Запускаем решатель и обнаруживаем, что блокировка не состоялась. Изучаем причину этого с помощью отладчика. Оказывается, что наш прием применяет к заменяющему терму, внутри оператора "заменавхождения", простейшее переупорядочение операндов, а в комментарии (класс ...) этого переупорядочения нет. В результате при последующий преобразованиях комментарий (класс ...) не корректируется синхронно с изменениями под описателем, и к моменту попытки разрешения наблюдается полное рассогласование между текущим выражением и блокирующим комментарием. Чтобы исправить рассогласование, обрабатываем в новом приеме заменяющий терм нормализатором "стандупорядочение". Компилируем прием, запускаем решатель и получаем ответ.

Рекомендуется самостоятельно создать такую версию решения данной задачи, при которой ответ не был явно разрешен относительно переменных x, y.

7. Пример аналогичен предыдущему. Нужно использовать копию уже имеющегося в нормализаторе "нормграница" приема, преобразовав ее в прием сканирования задачи. Указатели "замечание(класс ...)" при этом относятся к каждому из четырех классов заменяющего выражения, причем все они должны быть обработаны нормализатором "стандупорядочение".

8. Чтобы решить эту задачу, проще всего воспользоваться аппаратом нахождения множества всех частичных пределов последовательности, представленным (хотя и минимальным образом) в разделе базы приемов ("Математический анализ", "Предел", "Нахождение частичных пределов"). Собственно говоря, нет необходимости знакомиться с данным аппаратом - достаточно лишь знать о его существовании. Утверждение о том, что число a является частичным пределом последовательности f, записывается в решателе как "частичныйпредел(a f)". Вводим новый прием, позволяющий находить замыкание множества членов числовой последовательности путем добавления к нему всех конечных частичных пределов:

$\forall_{fPx}((\text{частичныйпредел}(x, \lambda_n(f(n), n - \text{натуральное})) \ \& \ x - \text{число}) = P(x) \rightarrow$ замыкание($\text{set}_y(\exists_n(n - \text{натуральное} \ \& \ y = f(n)))) = \text{set}_y(\exists_n(n - \text{натуральное} \ \& \ y = f(n))) \cup \text{set}_y(P(y)))$

Единственный антецедент выделен указателем "идентификатор". Он обращается к вспомогательной задаче на описание для нахождения утверждения $P(x)$,

определяющего искомые конечные частичные пределы x. Левая часть антецедента сопровождается нормализатором "задача(5 тип(описать) полный явное прямойответ цель(неизвестная(x23)))". Указатель "новаяпеременная(x23)" вводит вспомогательную переменную x.

9. Пример аналогичен предыдущему: снова используется описание множества частичных пределов последовательности.

10. Входим в раздел "Область" оглавления базы приемов и нажимаем "Ctrl-п". Вводим название оператор "усмОбласть" и выбираем пункт "Проверочный оператор". Нажимаем левую кнопку мыши в окне "Вид проверяемого утверждения" и набираем текстовым редактором "Область(x1)". После нажатия "Enter" входим в окно "Упорядочение входных переменных" и вводим формульным редактором "a". Число уровней срабатывания полагаем равным 3. Наконец, выбираем окно "Ввести". Рекомендуется самостоятельно создать приемы для усмотрения одномерной области (открытого промежутка) и двумерной области вида $\mathrm{set}_{xy}(a < x \;\&\; x < b \;\&\; f(x) < y \;\&\; y < g(x))$. Затем создать прием сканирования задачи, обращающийся к проверочному оператору "усмОбласть", ввести тестовую задачу на доказательство утверждения вида "Область(A)", и проверить правильность работы созданных приемов.

13.2.12 Упражнения по приемам элементарной алгебры, связанным с арифметическими операциями

Понимание компонент описания приема на ГЕНОЛО́ГЕ

Естественным способом ознакомится с техникой программирования на ГЕНОЛОГе является анализ уже созданных приемов. Чтобы облегчить читателю начало такой работы, предлагаем несколько упражнений. В них нужно рассмотреть (используя справочные средства интерфейса) основные элементы описания приема и объяснить, для чего эти элементы были введены. Можно пропускать элементы, заведомо относящиеся к другим разделам (интегрирование, дифференциальные уравнения и т.п.). При необходимости следует найти отладчиком ГЕНОЛОГа все задачи раздела "Элементарная алгебра", в которых прием применяется, и попытаться использовать для объяснений контексты его срабатываний. Ниже все подразделы указываются относительно раздела "Элементарная алгебра" оглавления базы приемов.

1. Прокомментировать последний прием подраздела "Минус - Равенство нулю разности $B - A$, если $B - A$ есть 0".

2. Прокомментировать прием подраздела "Цифры - Проверочный оператор "усмне0" - Произведение ненулевых сомножителей".

3. Прокомментировать прием подраздела "Умножение - Раскрывание скобок - Попытка раскрывания скобок для упрощения выражения".

4. Прокомментировать последний прием подраздела "Степени - Общая стандартизация выражений - Простейшие свойства степени - Умножение степеней с одинаковыми основаниями".

5. Прокомментировать последний прием подраздела "Умножение - Нормализатор разложения на множители "видумножение" - Тождества для непосредственного разложения на множители - Сумма кубов".

6. Прокомментировать последний прием подраздела "Умножение - Нормализатор разложения на множители "видумножение" - Разложение многочленов путем подбора целочисленных коэффициентов - Представление многочлена третьей степени в виде произведения линейного множителя и квадратного трехчлена".

7. Прокомментировать прием подраздела "Плюс - Решение уравнений - Системы уравнений - Вычитание линейных уравнений для устранения неизвестной".

8. Прокомментировать последний прием подраздела "Дробь - Устранение квадратичной иррациональности в знаменателе (общий случай)".

9. Прокомментировать прием подраздела "Степени - Решение уравнений - Разбор случаев по знаку неизвестного знаменателя дроби, являющейся основанием степени".

10. Прокомментировать прием подраздела "Число - Нормализатор выделения повторяющихся вхождений числовых выражений "повторчисло" - Умножение - Вынесение за скобку общего неизвестного множителя двух неизвестных слагаемых".

Указания

1. Теорема приема имеет вид $\forall_{ab}(a-b=0 \rightarrow b-a=0)$. Прием имеет заголовок "второйтерм", и можно предположить, что он выполняет замену $b-a$ на 0. Так как отсутствует указатель, уточняющий способ идентификации антецедента, то этот антецедент должен явным образом присутствовать в контексте заменяемого выражения. Фильтр "уровень(0 3)" определяет попытки применения приема на уровнях 0 и 3. Если на уровне 0 контекст еще не имел утверждения $a-b=0$, то на уровне 3 такое утверждение может появиться, и прием сработает при повторной попытке.

   В указателях приема находим терм "нормзнака(x2 минус плюс)". Чтобы определить его назначение, выделяем данный терм левой кнопкой мыши, а затем нажимаем правую кнопку. Появляются подсказки "нормзнака(x1 x2 x3)" и "Терм x2(x1) идентифицируется среди операндов операции x3 как множество x2-отрицаний x3-операндов терма, идентифицированного с x1". Следовательно, в нашем случае терм $-b$ идентифицируется среди слагаемых левой части антецедента как сумма взятых с обратными знаками слагаемых выражения b. Хотя лишь одно из имеющих знак "минус" слагаемых заменяемого терма будет идентифицировано с $-a$, рассматриваемый указатель позволит идентифицировать левую часть антецедента с суммой имеющих обратные знаки всех слагаемых заменяемого терма, вне зависимости от числа слагаемых с минусом. Для удаления подсказок дважды нажимаем клавишу "пробел".

   Далее идет указатель "эквивалентно". Чтобы определить его назначение, вызываем на экран подсказку "Теорема, имеющая вид равенства, используется для эквивалентной замены этого равенства на логическую константу "истина"

". Теперь видно, что наше исходное предположение о действии приема ошибочно - на самом деле не $b-a$ будет заменяться на 0, а равенство $b-a=0$ будет заменяться на константу "истина". Впрочем, пролистывая клавишами "курсор вверх-вниз" другие приемы того же подраздела, нетрудно найти и версию данного приема, лишенную указателя "эквивалентно", т.е. заменяющую $b-a$ на 0. Она срабатывает на уровнях 1 и 3. Рекомендуется самостоятельно, используя отладчик ГЕНОЛОГа, проанализировать возможность удаления одной из данных версий.

2. Теорема приема имеет вид $\forall_{ab}(\neg(a=0)\ \&\ \neg(b=0)\to\neg(ab=0))$. Согласно заголовку "спуск(усмне0)", прием относится к проверочному оператору "усмне0", усматривающему из контекста истинность утверждений вида $\neg(x=0)$. Текущий анализируемый терм оператора (значение программной переменной x1) - x. Фильтр "уровень(1)" определяет срабатывания приема на уровне 1. Указатель "блокпроверок(1 2)" означает, что антецеденты должны обрабатываться проверочными операторами. Вызываем подсказку для указателя "дистрибразвертка(фикс(0 1 1))". Из нее видно, что фактически прием не группирует сомножители анализируемого выражения в виде произведения двух частей a, b, а по отдельности рассматривает каждый сомножитель и обращается к проверочному оператору для определения отличия этого сомножителя от нуля. Наконец, указатель "спуск" означает, что при неудачной попытке усмотреть отличие от нуля какого-либо сомножителя сразу будет выдан отказ на усмотрение ненулевого значения всего произведения.

3. Теорема приема имеет вид $\forall_{abc}(c=a+b\to a+b=c)$. Так как она выглядит тавтологично, антецедент должен обращаться к каким-то нормализаторам, выполняющим необходимые преобразования. Наиболее простой способ просмотра нормализаторов - нажатие клавиши "5". В нашем случае после нажатия правая часть антецедента $a+b$ выделяется красным цветом, а под четвертым окном появляется текст нормализатора "стандплюс(замечание(буфер 0))". Таким образом, прием обращается к нормализатору "стандплюс" для обработки встретившейся при сканировании суммы $a+b$. Этому нормализатору передается комментарий (буфер 0), который заблокирует передачу в буферы нормализаторов всех промежуточных результатов раскрывания скобок. Фильтры "уровень(3), "условие", "тип(преобразовать)", "цель(упростить)" определяют срабатывание приема на уровне 3 при рассмотрении условия задачи на преобразование, имеющей цель "упростить". Фильтр "короче(результат теквхожд)" проверяет, что после раскрывания скобок получилось выражение, более короткое, чем исходное.

   Далее идет громоздкий фильтр "контекст(вид(теквхожд плюс(умножение(степень(плюс(x4 x5) x6) x7) x8)) не(контекст(подчинено(теквхожд x9)символ(x9 класс) пересекаются(связприставка(x9)x7)не(пересекаются(связприставка(x9) x4)) не(пересекаются(связприставка(x9)x5)))) единица(1 x6 x7)натуральное(x6) заменазнака(минус x7))". Чтобы прочитать его, используем многоцветную указку: нажимаем левую кнопку мыши на слове "вид(...)". Желтым цветом высвечивается фрагмент "вид(теквхожд плюс(умножение(степень(плюс(x4 x5) x6) x7) x8))". Он означает, что преобразуемое выражение $a+b$ должно быть представимо в виде $(d+e)^f g+h$. Таким образом, есть повод обращаться к попытке раскрывания скобок. Передвигая клавишами курсора выделенное желтое

пятно, находим сопровождающие указатели "единица(1 x6 x7)", "заменазнака(минус x7)". Они означают, что f, g могут обращаться в единицу, причем коэффициент g может оказаться отрицательным. Можно было бы уточнить, что f, g не должны равняться единице одновременно, однако это излишне: одновременное обращение их в единицу означает наличие в условии задачи вложенных сумм, которые устраняются приемом общей стандартизации, срабатывающим на меньшем уровне. Смещая желтое пятно на второй операнд фильтра, снова сталкиваемся с громоздкой конструкцией, имеющей вид "не(контекст(...))". Нажимаем "курсор вниз" дважды, чтобы можно было читать элементы вложенного "контекст"а. Первые три операнда "подчинено(теквхожд x9)", "символ(x9 класс)", "пересекаются(связприставка(x9)x7)" означают, что рассматриваемая сумма находится под описателем "класс", связывающая приставка которого пересекается со вторым множителем g. Далее идут два операнда, требующие, чтобы эта связывающая приставка не пересекалась с основанием степени $d+e$. Смысл ограничения ясен - здесь сумма $d + e$ играет роль коэффициента, и раскрывание скобок нецелесообразно.

Переходим к следующему фильтру приема - "или(не(контекст(подчинено(теквхожд x4) символ(x4 отображение))) контекст(подчинено(теквхожд x4) символ(x4 сигнум)))". Чтобы лучше понять его смысл, можно было бы провести эксперимент - убрать фильтр, перекомпилировать прием и провести тестовый запуск решателя на нескольких разделах, где вероятно появление описателя "отображение" (например, на математическом анализе). Те задачи, которые при этом перестанут решаться, изменят ответ либо замедлятся, могут прояснить ситуацию. Разумеется, после всего это нужно восстановить прием нажатием клавиши "Shift-Б". Мы не будем здесь прибегать к такому эксперименту, а заметим лишь, что раскрывание скобок при наличии внешнего описателя "отображение" требует особого управления - здесь легко нарушить специфическую стандартизацию, связанную с заданием функций. В порядке исключения, разрешается раскрывать скобки под "сигнумом", который вряд ли окажется связан с подобной стандартизацией.

Фильтр "не(цель(редуцирование))" относится к процедурам базы теорем и нас сейчас не интересует. Наконец, фильтр "или(не(цель(известны))меньше(число(входит(x4 корень))набор(1 2 0)))" блокирует раскрывание скобок в очень громоздких выражениях при упрощении ответа задач, имеющих цель "известно ...". Такие задачи возникают в планиметрии и в аналитической геометрии.

Переходим к указателям приема. Первый из них - "идентификатор(1)" - уточняет способ обработки антецедента. Переменные правой части антецедента будут к моменту его обработки уже идентифицированы, в то время как переменная c еще не определена. Поэтому правая часть обрабатывается нормализатором "стандплюс", и c идентифицируется с результатом обработки. Указатель "попытка(стандплюс теквхожд)" проверяет отсутствие комментария (стандплюс $a + b$). Если его не было, и прием не сработал (из-за того, что новое выражение оказалось сложнее старого), то такой комментарий вводится и блокирует повторные попытки раскрывания скобок. Указатель "замечание(стандплюс результат)" вводит комментарий (стандплюс c), чтобы заблокировать попытки применения приема к выражению c. Этот комментарий будет автоматически корректироваться при последующих преобразованиях c и может пригодиться, если снова появится выражение, в котором станет возможным раскрывание

скобок.

4. Теорема приема имеет вид $\forall_{abc}(0 \le a \to a^b a^c = a^{b+c})$. Заголовок "второйтерм" определяет замену слева направо. Фильтр "уровень(0 3)" инициирует попытки применения приема на уровнях 0 и 3. Уровень 3 понадобится, если на уровне 0 еще не усматривалась неотрицательность a, но после вывода следствий или эквивалентных преобразований она стала очевидной. Фильтр "или(не(целое(х2))не(десчисло(х1))не(констдробь(х3)))" блокирует срабатывание, если b - целое, a - десятичная константа, c - дробная константа. Например, будет заблокировано нарушающее принятую стандартизацию преобразование $2\sqrt{2}$ в $2^{3/2}$/. Ту же роль играет двойственный фильтр "или(не(целое(х3))не(десчисло(х1))не(констдробь(х2)))".

   Далее идет фильтр "или(не(тип(преобразовать)) не(контекст(комментарий(нормИнтеграл х4)входит(х4 х1)или(контекст(вид(х2 дробь(1 х5)) целое(х3)) контекст(вид(х3 дробь(1 х5))целое(х2))))))". Он означает следующее. Пусть решается задача на преобразование, имеющая комментарий (нормИнтеграл х4), т.е. преобразуется подынтегральное выражение при вычислении неопределенного интеграла, и х4 - переменная интегрирования. Тогда, если х4 входит в a, запрещается, чтобы один из множителей был целочисленной степенью, а другой - радикалом.

   Указатель "блокпроверок(1)" определяет обработку антецедента с помощью проверочного оператора. Указатель "нормализатор" включает процедуру, предпринимающую попытку немедленной общей стандартизации надтермов заменяемого терма после проведения замены. Указатель "единица(1 х2 х3)" разрешает вырожденные значения 1 показателей степени b, c.

5. Теорема приема имеет вид $\forall_{ab}(a^3+b^3 = (a+b)(-ab+a^2+b^2))$. Заголовок "замена(второйтерм видумножение)" показывает, что прием относится к нормализатору "видумножение" и выполняет замену слева направо. Фильтр "постпозиция(фикс(0 1 2)фикс(0 1 1))" требует, чтобы второе слагаемое a^3 располагалось в сумме после первого. Таким образом отбрасываются симметричные случаи, и трудоемкость попыток идентификации уменьшается вдвое.

   Далее идет фильтр "коммент(числоценка)", означающий, что при наличии комментария "числоценка" прием блокируется. В принципе, мы могли бы и не уточнять смысл этого ограничения - ясно, что данный комментарий нужен для каких-то специальных случаев. Однако, в качестве иллюстрации попробуем провести небольшое исследование и получить нужную информацию. Попытка найти что-либо полезное в справочнике символа "числоценка" успеха не приносит. Поэтому, чтобы понять смысл данного фильтра, попробуем найти те приемы, которые вводят комментарий "числоценка" при обращении к нормализатору. Возвращаемся в главное меню системы и переходим через него в программу символа "текприем". Это - процедура поиска нужного приема в базе приемов. При сканировании базы приемов на каждом приеме будет происходить обращение к данной процедуре. Нажимаем F3 для получения справки о ее программных переменных. Находим, что значением переменной х5 будет описание текущего приема (объединение списка указателей, нормализаторов и заключенных в общие "скобки" терма "условие(и(...))" фильтров). Возвращаемся в просмотр программы и нажимаем "р" для ее редактирования. После оператора "метка(икс(одиннадцать))" начинаем составление списка усло-

вий на отбираемые приемы. Оператор "входит(x11 x5)" позволит нам просматривать все элементы x11 описания приема. Оператор "не(логсимвол(x11))" отбросит те из них, которые являются логическими символами. Таким образом, далее x11 - терм. Нам нужно найти обращения к нормализаторам, содержащие логический символ "числоценка". Поэтому помещаем в программу операторы "входит(числоценка x11)" и "входит(быстрпреобр x11)". По-видимому, пока этим можно ограничиться, и далее помещаем заключительный оператор "ответ(набор(См))". Напомним, что для просмотра будут выбираться те приемы, на которых операторное выражение "текприем" имеет своим значением однобуквенный терм "См". Завершив редактирование программы, возвращаемся в оглавление базы приемов (не входя в просмотр приема) и нажимаем F8. Начнется просмотр всех приемов, удовлетворяющих введенным ограничениям. Переход к очередному приему будет обеспечиваться нажатием клавиши "курсор влево". Заметим, что в цикле поиска невозможно выйти из просмотра приема в оглавление приемов. Если это все же нужно сделать, обрываем цикл просмотра нажатием клавиши "Esc". Тогда возвращение в базу приемов выведет нас на последний просматривавшийся прием. В нашем случае первый найденный прием относится к нормализатору "стандменьшеилиравно". Нажимая клавишу "5", входим в цикл просмотра нормализаторов приема (PageUp-PageDn, выход - End) и обнаруживаем нормализатор "видумножение(замечание(нормуравнение) замечание(числоценка))", который и вводил комментарий "числоценка". Выходя в оглавление, выясняем, что нормализатор "стандменьшеилиравно" используется при стандартизации нестрогих неравенств для известных параметров на этапе редактирования ответа задачи на описание. Видимо, здесь комментарии "нормуравнение" и "числоценка" играли роль опций, ослабляющих попытки разложения на множители в ситуации завершающего редактирования. Очевидно, удлиняющее запись разложение куба суммы на множители здесь нецелесообразно. Снова запускаем цикл просмотра приемов (F8) для поиска других ситуаций использования комментария. Пропуская несколь приемов, аналогичных рассмотренному, в некоторый момент обнаруживаем прием для разложения на множители числителя: $\forall_{abcdef}(f = b{+}c \,\&\, d{-}\text{rational} \,\&\, \neg(\text{знаменатель}(d){-}\text{even}) \to a(b{+}c)^d/e = af^d/e)$. Так как его описание достаточно громоздкое, нажимаем "Ctr-F3" и убираем с экрана третье окно. Затем нажимаем "5" и просматриваем нормализаторы. Обнаруживаем громоздкий нормализатор "видумножение(...)", внутри которого усматривается фрагмент "замечание(условие(и(константа(x2)константа(x3)десчисло (x5))) числоценка)". Он означает, что комментарий "числоценка" будет вводиться, если выражения b, c, e - константные. Видимо, изначальное происхождение данного комментария - ориентировать нормализатор на работу с константными выражениями. Некоторые действия (типа преобразования суммы кубов) здесь будут блокироваться, некоторые (типа вынесения наружу общего натурального множителя целочисленных коэффициентов суммы) - активироваться. Судя по первому приему, позднее этот комментарий был использован в прочих ситуациях как дополнительный способ ограничения разложения на множители.

Возвращаемся к рассмотрению компонент описания приема для суммы кубов. Следующий фильтр - "или(входит(нормИнтеграл комментарии)не(контекст(комментарий(нормИнтеграл x3)тригаргумент(корень x4)входит(x3 x4))))". Если обращение к разложению на множители произошло из задачи преобразова-

ния подынтегрального выражения, причем переменная интегрирования встречается под тригонометрической операцией, то прием применяется только при наличии комментария "нормИнтеграл". Такие комментарии вводятся нормализатором неопределенного интегрирования "нормИнтеграл" при некоторых рекурсивных обращениях.

Еще один фильтр из математического анализа - "не(контекст(текущаязадача описать)цель(текущаязадача областьграницы))". Прием блокируется, если решается задача на явное задание области по системе ее граничных линий. Рекомендуется в качестве упражнения найти ту задачу из раздела "Кратные интегралы", на решение которой повлияет отмена данного фильтра. Для этого фильтр удаляется, прием перекомпилируется, и выполняется цикл решения задач в указанном разделе.

Фильтр "не(контекст(тип(текущаязадача преобразовать)или(цель (текущаязадача смпосылка) цель(текущаязадача новаяпеременная)цель(текущаязадача значениеинтеграла)цель(текущаязадача редакция))))" перечисляет целевые установки задач на преобразование, при которых выполнение приема нецелесообразно. Как и в случае предыдущего фильтра, рекомендуем найти задачи, для которых отсутствие данного ограничения приводит к трудностям - замедлению либо утере ответа. Так как многие фильтры вводились при обучении ради единственного примера, повторный анализ, по накоплении многих примеров, может привести к существенному усовершенствованию решающих правил. Наличие циклов тестирования по задачнику обычно позволяет модифицировать даже очень старые приемы, сохраняя полную работоспособность решателя на освоенном обучающем материале. Если коррекция приема приводит к утере ответов и замедлению решений, то обычно достаточно бывает создать дополнительные приемы, реже - приходится продолжать коррекцию других приемов. Разумеется, могут встретиться особые точки, для которых процесс коррекций и модификаций ветвится и требует очень большой работы, однако чаще всего глубина модификаций невелика. Размеры решателя не должны пугать тех, кто столкнется с "плохими" приемами и будет вынужден их переделывать - опыт показывает, что такие переделки, как правило, имеют локальный характер.

Фильтр "не(контекст(тип(текущаязадача описать)цель(текущаязадача неравенства)))" блокирует применение приема, если решается задача на описание для определения области интегрирования.

Указатель "уровень(4)" определяет срабатывание приема нормализатора на уровне 4. Указатель "знаксуммы(минус фикс(0 1))" определяет попытку идентификации с одновременным изменением знаков кубов на "минус". При этом будет изменен знак заменяющего терма. Указатель "модификатор" требует, чтобы сумма не имела других слагаемых. Указатель "замечание(6 фикс(0 2 2))" вводит комментарий $(6, \quad -ab+a^2+b^2)$, который заблокирует попытку разложения второго сомножителя по формуле корней квадратного трехчлена. Рекомендуем найти все использующие этот комментарий приемы нормализатора "видумножение", написав подходящую программу поиска "текприем". Указатель "замечание(условие(и(входит(нормчислитель комментарии) контекст(тригаргумент(корень x4)))) фильтрудвоения)" вводит комментарий "фильтрудвоения", если нормализатор имел комментарий "нормчислитель", а преобразуемое выражение имело тригонометрические операции. Рекомендуем выяснить роль комментария "фильтрудвоения", найдя с помощью процедуры "текприем" все при-

емы нормализатора, фильтры которых содержат ссылку на этот комментарий.

6. Теорема приема имеет вид $\forall_{abcdefghijk}(fh = d \ \& \ gj = a \ \& \ 0 < g \ \& \ gi = b - fj \ \& \ gh + fi = c \rightarrow ae^3 + be^2k + cek^2 + dk^3 = (fk + ge)(hk^2 + iek + je^2))$. Заголовок такой же, как в предыдущем случае - "замена(второйтерм видумножение)". Фильтры "целое(x1)", "целое(x2)", "целое(x3)", "целое(x4)" определяют идентификацию коэффициентов a, b, c, d с целочисленными константами. Фильтр "постпозиция(фикс(0 1 4)фикс(0 1 1))" требует, чтобы dk^3 идентифицировалось со слагаемым, идущим после слагаемого ae^3. Ввиду симметрии теоремы, это вдвое сокращает трудоемкость попыток идентификации. Фильтр "меньше(количествооперандов(теквхожд)5)" - ускоряющий. Он размещается компилятором в начале программы, так как не содержит каких-либо переменных, которые вначале нужно было бы идентифицировать. Поэтому заведомо неприемлемые длинные суммы сразу отбрасываются. Фильтр "коммент(группировка)" блокирует применение приема, если внешняя процедура "видумножение" обратилась к данной процедуре для разложения на множители выражения, в котором часть слагаемых уже сгруппирована. Фильтр "контекст(разряд(теквхожд степень x12)не(второйсимвол(x12 2)))" тоже ускоряющий. Он обрабатывается до начала идентификации слагаемых и отбрасывает суммы, не содержащие степеней, кроме, быть может, квадратов. Фильтр "коммент(6 теквхожд)" блокирует разложение на множители трехчленов, возникших при разложении на множители суммы или разности кубов. Фильтр "не(константа(корень))" отбрасывает константные суммы. Фильтр "коммент(нормуравнение)" блокирует прием, если имеется комментарий "нормуравнение", указывающий на ослабленный режим разложения на множители. Аналогичную роль играет фильтр "коммент(множителимн)". Наконец, фильтр "меньше(2 количествооперандов)" блокирует попытку идентификации для суммы с двумя слагаемыми.

Указатель "модификатор" запрещает наличие в сумме слагаемых, не указанных в теореме. Указатель "уровень(3)" определяет срабатывание приема нормализатора на уровне 3. Указатель "программа(1 2 3 4 5)" определяет реализацию всех антецедентов с помощью арифметических вычислений. Указатели "подстановка(фикс(0 1 2)x2 0)", "подстановка(фикс(0 1 3)x3 0)" разрешают обращение в 0 коэффициентов b, c. Так как суммы длины 2 не рассматриваются, хотя бы один из этих коэффициентов окажется отличным от 0. Указатели "перечень(x1 десчисло(x1))", "перечень(x4 десчисло(x4))" определяют идентификацию a, d с произведением всех численных коэффициентов соответствующих слагаемых. Лишь после этого будут выполняться попытки усмотреть в остатках их множителей кубы некоторых выражений e, k. Указатель "пересечениесписков(фикс(0 1 2)фикс(0 1 3))" объясняет компилятору, что второе и третье слагаемое могут не иметь своим заголовком символ "умножение", хотя каждое из них в теореме содержит по три сомножителя. Как следствие, компилятор не будет уточнять заголовки этих слагаемых. Корректная идентификация будет обеспечиваться применением процедур, задаваемых справочником "пересечениесписков". Указатель "единица(1 x1 x2 x3 x4)" разрешает обращаться коэффициентам a, b, c, d в единицу. Указатель "заменазнака(минус x1 x2 x3 x4)" определяет передачу этим коэффициентам знака "минус" перед слагаемым. Наконец, указатель "титр(набор(1 терм(фикс(0 2)фикс(1)фикс(2)фикс(4)фикс(5))параметры(x6 x7 x8 x9 x10)))" определяет составление сопровождающих срабатывание приема пояснений. Для просмотра

шаблона текста пояснений нажимаем клавишу "6". Под изображением приема появляется запись "Подбираем целочисленные коэффициенты разложения (), используя соотношения (),(),(),() ". Элемент указателя "терм(фикс(0 2)фикс(1)фикс(2)фикс(4)фикс(5)" ссылается на подставляемые вместо пар скобок термы. Элемент "параметры(x6 x7 x8 x9 x10)" уточняет, что вместо вспомогательных переменных f, g, h, i, j в текст пояснения будут подставляться первые неиспользуемые большие буквы.

7. Теорема приема имеет вид: $\forall_{abcdefgx}(agx + b = c \ \& \ \neg(a = 0) \rightarrow dgx + e = f \leftrightarrow ae - bd = af - cd)$.

   Заголовок - "второйтерм", т.е. прием выполняет эквивалентную замену слева направо. Фильтры "уровень(1)", "тип(описать)", "условие", "корень" определяют применение приема к условию задачи на описание на уровне 1. Первый антецедент не выделен указателями и идентифицируется с другим условием. Фильтр "неизвестные(2)" означает, что задача должна иметь более одной неизвестной. Фильтры "неизвестная(x23)", "известно(x1)", "известно(x4)", "известно(x3)", "известно(x6)", "не(входит(x23 x2))", "не(входит(x23 x5))" указывают, что x - неизвестная, выражения a, d, c, f не содержат неизвестных, а выражения b, e не содержат неизвестной x. Таким образом, оба уравнения задачи линейны относительно x. Фильтр "не(контекст(неизвестная(x8)входит(x8 x2) не(входит(x8 x5))))" требует, чтобы каждая неизвестная выражения b входила в выражение e. Тогда новое уравнение не приобретет новых неизвестных, а старая неизвестная x окажется исключенной. Фильтр "или(не(линейноеуравнение(корень неизвестные))линейноеуравнение(фикс(1)неизвестные))" блокирует применение приема, если он нарушает линейность текущего уравнения, комбинируя его с нелинейным. Фильтр "не(Входит(известно цели))" блокирует применение приема в задаче, имеющей цель (известно ...), так как эта задача решается путем вывода следствий в блоке анализа. Фильтр "не(Входит(независит цели))" блокирует применение приема в задаче, имеющей цель (независит ...), запрещающей зависимость ответа от заданных переменных. Такие задачи обычно решаются другими средствами - путем приравнивания нулю неизвестного коэффициента перед теми переменными, которые должны быть устранены.

   Указатель "блокпроверок(2)" определяет обработку второго антецедента проверочным оператором. Указатели "единица(0 x2 x5)", "единица(1 x1 x4 x7)" разрешают вырожденные нулевые значения b, e и единичные значения a, d, g. Указатель "заменазнака(минус x1 x4)" обеспечивает передачу коэффициентам a, d знака "минус" перед слагаемым. Указатель "титр(набор(1 терм(x23 фикс(фикс(1))x1 минус(x4))))" определяет подстановку в шаблон поясняющего текста необходимых термов. Указатель "комментарий(2 титр(стоп))" передает проверочному оператору "усмне0", обрабатывающему второй антецедент, комментарий (титр стоп), блокирующий отображение на экране обращения к проверке.

   Для просмотра нормализаторов приема нажимаем клавишу "5", и далее сменяем текущий нормализатор клавишами "PageUp-Page-Dn". Обратим внимание на нормализаторы "стандплюс(замечание(титр(набор(2))))", "стандплюс(замечание(титр(набор(3))))", обрабатывающие разности $ae - bd$, $af - cd$ соответственно. Термы "замечание(титр(набор(...)))" обеспечивают выдачу на экран второго и третьего фрагментов сопровождающего текста, комментирующих об-

ращения к этим нормализаторам.

8. Теорема приема имеет вид $\forall_{abcdefgpq}(\neg(a+b=0)\ \&\ g=a-b\ \&\ p=a^2\ \&\ q=b^2\ \&\ c=p-q\ \&\ (\neg(g=0)\ \vee\ \neg(c=0))\rightarrow f/d(a+b)^e=f(g/c)^e/d)$. Заголовок приема - "второйтерм", т.е. выполняется тождественная замена слева направо. Фильтры "уровень(4)", "условие", "тип(преобразовать)", "цель(упростить)" определяют срабатывание приема на уровне 4 в условии задачи на преобразование, имеющей цель "упростить". Фильтры "не(цель(нормИнтеграл))", "коммент(длина)" блокируют применение приема при интегрировании и на этапе заключительной обработки ответа. Фильтр "контекст(вид(x1 умножение(x9 степень(x10 дробь(x11 умножение(2 x12)))))или(заголовок(x10 плюс)контекст(алгебрвхождение(x10 x13)заголовок(x13 плюс)))заменазнака(минус x9)единица(1 x9 x12))" требует, чтобы выражение a имело своим сомножителем степень с дробным показателем, знаменатель которого делится на 2. При этом требуется, чтобы основание степени либо являлось суммой, либо содержало сумму, достижимую из корня через алгебраические операции. Фильтр "не(контекст(вид(x2 плюс(x9 умножение(x10 степень(x11 дробь(x12 x13)))))не(константа(x11)) заменазнака(минус x10)единица(1 x10)))" означает, что b не должно представлять собой невырожденную сумму, некоторое слагаемое которой имеет своим сомножителем степень с дробным показателем и неконстантным основанием. Следующий фильтр имеет ссылку на переменную c, отсутствующую в заменяемом выражении. Поэтому временно откладываем чтение фильтров и переключаемся на указатели, определяющие обработку антецедентов.

   Указатель "идентификатор(2 3 4 5)" означает, что антецеденты со второго по пятый представляют собой равенства, используемые для ввода в рассмотрение новых объектов. Второй антецедент идентифицирует переменную g с разностью выражений a и b. На нее будут домножаться числитель и знаменатель для исключения иррациональности. К разности применяются нормализаторы общей стандартизации. Третий и четвертый антецеденты идентифицируют выражения p, q с квадратами выражений a, b, обработанными нормализаторами "нормстепень" и "стандплюс". Наконец, пятый антецедент идентифицирует переменную c с разностью квадратов, обработанной нормализаторами "нормплюс" и "видумножение".

   Теперь возвращаемся к третьему окну описания приема. Следующий фильтр имеет вид "не(контекст(вид(x3 плюс(x9 умножение(x10 степень(x11 дробь(x12 умножение(2 x13))))))единица(1 x10 x13)заменазнака(минус x10)))". Он проверяет, что разность квадратов c не имеет слагемого, сомножителем которого была бы степень с дробным показателем, имеющим четный знаменатель. Это означает, что от иррациональности в знаменателе удалось избавиться. Фильтр "не(константа(фикс(0 1 2 2 1)))" требует, чтобы выражение $a+b$ не было константным - для этого случая созданы отдельные приемы. Фильтр "не(контекст(алгебрвхождение(фикс(0 1 2 2 1)x9)символ(x9 дробь)))" блокирует применение приема, если выражение $a+b$ имеет дробь, достижимую из его корня через алгебраические операции. В таких ситуациях сначала должно быть выполнено исключение дробей (например, путем сложения дробных выражений). Наконец, фильтр "меньше(числочленов(x3 плюс)плюс(числочленов(x15 плюс)числочленов(x16 плюс)))" требует, чтобы при вычитании b^2 из a^2 произошло приведение подобных членов, уменьшившее число слагаемых. Без этого преобразование будет фиктивным - разложение на множители знаменателя и

сокращение дроби вернут к исходному выражению.

Возвращаемся к рассмотрению указателей приема. Указатель "блокпроверок(1 6)" определяет обработку первого и шестого антецедентов проверочным оператором. Первый антецедент почти избыточен - он перепроверяет о.д.з. исходного выражения. Однако, явное упоминание его в теореме будет использовано указателем "вывод(не(равно(x3 0))эквивалентно(1))", выводящим дополнительное следствие $c \neq 0$ и регистрирующим его эквивалентность (в предположении истинности прочих антецедентов) первому антецеденту. Шестой антецедент проверяет, что либо $a - b \neq 0$, либо $a^2 - b^2 \neq 0$. Иногда последнее оказывается легче проверить, чем первое. Таким образом обеспечивается корректность преобразования. Указатели "заменазнака(минус x6)" и "единица(1 x4 x5)" разрешают вырожденные единичные значения d, e и отнесение внешнего минуса к f. Наконец, указатель "титр(набор(1 терм(фикс(фикс(0 1))степень(x7 x5))))" определяет выражения, подставляемые в шаблон пояснений к срабатыванию приема.

9. Теорема приема имеет вид $\forall_b(\neg(b = 0) \to 0 < b \ \vee \ b < 0)$. Указатель "контрольвывода(степень(дробь(x1 x2)x3))" определяет инициализацию приема при усмотрении выражения вида $(a/b)^c$. Заголовок приема - "выводусловия", т.е. происходит занесение утверждения $0 < b \ \vee \ b < 0$ в список условий задачи. Фильтры "уровень(0)", "условие", "тип(описать)" означают, что прием применяется на уровне 0 в задаче на описание, причем усмотренное выражение $(a/b)^c$ входит в условие задачи. Фильтр "не(известно(x2))" требует, чтобы знаменатель b содержал неизвестные. Фильтр "контекст(вид(корень равно(x4 x8))не(известно(x4))или(заголовок(x8 0)контекст(подтерм(умножение(теквхожд x2 x9))единица(1 x9)))не(контекст(вид(x4 плюс(x5 x6))не(контекст(обобщ-множитель(x5 x7)или(контекст(вид(x7 x1))контекст(вид(x7 x2))))) единица(0 x6))))" достаточно громоздкий, поэтому читаем его по частям, используя выделение фрагментов многоцветной указкой. Получаем, что текущее условие задачи должно иметь вид уравнения "равно(x4 x8)", причем часть x4 содержит неизвестные, а часть x8 либо нулевая, либо имеет вид $(a/b)^c \cdot b \cdot A$. Кроме того, каждое слагаемое x5 части x4 должно иметь своим обобщенным множителем либо a, либо b. Выполнение этого условия означает, что после разбора случаев, возможно, удастся сократить уравнение на b или на a. Фильтр "не(контекст(условие(x4)входит(разборслучаев комментариийусловия(x4))))" означает отсутствие условия, помеченного комментарием "разборслучаев". Таким образом, сначала должны быть реализованы разборы случаев, инициированные ранее. Фильтр "неизвестные(1)" говорит, что задача должна иметь единственную неизвестную. В противном случае срабатывание приема на уровне 0 может оказаться слишком поспешным. Фильтр "не(контекст(подтерм(равно(умножение(теквхожд x4)0))единица(1 x4)заменазнака(минус x4)))" блокирует срабатывание приема в вырожденной ситуации, когда уравнение имеет вид $(a/b)^c \cdot A = 0$ и разбор случаев не нужен. Фильтр "не(входит(связка цели))" блокирует применение приема при решении дифференциальных уравнений. Фильтры "не(легковидеть(меньше(0 x2)))", "не(легковидеть(меньше(x2 0)))" блокируют срабатывание приема в ситуациях, когда знак знаменателя усматривается из контекста.

Указатель "примечание(разборслучаев)" сопровождает выведенную дизъюнкцию комментарием "разборслучаев", форсирующим ускоренный переход к рас-

смотрению случаев. Кроме того, данный комментарий блокирует возможные попытки усмотреть тавтологичность дизъюнкции и заменить ее на константу "истина". Указатель "блокпроверок(1)" определяет обработку антецедента проверочным оператором. Наконец, указатели "титр(набор(1 терм(x2)))", "комментарий(1 титр(стоп))" обеспечивают сопровождение срабатывания приема пояснениями.

10. Теорема приема имеет вид $\forall_{abc}(ab+ac=a(b+c))$. Фильтры "не(известно(x1))", "не(известно(x2))", "не(известно(x3))", "неизвестные(2)" означают, что как общая часть a двух слагаемых, так и остаточные произведения b, c содержат неизвестные, причем общее число неизвестных задачи более единицы. Фильтр "или(контекст(внешвхождение(x4)вид(x4 плюс(степень(x2 x6)степень(x3 x6) x5))единица(0 x5)единица(1 x6)буфер(новыенеизвестные базавхождения(x4))) и(контекст(вид(x2 минус(x4))вид(x3 минус(x5))контекст(внешвхождение(x6) вид (x6 плюс(степень(x4 x7)степень(x5 x7)x8))единица(0 x8)единица(1 x7) буфер(новыенеизвестные базавхождения(x6))))))" анализирует контекст вхождения текущей суммы. Этот контекст состоит из вхождений, расположенных вне суммы в том же терме задачи, либо в других термах, отобранных для усмотрения повторяющихся вхождений выражений. Первая часть фильтра указывает на существование в контексте вхождения x4 выражения вида $b^k + c^k + r$, вторая - вхождения x6 выражения вида $(-b)^k + (-c)^k + r$. Элементы "буфер(новыенеизвестные базавхождения(x4))", "буфер(новыенеизвестные базавхождения(x6))" нужны для регистрации в комментарии к нормализатору (новыенеизвестные ...) ссылок на те термы задачи, в которых обнаруживаются указанные вхождения. По окончании преобразований текущего терма будет предпринято повторное сканирование имеющихся в данном комментарии термов.

Анализ целесообразности применения приема

При обучении решателя ранее введенные фильтры приема могут устаревать из-за появления новых приемов или изменения старых. В новой задаче, где возникает срабатывание приема, оно может оказаться бесполезным и заводящим задачу в тупик. Наоборот, иногда старый фильтр приема оказывается чрезмерно жестким и блокирует необходимое для решения новой задачи срабатывание. Во всех этих ситуациях коррекция фильтра, подсказываемая новой задачей, требует учета возможных срабатываний приема в других задачах. Здесь различаются два случая. Если нужно усилить ограничения на срабатывание приема, то достаточно рассмотреть лишь те задачи, в которых прием срабатывал раньше. Их список быстро определяется отладчиком ГЕНОЛОГа. После усиления фильтра выполняется повторная прокрутка найденных задач и анализируются возможные коллизии. Во втором случае нужно ослабить ограничения на срабатывание приема. Это - более трудоемкая работа, так как требует уже полной прокрутки по всем разделам задачника, где прием может сработать. Для иллюстрации использования в таких ситуациях отладчика ГЕНОЛОГа разберем ряд примеров, относящихся к вышеизложенным первым подразделам элементарной алгебры.

1. Найти все задачи раздела "Элементарная алгебра", в которых срабатывает последний прием подраздела "Умножение - Усмотрение разности квадратов - Усмотрение разности квадратов". В каких задачах его срабатывание не нужно?

2. Найти все задачи раздела "Элементарная алгебра", в которых срабатывает последний прием подраздела "Дробь - Устранение кубической иррациональности в знаменателе". Выяснить, к каким последствиям для решения прочих задач данного раздела приведет отбрасывание двух последних фильтров "не(константа(фикс(0 1 2 2 1)))", "меньше(числочленов(х3 плюс) плюс(числочленов(х15 плюс)числочленов(х16 плюс)))".

3. Найти все задачи раздела "Элементарная алгебра - Упрощение выражений", в которых срабатывает последний прием подраздела "Степени - Разбор случаев по знаку сомножителя под радикалом при упрощении выражений". Выяснить, к каким последствиям приведет отбрасывание фильтра "контекст(подчинено(теквхожд х4)символ(х4 степень)второйсимвол(х4 дробь)второйсимвол(второйоперанд(х4)2 4))".

4. Найти задачи раздела "Элементарная алгебра", в которых последний прием подраздела "Степени - Решение уравнений - Показательные уравнения - Представление произведения неизвестных степеней с известными основаниями в виде степени произведения этих оснований, домноженной на известный коэффициент", - имеет ненужное срабатывание. Найти все задачи того же раздела, которые перестают решаться при отключении данного приема. Попробовать усилить фильтры таким образом, чтобы прием срабатывал лишь в последних задачах.

5. Найти задачи раздела "Элементарная алгебра", в которых прием подраздела "Степени - Решение уравнений - Выражение одной неизвестной через другую, если известно произведение их степеней" - имеет ненужное срабатывание. Решение каких задач данного раздела задачника нарушится, если у приема отбросить фильтр "конец(альтернатива(уровень(3)...))" ?

Указания

1. Начинаем с того, что заходим в корневое меню оглавления задачника и выделяем пункт "Элементарная алгебра". Затем нажимаем клавишу "Shift-2" и попадаем в оглавление приемов. Переходим в просмотр нужного нам приема и нажимаем клавишу "А" (кириллица). После небольшой паузы, необходимой программе для поиска по разделу задачника, нажимаем "Shift-3" и возвращаемся в оглавление задачника. Все найденные задачи, имеющие хотя бы одно срабатывание рассматриваемого приема, оказались зарегистрированы в буфере задачника. Чтобы перейти к просмотру буфера, нажимаем "ф". В буфере обнаруживаем единственный подраздел "Усмотрение разности квадратов", название которого копирует название соответствующего концевого пункта оглавления приемов. Входим в данный подраздел. В нем оказывается список из двух десятков номеров отобранных задач. Здесь дублируются только ссылки на задачи, но не сами задачи. Поэтому, изменяя задачу через буфер, одновременно изменяем ее и по "основному" месту регистрации в задачнике. Однако, удаление задачи из буфера не означает удаления основной ссылки на нее. Для анализа срабатываний приема на отобранных задачах можно либо включить цикл "прокрутки", начинающейся с текущей выделенной задачи (клавиша "Ctr-з"), либо предпринять трассировку отдельных задач. Чтобы определить, в каких задачах срабатывание приема необходимо, выделяем первый пункт списка отобранных

задач и нажимаем клавишу "й". Это вызовет цикл попыток решения данных задач с заблокированным приемом. В тех случаях, когда ответ не изменяется и находится быстрее, чем при использовании приема, пункт списка помечается знаком вопроса. Для нашего примера таких знаков вопроса оказывается два - на пунктах 16 и 17. Можем посмотреть, например, срабатывание приема в пункте 16. Для этого заходим в просмотр задачи и нажимаем "г" для перехода в оглавление приемов. Далее выходим в просмотр нашего приема и нажимаем "Ctr-Enter". При попытке обратиться к приему происходит прерывание отладчика ЛОСа. На экране видна лишь часть выполняемого фрагмента программы, обрывающаяся на текущем операторе. Для просмотра всего фрагмента нажимаем "ф". Нас интересует заключительный момент срабатывания приема, поэтому далее ведем трассировку через отладчик ЛОСа. Нажимаем клавишу "курсор вниз" для выделения оператора программы и перемещаем выделенный оператор клавишей "курсор вправо", пока он не достигнет оператора "ветвь 1". Вместо использования клавиш курсора можно просто подвести курсор мыши к указанному оператору и нажать левую кнопку (отказ от выделения оператора - путем нажатия клавиши "курсор вверх"). Снова нажимаем "Ctr-Enter", чтобы отладчик ЛОСа выполнил действия вплоть до выделенного оператора включительно. Для прорисовки полного фрагмента программы опять нажимаем "ф". Видно, что сейчас будет выполняться реализующий действия приема оператор "заменавхождения". Чтобы просмотреть его выполнение на уровне трассировки ГЕНОЛОГа, нажимаем клавиши "пробел" и "Enter". Появляется кадр "Условие $(a-b)(b-a)$ заменяется на $-a^2+b^2$". Можно просмотреть описание приема, нажав клавишу "б". Если также нажать "п", то на экране появится группа равенств, определяющих для каждой теоремной переменной то значение, которое она получила при данном срабатывании. Через отладчик ЛОСа нетрудно выяснить, что прием был использован при упрощении условия $\neg((a+b)(a-b)=0)$. Очевидно, здесь его срабатывание действительно ненужно. По завершении упражнения сбрасываем буфер задачника, нажимая из оглавления клавишу "О" (кир.)

2. Так же, как в предыдущем случае, создаем в буфере задачника список задач, имеющих срабатывание рассматриваемого приема. Оказывается, что в нем всего одна задача - на упрощение выражения. Затем возвращаемся в просмотр приема, входим в редактирование третьего окна (Ctr-3) и удаляем два последних фильтра. По окончании редактирования нажимаем "Enter" и "F3". Таким образом возникает измененная версия приема. Переходим в оглавление задачника, входим в меню подраздела "Элементарная алгебра" и выделяем первый пункт этого меню. Для прокрутки решателя по всему подразделу нажимаем "Ctr-з" (кир.). Начинается цикл решения задач. После него снова оказываемся в меню подраздела "Элементарная алгебра". Для просмотра результатов прокрутки нажимаем клавишу "з". Появляется диалоговый блок со статистикой прокрутки, из которого видно, что ни одна задача не замедлилась и не изменила ответа. Это означает, что данные фильтры, возможно, вообще не нужны. Они были введены из общих соображений, по аналогии с приемами для устранения квадратичной иррациональности. При желании их можно удалить, не исказив поведения решателя на обучающем материале. В качестве упражнения рекомендуем придумать задачу, которая решалась бы по-разному при наличии этих фильтров и при их удалении. По окончании упражнения возвращаемся

в просмотр измененного приема и восстанавливаем его первоначальный вид нажатием клавиши "Б".

3. Находим список всех задач, имеющих срабатывание приема, используя указанный выше интерфейс отладчика ГЕНОЛОГа. Этот список содержит две задачи. Отбрасываем указанный в упражнении фильтр, перекомпилируем прием нажатием F3 и запускаем прокрутку решателя по рассматриваемому подразделу задачника. Анализируя результаты прокрутки, обнаруживаем, что 4 задачи изменили свои ответы, одна задача замедлилась более чем на 4 млн. шагов интерпретатора ЛОСа и три задачи - примерно на 1 млн. Суммарное замедление по разделу составило 9 млн. шагов. Чтобы подробнее рассмотреть причины изменений, нажимаем из диалогового блока просмотра статистики прокрутки клавишу "и". Попадаем на первую задачу с изменившимся ответом. Чтобы пропустить текущую задачу и перейти к следующей задаче списка задач с измененным ответом, будем каждый раз нажимать клавишу "ш". Однако, нужно помнить, что запуск решения любой задачи задачника сбрасывает буфер, хранящий список отобранных для просмотра задач. Тогда дальнейшие нажатия "ш" ничего не дадут, и для возобновления просмотра списка с начала нужно будет снова нажать "и" из диалогового блока.

   Продолжаем анализ первой задачи списка измененных ответов. В ней нужно было разложить на множители $a\sqrt{ba}-\sqrt{ba}-ba+a$. Исходный ответ имел вид $(-\sqrt{ab}+a)(\sqrt{ab}+1)$. Видно, что новый ответ более громоздкий и состоит из двух случаев: при $a\le 0, b\le 0$ имеем выражение $(-\sqrt{-a}\sqrt{-b}+a)(\sqrt{-a}\sqrt{-b}+1)$, иначе - $(\sqrt{a}\sqrt{b}+1)(-\sqrt{b}+\sqrt{a})\sqrt{a})$. Разбор случаев здесь позволил измельчить выражения под радикалами, однако неясно, насколько полезно такое измельчение. Чтобы выяснить, что давал отброшенный фильтр, восстанавливаем программу приема нажатием "Б", и запускаем найденные выше две задачи, в которых срабатывала его исходная версия. Выясняется, что в обоих случаях применение приема позволяло усмотреть полный квадрат в сумме под радикалом. Это и объясняет появление фильтра.

4. Список задач раздела "Элементарная алгебра", в которых срабатывает рассматриваемый прием, имеет четыре элемента. Чтобы найти те из них, где прием не нужен, выделяем первый элемент списка и нажимаем "й". Знак вопроса появляется около третьей задачи. В ней требуется решить уравнение $(15\sqrt{3}+26)^x-5(4\sqrt{3}+7)^x+6(\sqrt{3}+2)^x+(-\sqrt{3}+2)^x=5$. Чтобы найти задачи, которые перестают решаться при отключении приема, входим в просмотр его описания и нажимаем F7. Это уничтожает программу приема. При запуске решателя на отобранной в буфере четверке задач выясняется, что два ответа утеряны. Это первая и вторая задачи. В них требовалось решить уравнения $10^x-5^{x-1}2^{x-2}=950$, $a^{x+2}b^{x+3}=4(ab)^{2x}+1$. Последняя задача списка, хотя и замедляется немного (примерно на 20 процентов), все же остается решенной. Ее условие - неравенство $6\cdot 9^{1/x}-13\cdot 3^{1/x}\cdot 2^{1/x}+6\cdot 4^{1/x}\le 0$. Нам нужно найти какой-то простой признак, отличающий срабатывания приема в первых двух задачах от срабатываний в двух последних. Для этого переходим в просмотр приема, восстанавливаем его программу нажатием клавиши F5 и возвращаемся в буфер задачника. Используя отладчик ЛОСа, анализируем точки срабатывания приема в указанных задачах. При этом выясняется, что в первых двух случаях условие уже имело степень вида $(ad)^X$ с неизвестным показателем

X, а в последних двух - не имело. Поэтому создаем дополнительный фильтр "контекст(позиция(x6 корень)вид(x6 степень(x7 x8))не(известно(x8))равно(x7 терм(умножение(x1 x4))))". Он проверяет наличие в том же условии степени "степень(x7 x8)" с неизвестным показателем x8 и основанием, равным результату обработки нормализатором "нормумножение" выражения ad.

5. Создаем в буфере задачника список всех задач указанного раздела, в которых прием срабатывал. Оказывается, что в нем всего пять задач. Нажимая клавишу "й", выясняем, что во второй задаче срабатывание приема было избыточным. Входим в трассировку и анализируем точку срабатывания. Прием пытался выразить u через z из уравнения $uz = 9$, в то время как уже имелась дизъюнкция $u = 9z \ \vee u = z/9$. Очевидно, такая попытка не нужна. Можно было бы отсечь ее, добавив фильтр, запрещающий срабатывание приема при наличии дизъюнкции уравнений.

   Чтобы выяснить, к чему приведет удаление фильтра "конец(альтернатива(...))", перекомпилируем прием с отброшенным фильтром и запускаем прокрутку подраздела "Решение уравнений" (в других подразделах срабатывание приема маловероятно, хотя для полной гарантии можно было бы реализовать и их тестирование). Итоги прокрутки таковы: 8 задач изменили свои ответы, одна замедлилась на 10 млн. шагов работы интерпретатора и еще несколько замедлились на меньшие величины. Общее замедление составило 20 млн. шагов. Нажимая "и" из диалогового блока, просматриваем изменившиеся ответы. В нескольких случаях они чуть-чуть усложнились, в нескольких - были переставлены их элементы. Видимо, наиболее чувствительной точкой оказалась сильно замедлившаяся задача. Нажимаем "з" и находим ее условие: $-y\sqrt{y} + x\sqrt{x} = a(-\sqrt{y} + \sqrt{x}), x^2 + xy + y^2 = b^2$. Параметры a, b положительны. Входим в трассировку и обнаруживаем, что прием применяется для выражения u через z из уравнения $uz = (a + b)(a - b)/2a$. При этом имеется второе уравнение $(u + z)^2 = (3a^2 - b^2)/2a$. Очевидно, в такой ситуации лучше было бы выразить u через z из второго уравнения, предварительно возведенного в степень $1/2$.

13.2.13 Упражнения на повторное создание приема

В этом разделе будут представлены упражнения на самостоятельное создание приема, уже имеющегося в решателе. Чтобы предотвратить срабатывание "старой" версии приема, уровень срабатывания новой версии нужно выбрать равным 0. Примеры для тестирования работы приема создайте самостоятельно. По окончании выполнения упражнения новый прием следует удалить. В качестве указаний будут приведены ссылки на старые версии приемов.

1. Создать прием для решения квадратного уравнения $ax^2 + bx = c$.

2. Создать приемы для возведения в квадрат уравнения с двумя либо тремя радикалами, содержащими неизвестные.

3. Создать прием для логарифмирования показательного уравнения.

4. Создать прием, решающий простейшее тригонометрическое уравнение с синусом.

5. Создать прием, усматривающий и решающий квадратное уравнение относительно линейной комбинации синуса и косинуса неизвестного аргумента.

6. Создать прием, преобразующий сумму арксинусов к виду арккосинуса.

7. Создать прием, выписывающий соотношение пропорциональности длин отрезков, отсекаемых параллельными прямыми.

8. Создать прием, выписывающий соотношение для высоты треугольника и длин его сторон.

9. Создать приемы для равенства треугольников по двум сторонам и углу между ними.

10. Создать прием для вывода уравнения окружности, имеющей заданный центр и касающейся заданной прямой.

11. Создать прием, выполняющий умножение двух вещественных матриц.

12. Создать прием для вычисления предела по правилу Лопиталя.

13. Создать прием, сводящий вычисление неопределенного интеграла от суммы двух функций к вычислению интегралов от слагаемых.

14. Создать прием для вычисления неопределенного интеграла интегрированием по частям.

15. Создать прием для решения дифференциального уравнения с разделяющимися переменными.

Указания

В качестве указания будем приводить ссылку на прием в оглавлении базы приемов - путь в оглавлении к концевому пункту. При необходимости будем уточнять номер приема в списке приемов данного пункта.

1. "Элементарная алгебра" - "Степени" - "Решение уравнений" - "Решение квадратного уравнения". Последний прием серии приемов данного пункта.

2. "Элементарная алгебра" - "Степени" - "Решение уравнений" - "Уравнение с дробными степенями" - "Возведение в квадрат уравнения с двумя (тремя) неизвестными радикалами.

3. "Элементарная алгебра" - "Степени" - "Решение уравнений" - "Показательные уравнения" - "Логарифмирование показательных уравнений". Два последних приема серии приемов.

4. "Элементарная алгебра" - "Тригонометрия" - "Синус" - "Уравнения" - "Простейшее уравнение для синуса". Последний прием серии приемов.

5. "Элементарная алгебра" - "Тригонометрия" - "Синус" - "Уравнения" - "Квадратное уравнение относительно линейной комбинации синуса и косинуса".

6. "Элементарная алгебра" - "Тригонометрия" - "Арксинус" - "Сумма арксинусов".

7. "Элементарная геометрия" - "Параллельны" - "Пропорциональность отрезков, отсекаемых параллельными прямыми" - "Соотношение пропорциональности длин отрезков, отсекаемых параллельными прямыми".

8. "Элементарная геометрия" - "Фигуры"/ - "Треугольник" - "Высоты треугольника" - "Соотношение для высоты треугольника и длин его сторон".

9. "Элементарная геометрия" - "Фигуры"/ - "Треугольник" - "Признаки равенства треугольников" - "Равенство треугольников по двум сторонам и углу между ними".

10. "Аналитическая геометрия" - "Линии второго порядка" - "Окружность" - "Ввод уравнения окружности, имеющей заданный центр и касающейся заданной прямой".

11. "Линейная алгебра" - "Матрицы" - "Простейшие свойства матриц" - "Умножение матриц". Последний прием серии.

12. "Математический анализ" - "Вычисление пределов" - "Нормализатор НОРМПРЕДЕЛ" - "Предел дроби" - "Правило Лопиталя". В нашем упражнении нужно создать не прием нормализатора, а аналогичный прием сканирования задачи, срабатывающий на уровне 0.

13. "Математический анализ" - "Интегралы" - "Нормализатор нахождения первообразной "нормИнтеграл"¨ "Интеграл суммы". В упражнении нужно создать аналогичный прием сканирования задачи.

14. "Математический анализ" - "Интегралы" - "Нормализатор нахождения первообразной "нормИнтеграл"¨ "Интегрирование по частям". В упражнении нужно создать аналогичный прием сканирования задачи.

15. "Дифференциальные уравнения" - "Уравнения первого порядка" - "Уравнение с разделяющимися переменными" - "Интегрирование уравнения с разделяющимися переменными".

Глава 14

Логический ассемблер

Задание приемов на ГЕНОЛОГе - важный шаг к пониманию того, как приемы возникают из теоретических знаний. В этом задании явным образом присутствует теорема, сопровождаемая подробными инструкциями как и когда ее применять. Компилятор ГЕНОЛОГа, получая такие инструкции, однозначно синтезирует по ним программу приема на ЛОСе, и она сразу готова к употреблению.

Однако, необходимо делать следующий шаг, направленный на выявление источников приемов. Приведенное в предыдущих томах монографии описание приемов ГЕНОЛОГа, возникших при проработке множества различных предметных областей, демонстрирует огромное разнообразие их устройства. Чтобы автоматизировать синтез приемов, необходимо как-то уменьшить это разнообразие - классифицировать приемы так, чтобы для каждого типа приемов можно было предложить свой способ их создания. Наиболее естественным подходом представляется классификация приемов по целевой направленности. Вместо подробного описания того, как решатель должен использовать теорему приема, будет рассматриваться лишь указание на то, с какой целью данная теорема будет использоваться. Вся накопленная база приемов ГЕНОЛОГа тогда будет выступать в качестве обучающего материала, дающего ответ на вопрос, каким образом из сопровождающей теорему целевой установки извлечь подробное описание способа ее применения , т.е. прием ГЕНОЛОГа.

Классификация приемов по целевому признаку была предпринята, хотя бы частично, для ряда математических разделов решателя: алгебра логики и теория множеств, комбинаторика, элементарная алгебра, элементарная геометрия, аналитическая геометрия, линейная алгебра, математический анализ, комплексный анализ, теория вероятностей, общая алгебра. В некоторых разделах она охватывает значительную часть приемов, в других - лишь едва обозначена. Ничто не мешает продолжить ее на приемы нематематических разделов, например, такие, как физика и химия.

Целевая установка, сопровождающая теорему приема, складывается из логического символа, кодирующего тип приема, и нескольких (обычно одного - двух) сопровождающих этот тип данных. Такими данными могут служить указатель направления при применении тождества или эквивалентности, выделенная переменная (например, неизвестная), и т.п. В целом, ситуация напоминает язык ассемблера: тип приема является аналогом кода операции, дополнительные данные - аналогом операндов.

Так как имеется компилятор для преобразования в прием ГЕНОЛОГа теоремы приема, снабженной целевой установкой, то фактически мы имеем дело с языком задания приемов, уровень которого выше уровня ГЕНОЛОГа. Ввиду указанной выше аналогии с обычным ассемблером, этот язык получил название логического ассемблера,

а целевая установка названа спецификацией приема. Обычно спецификация приема совсем невелика по объему, в отличие от описания приема на ГЕНОЛОГе. Заметим, впрочем, что компилятор, преобразующий спецификацию в прием ГЕНОЛОГа, редко выдает окончательную версию приема, пригодную к использованию. Он дает лишь исходную версию, которая требует доводки на примерах и лишь после этого становится полноценным приемом. В этом отличие данного компилятора от обычных компиляторов: доводка на примерах становится частью компиляции. Такая ситуация продиктована чрезвычайно высоким уровнем языка.

Для развития логического ассемблера был создан несложный интерфейс, позволяющий пролистывать базу приемов и выполнять предварительную их классификацию - выявление типов приемов. Предпринят также предварительный цикл пролистывания этой классификации и ее уточнения. Многие созданные вручную приемы иногда оказывались слишком частными, иногда - неоправданно общими, а в ряде случаев и вовсе неадеквантными, так как задачу следовало решать совсем другими средствами. Поэтому, наряду с классификацией приемов, потребовалась существенная их переработка, направленная на исключение (по мере возможности) "одноразовых" типов приемов и повышение степени мотивированности срабатываний. Так как даже небольшое изменение приема, срабатывающего в десятках, а тем более сотнях задач, обычно влечет за собой необходимость переработки траекторий решения задач, пусть даже и не всех, трудоемкость работы по расчистке системы типов приемов оказалась весьма значительной. Впрочем, опыт показал, что эта расчистка вполне возможна (например, практически полностью были модернизированы приемы планиметрии), а после нее качество работы решателя повышается.

14.1 Предварительные сведения о логическом ассемблере

Оглавление типов приемов

Оглавление типов приемов представляет собой подоглавление оглавления программ. Перейти к его корню от корня оглавления программ можно вдоль пути "Синтез приемов" - "Оглавление типов приемов (Логический ассемблер)".

В концевых пунктах оглавления типов приемов располагаются краткие тексты, характеризующие тип приема. Чтобы определить логический символ, кодирующий тип приема, нужно в таком концевом пункте нажать клавишу "курсор вправо". Она вызовет переход в просмотр программы справочника "заголовокприема", причем логический символ, к которому относится эта программа, и есть код типа приема. Для возвращения в оглавление программ, как обычно, нажимается "End".

Совмещение оглавления типов приемов с оглавлением программ справочника "заголовокприема" имеет простое объяснение: данный справочник инициирует процесс создания приема ГЕНОЛОГа по его спецификации, и подробности этой инициализации можно сразу уточнить, выйдя на нужный пункт оглавления типов.

Чтобы просмотреть все приемы решателя, отнесенные к заданному типу, нужно, находясь в концевом пункте этого типа, нажать "Ctr-курсор вправо". На экране появится описание приема ГЕНОЛОГа - первого из серии приемов, отнесенных к заданному типу. Переход между приемами этой серии выполняется клавишами "курсор вверх" -

"курсор вниз". Если список приемов данного типа очень большой, то будут показаны лишь первые 50 приемов. Обычно этого достаточно.

Просматривая описание приема, следует понимать, что просмотр происходит не из оглавления приемов. Если в этот момент нажать "курсор влево", то произойдет возвращение к концевому пункту оглавления типов приемов. Для перехода в просмотр того же самого приема из оглавления ГЕНОЛОГа нужно нажать "Enter". Правда, тогда для возвращения в исходный концевой пункт оглавления типов придется сначала выйти в главное меню (например, нажатием "End"), а затем вернуться в оглавление программ нажатием "л".

При просмотре списка приемов, отнесенных к заданному типу, работают почти все стандартные средства просмотра описания приема ГЕНОЛОГа. Заблокированы попытки изменить прием и перейти к его ЛОС-программе.

В дополнение к стандартным средствам просмотра описания приема ГЕНОЛОГа, уже рассмотренным ранее, имеется возможность просмотра его спецификации. Для этого достаточно нажать клавишу "и". Описание приема на ГЕНОЛОГе будет спрятано, а непосредственно под теоремой приема появятся два других окна. В первом из них скопирован текст концевого пункта оглавления типов приемов, характеризующий данный тип. Во втором будут приведены прочие элементы спецификации - логические символы либо термы. Количество и смысл этих элементов зависят от типа приема. Далее, перечисляя типы, мы будем каждый раз уточнять необходимые сопровождающие их данные.

Для перехода от описания приема ГЕНОЛОГа, расположенного в базе приемов, к соответствующему пункту оглавления типов приемов, придется снова делать два шага - выйти в главное меню ("End") и нажать "л" для перехода в оглавление типов. Предварительно нужно нажать "и" для просмотра спецификации. Это настроит оглавление программ для правильного перехода.

Если при нажатии на "и" спецификация приема не прорисовывается, это означает, что прием пока не отнесен к какому-либо типу.

Отнесение созданного вручную приема ГЕНОЛОГа к тому или иному типу выполняется следующим образом. Из просмотра приема нажимается "х" (кир.). После этого на экране возникает корневая вершина оглавления типов приемов. В этом оглавлении следует выбрать нужный концевой пункт и нажать на нем "курсор вправо". Сразу же вслед за этим произойдет возвращение к просмотру описания приема, в котором будут прорисованы: теорема, под ней - текст выбранного концевого пункта оглавления типов, и ниже - список дополнительных элементов спецификации, созданных автоматически. Для некоторых типов приемов такие элементы автоматически не создаются или создаются лишь частично. В обоих случаях в начале списка дополнительных элементов (возможно, пустого) располагается курсор текстового редактора, так что этот список можно сразу же скорректировать. В конце редактирования нажимается "Enter", после чего спецификация приема уже создана. Автоматически произойдет возвращение к просмотру описания приема на ГЕНОЛОГе.

Эту же процедуру следует проделать, если нужно изменить ранее введенный тип приема. Как только в оглавлении типов будет нажата клавиша "курсор вправо" на концевом пункте, вся информация о ранее созданной спецификации будет безвозвратно утеряна.

Если тип приема желательно сохранить, а изменить лишь дополнительные элементы спецификации, нажимается "и" и выполняется необходимая коррекция.

Напомним, что вся информация о приемах ГЕНОЛОГа сохраняется в 8-м информационном блоке решателя (см. первый том монографии). В дереве номеров узлов этого блока концевые вершины суть узлы теорем приемов. Если прием имеет заголовок P, то от узла теоремы по меткам "команды", "P" имеет место переход к узлу приема. Спецификация приема сохраняется в логическом терминале "примечание" узла приема. Она представляет собой набор термов и логических символов, в котором выделяется элемент "тип(T)", где T - логический символ, являющийся типом приема.

В процессе обучения системы могут исключаться старые приемы, появляться новые, изменяться тип ранее созданного приема. Поэтому ссылки из оглавления типов на приемы, имеющие данный тип, нужно время от времени обновлять. Для этого, зайдя в оглавление программ (не базы приемов!), нужно нажать "Ctr-тильда". Все старые ссылки будут отброшены, и после просмотра базы приемов создадутся новые. Следует заметить, что этот процесс занимает несколько минут. Проверить, что система не зависла, можно нажатием "Break". Если появляется кадр отладчика ЛОСа, то все в порядке, и процесс может быть продолжен нажатием клавиш "0" и "Enter". По окончании система выйдет в главное меню.

Если прием ГЕНОЛОГа сопровожден спецификацией, то можно получить его альтернативную версию, созданную компилятором спецификаций. Для этого из просмотра приема нажимается "г". Появится новая версия приема, отделенная снизу голубой чертой. Можно некоторое время переходить от просмотра этой версии к просмотру старой версии и обратно, нажимая "р". Чтобы фактически поменять старую версию на новую, нужно при просмотре старой версии нажать "Ctr-ъ". Делать это следует осмотрительно, так как обычно компилятор спецификаций создает лишь исходную версию приема, требующую доводки на примерах. В сомнительных случаях рекомендуется устроить прокрутку по задачнику для выявления тех точек, где решение сильно замедлилось либо было утеряно. Анализ этих точек позволит либо уточнить прием вручную, либо попытаться продолжить обучение компилятора спецификаций, чтобы в будущем он проявлял большую предусмотрительность. Заметим, что при нажатии "Ctr-ъ" старая версия полностью удаляется, даже без сохранения в буфере. Новая версия не только замещает ее, но и компилируется в программу ЛОСа.

Справочник "заголовокприема"

Как уже говорилось выше, справочник "заголовокприема" инициирует работу компилятора спецификаций. Так как его программы соответствуют типам приемов, то целесообразно совместить описание этих типов с описанием программ справочника. Поэтому, прежде чем перейти к подробному ознакомлению с логическим ассемблером, сообщим основные сведения о данном справочнике.

Входными данными справочника служат: текущий логический символ - тип приема; x1 - теорема приема; x2 - спецификация приема; x3 - вспомогательная структура данных компилятора спецификаций, называемая блоком приема. Справочник синтезирует описания приемов согласно x1 и x2 (их может оказаться несколько) и региструет в накопителе (приемы ...) блока приемов (см. ниже).

Каждая программа справочника "заголовокприема" создает лишь ту часть описания приема ГЕНОЛОГа, для которой необходимо знание типа приема. Затем действия

всех этих программ объединяются в один общий поток компилятора спецификаций, который будет рассмотрен ниже. Объединение происходит в результате обращения всех таких программ к процедуре "схемапосылок".

Перечислим основные информационные элементы, хранящиеся в блоке приема. Некоторые из них ("приемы", "теорема") вводятся в блок приема еще до обращения к справочнику "заголовокприема", другие ("заголовок", "условие", "прием", "примечание") создаются этим справочником, третьи появляются на более поздних этапах компиляции.

1. (заголовок A) - A есть заголовок приема.
2. (условие A) - A есть набор фильтров создаваемого приема ГЕНОЛОГа.
3. (прием A) - A есть набор элементов описания создаваемого приема ГЕНОЛОГа, за исключение элемента "условие(...)", перечисляющего фильтры.
4. (приемы A) - A есть накопитель четверок (теорема приема - заголовок приема - спецификация приема - описание приема) для созданных приемов ГЕНОЛОГа. Набор (приемы пустоеслово) должен быть создан еще при обращении к справочнику "заголовокприема". Его следует зарегистрировать в какой-либо программной переменной. Справочник не изменяет сам набор, а только его второй элемент. Поэтому, после обращения к справочнику, можно рассмотреть значение указанной программной переменной и определить созданные приемы.
5. (примечание A) - A есть спецификация приема.
6. (теорема A_1 A_2) - пара A_1, A_2 есть ссылка на теорему из базы теорем (т.е. логический символ и номер узла этого символа), являющуюся источником создаваемого приема.
7. (квантор A) - A есть тип квантора ("длялюбого", "существует"), вводимого приемом кванторной расшифровки.
8. (антецедент A_1 A_2) - A_1 есть набор вхождений корней антецедентов теоремы, A_2 - соответствующий набор указателей типов их обработки. Если антецедент непосредственно идентифицируемый, то на соответствующей позиции в A_2 стоит 0.
9. (текприем A_1 A_2 A_3) - ссылка на прием, для которого происходит тестирование его синтеза либо доопределение. Здесь A_1 - логический симввол, A_2 - номер узла теоремы приема в ветви этого символа, A_3 - заголовок приема.
10. (идентификатор A) - A есть список вхождений в теорему, подлежащих идентификации.
11. (нормализатор A) - A есть список вхождений в теорему, а также в фильтры и указатели приема, на которых расположены новые термы, создаваемые приемом.
12. (заменазнака X A) - A есть тот внешний знак, который при идентификации может быть передан переменной X.
13. (смтеор A) - A есть теорема приема.

Подробнее с назначением элементов блока приема можно ознакомиться в описании компилятора спецификаций, приведенном в 7 томе монографии "Компьютерное моделирование логических процессов". Пока нам понадобится лишь та их часть, которая создается справочником "заголовокприема". Как правило, это лишь элементы (заголовок A), (условие B) и (прием C). Перечисляя типы приемов, будем давать также краткое описание соответствующих программ справочника. При этом элементы набора B будем называть фильтрами, элементы набора C - указателями.

14.2 Логический ассемблер

Предлагаемая классификация приемов является результатом простейшей их сортировки по целевому признаку. Во многих случаях вся информация о типе приема сосредотачивалась лишь в словесном описании преследуемой им цели, а справочник "заголовокприема" был, по существу, пустой заглушкой. Однако, по мере развития системы автоматического синтеза приемов, появилось множество типов приемов, хорошо укомплектованных средствами их автоматического создания. Уже сейчас этот аппарат работает в циклах автоматического анализа теорем и порождает множество приемов, вполне достойных занесения в основную базу приемов. Подробнее об этих циклах будет рассказано в следующем томе монографии. При этом требуется продолжение работы по расчистке системы типов. Многие из них вводились ради единственного приема. Если такой прием заменить чем-то другим, более стандартным, то надобность в его типе отпадает, и он может быть удален. Иногда так и удается поступить. Вместе с тем, для некоторых "одноразовых" типов складывается впечатление о их необходимости.

Типы приемов часто пересекаются друг с другом и даже поглощают друг друга. Это объясняется различием в степени мотивированности их срабатываний. Для тех типов приемов, у которых степень мотивированности выше, можно предлагать меньший уровень срабатывания. Иногда по одной и той же теореме целесообразно создавать несколько различных приемов, имеющих одинаковую целевую направленность, но различные степени мотивированности и различные уровни срабатывания. Поэтом не следует удивляться определенной избыточности приводимой системы типов приемов. Она помогает точнее адаптироваться к конкретной ситуации, в том числе и при автоматическом создании приема. Кроме того, могут показаться заниженными количества приемов того или иного типа. Это следствие того, что близкие действия по тем или иным причинам были отнесены при обучении решателя к приемам других типов.

Как уже говорилось, далеко не все типы приемов сейчас подключены к системе автоматического их создания, хотя формально какие-то программы справочника "заголовокприема" были созданы, чтобы зарегистрировать тип в оглавлении. В этих случаях ниже будем констатировать, что справочник "заголовокприема" не прорабатывался.

На самом деле почти для всех типов приемов их автоматический синтез сейчас дает лишь часть необходимых фильтров и указателей. С другой стороны, логика происхождения элементов описания приема, которые пока не создаются автоматически, обычно легко прослеживается. Это позволяет продолжать обучение генератора приемов, постепенно устраняя всевозможные его пробелы. Интересно, что уже реализованный полный цикл автоматического создания приемов, несмотря на все его

недостатки, способен порождать десятки вполне работоспособных приемов, сопровожденных доказывающими их необходимость тестовыми задачами и прошедших проверку путем полной прокрутки по задачнику.

Чтобы сделать классификацию хоть сколь-нибудь понятной, приводимые типы приемов придется сопровождать примерами самих приемов. Будем также указывать число приемов заданного типа, созданных на момент написания книги. Заметим, что для многих приемов их тип не указан, и они в данной статистике не учтены.

Хотя элементы спецификации индивидуальны для каждого типа приемов, имеется ряд элементов, общих для всех типов:

1. тип(A) - A есть тип приема.
2. направл(A) - A есть направление тождественной либо эквивалентной замены (символ "первыйтерм" либо "второйтерм").
3. см(A) - A есть список фильтров приема, известных на момент создания его теоремы.
4. указатель(A) - A есть список указателей приема, известных на момент создания его теоремы.
5. быстрпреобр(A_1, A_2) - A_2 есть список нормализаторов, которыми должен обрабатываться подтерм A_1.

Будем следовать оглавлению типов приемов, достижимому из корня оглавления программ по пути "Синтез приемов" - "Оглавление типов приемов (Логический ассемблер)". Приведем лишь сокращенный список типов, отбрасывая все те, для которых число приемов решателя менее 10. Впрочем, пока не все приемы ГЕНОЛОГа связаны с логическим ассемблером, и реальное количество приемов того или иного типа может оказаться большим. Полный список типов приемов можно найти в 7 томе монографии "Компьютерное моделирование логических процессов".

14.2.1 Справочники и сопровождающие их простейшие приемы

Автоматический синтез приемов справочников несложен. Все компоненты приема однозначно определяются по его теореме и заголовку справочника. В основном, типами приемов справочников служат всего два логических символа - "справка" и "короче". Первый из них относится к справочникам "общего характера" (например, арность, тип значения, о.д.з и т.п.), второй - к так называемым справочникам поиска теорем. Они обеспечивают нахождение нужных теорем в базе теорем и будут рассмотрены в следующем томе. Таким образом, спецификация приема справочника практически вырожденная - как правило, состоит лишь из элемента "тип(справка)". Поэтому она не сохраняется в структурах данных, сопровождающих прием ГЕНОЛОГа, и не прорисовывается интерфейсом просмотра спецификаций. Впрочем, при автоматическом создании приемов справочников действия те же, что и для остальных приемов - сначала создается спецификация, потом она преобразуется в описание приема ГЕНОЛОГа.

Для автоматического создания приемов типа "справка" создан альтернативный интерфейс. В оглавлении ГЕНОЛОГа вводится новый концевой пункт, и в нем текстовым редатором набирается теорема приема. Например, "коммутативно(плюс)". Далее нажимается "а", и на экране возникает корневое меню оглавления типов приемов. В нем нажатием "курсор вправо" выбирается первый пункт - "Справочники и сопровождающие их простейшие приемы". Сразу после этого создается спецификация "тип(справка)" и происходит обращение к программе "заголовокприема" на символе "справка". Эта программа сравнительно большая и создает сразу все приемы справочников, а иногда и простейшие приемы нормализаторов или приемы сканирования задач, которые могут быть созданы по введенной теореме. В случае теоремы "коммутативно(плюс)" будут созданы прием справочника "коммутатвно", прием нормализатора "нормплюс", выполняющий лексикографическое упорядочение операндов, прием лексикографического упорядочения операндов в термах задач, и приемы устранения вложенных сумм - в нормализаторе "нормплюс" и в термах задач. На экран будет выдано описание первого из предлагаемых приемов, под которым проведена голубая черта. Это означает, что пока прием нигде не зарегистрирован. Как и обычно, для его регистрации без компиляции нажимается F4, для регистрации и компиляции - F3. Чтобы перейти к следующему приему (в частности, пропустив регистрацию ненужного приема), нажимается "ш". Если следующего приема нет, программа на нажатие "ш" не реагирует. Для выхода из цикла создания приемов достаточно нажать Esc. Если ранее прием справочника уже был создан, его описание будет предложено, но повторной компиляции не произойдет - снизу появится красная линия.

Программа справочника "заголовокприема" на символе "справка" отличается от всех прочих программ этого справочника. Она не обращается к компилятору спецификаций, а сразу создает описание приема и регистрирует его в накопителе (приемы ...) блока приема. По существу, это вырожденный случай, и мы его рассматривать не будем.

Лишь малая часть приемов справочников (кроме справочников поиска теорем) в настоящее время создается через справочник "заголовокприема". Обычно их описания на ГЕНОЛОГе вводятся вручную. Это происходит не из-за сложности автоматического синтеза, а просто вводу отсутствия такой необходимости. Впрочем, приемы справочников формульного редактора, сопровождающие вновь вводимые понятия, удобно создавать указанным выше полуавтоматическим способом. По мере развития средств автоматического синтеза приемов по теоремам базы теорем, нетрудно будет полностью автоматизировать и синтез приемов справочников.

14.2.2 Приемы тождественной замены

Начиная с этого места, идут "обычные" типы приемов, спецификация которых связывается с их описанием. Комментируя действия справочника "заголовокприема", начинающего цепочку шагов по синтезу приема ГЕНОЛОГа, будем по умолчанию понимать, что этот справочник всегда обращается к процедуре "схемапосылок", продолжающей данную цепочку. Она будет описана в главе, посвященной компилятору спецификаций. В большинстве случаев работу справочника "заголовокприема" не будем описывать в общем виде, а лишь указывать, какие фильтры и указатели он создает для рассматриваемого примера приема.

Приемы общей стандартизации

Понятие "общая стандартизация" достаточно условное. В широком смысле оно охватывает все приемы тождественной замены, которые применяются без дополнительных ограничений. Иногда это решение может казаться спорным, но все-таки на текущий момент примеров, требующих создания таких ограничений, не возникало. Однако, можно выделить подмножество приемов без дополнительных ограничений, которые вряд ли когда-либо будут как-то ограничены. Они и называются далее приемами общей стандартизации. Это выделение сугубо эвристическое и опирается на эвристическую же характеризацию теорем, рассматриваемую в следующем томе. Однако, данная характеризация позволяет системе самостоятельно выделять случаи общей стандартизации и создавать для них приемы.

1. Общая стандартизация.

   Теорема приема представляет собой тождество, применяемое для общей стандартизации выражений в любых ситуациях. Заменяемая и заменяющая части тождества не имеют связанных переменных. Примеры приемов данного типа:

   $\forall_{ab}(a \subseteq b \to a \cup b = b)$

   $\forall_{a}(a + 0 = a)$

   $\forall_{a}(\log_a(a) = 1)$

   Спецификация приема имеет вид "тип(общнорм)", "направл(N)". Здесь N указывает направление замены. Если N - символ "первыйтерм", то замена происходит справа налево; если символ "второйтерм", то слева направо. Справочник "заголовокприема" определяет заголовок приема "N" и вводит единственный фильтр - "уровень(u)", где $u = 0$, если теорема приема не имеет посылок, не являющихся условиями на о.д.з. заменяемой части (такие посылки называются существенными), и $u = 1$, если существенные посылки имеются. Если заменяющий и заменяемый термы различаются лишь внутри некоторого своего собственного подтерма, то указатель приема "нормализатор" не вводится, иначе - вводится. Напомним, что такой указатель инициирует цикл попыток немедленной стандартизации надтермов преобразованного терма. Если заменяемый терм имеет единственную переменную, то вводится также указатель "сопровождение". Он разрешает преобразование даже в сопровождающих по о.д.з. утверждениях задачи.

   Число приемов данного типа - 1045.

2. Общая стандартизация выражения, использующая явно идентифицированный антецедент для подмножества операндов ассоциативно-коммутативной операции.

   Теорема приема имеет существенный антецедент, содержащий переменную x, являющуюся операндом ассоциативно-коммутативной операции F в заменяемом выражении. Этот антецедент идентифицируется с утверждением из контекста, подсказывающим, с каким именно подмножеством операндов операции F следует идентифицировать данную переменную. Без такой подсказки пришлось бы идентифицировать ее с отдельным операндом, и это был бы предыдущий тип приема. Пример приема данного типа:

   $\forall_{ab}(a \subseteq b \to a \cup b = b)$

Здесь имеется два варианта выбора переменной - a либо b. В первом случае спецификация приема имела бы вид "тип(уникопия)", "направл(второйтерм)", "переменная(a)", во втором - "тип(уникопия)", "направл(второйтерм)", "переменная(b)". Справочник "заголовокприема" создает прием с заголовком, равным указателю направления замены и фильтрами "уровень(1)", "заголовок(x, F)", где x - рассматриваемая переменная, F - символ операции, операндом которой она является в заменяемом терме. Если заменяемый терм имеет единственную переменную, то вводится указатель "сопровождение".

Число приемов данного типа - 5.

3. Лексикографическая стандартизация.

Теорема приема имеет вид тождества, у которого левая и правая части примерно равноценны, а замена выполняется лишь для лексикографической стандартизации: большее в лексикографическом порядке выражение заменяется на меньшее. При этом сравниваются лишь самые сложные в смысле справочника "оценка" подвыражения заменяемого и заменяющего термов. Необходимость лексикографической стандартизации очевидна: таким образом подготавливается усмотрение совпадающих подтермов в различных частях задачи. При редактировании уже найденного ответа возможны встречные замены.

Справочник "оценка" часто будет возникать в связи с синтезом приемов. Он получает на вход в качестве значения переменной x1 вхождение выражения, заголовком которого служит текущий логический символ. Выдается эвристическая оценка сложности данного выражения в формате десятичного числа. Фактически, оценивается сложность текущего символа, но делается это с учетом контекста x1. Приемы данного справочника пока вводятся вручную, хотя логика определения эвристических оценок для новых понятий несложна. Обычно такая оценка немного превосходит максимум оценок тех понятий, через которые определяется данное понятие.

Типичным примером служит следующий прием:

$\forall_{ab}(\sin(b-a) = -\sin(a-b))$

Проверяется лексикографическое предшествование выражения $\sin(a-b)$ выражению $\sin(b-a)$. Прием имеет фильтры, блокирующие его применение на этапе редактирования ответа либо уменьшения длины (свертки) условия, так как в этих случаях работает обратный прием, устраняющий внешний минус.

Спецификация приема имеет вид "тип(цветпункта)", "направл(N)", где N - направление замены. Справочник "заголовокприема" прежде всего сравнивает заменяемый и заменяющий термы, отбрасывая у заменяющего терма внешнюю одноместную операцию, если она есть и заменяемый терм короче заменяющего. Проверяется, что после этого оба терма имеют равные длины. Затем внутри них находятся минимальные подтермы t_1, t_2, в которых эти термы различаются. Создаются следующие фильтры:

(a) "уровень(3)".

(b) "лексикопредшествует(t_2, t_1)".

(c) Если заменяемый терм короче заменяющего, то добавляются фильтры "или(не(тип(описать)) не(цель(редакция)))", "или(не(тип(преобразовать)) и(коммент(длина) не(цель(длина))))".

(d) Если в заменяющем терме глубина некоторой переменной x, вычисленная относительно более чем одноместных операций, больше ее глубины в заменяемом терме, то вводится фильтр "или(не(тип(описать))посылка известно(x))".

Число приемов данного типа - 11.

Приемы преобразования описателей

1. Исключение описателя "класс".

   (a) Исключение описателя "класс".

   Теорема приема имеет вид тождества, у которого заголовком заменяемой части служит символ "класс", а заменяющая часть не имеет связанных переменных. Фактически, это особый случай общей стандартизации, так как обычно выгодно от описателя избавиться. Обратный переход делается, когда некоторое надвыражение либо надутверждение допускает упрощение после расшифровки через описатель. Впрочем, как правило для этого используются заранее созданные приемы, исключающие промежуточный шаг расшифровки.

   Пример теоремы приема:

   $\forall_{Af}(\mathrm{set}_x(x \in \mathrm{Dom}(f)\ \&\ f(x) \in A) = \text{прообраз}(f, A))$

   Спецификация приема имеет вид "тип(цепьвоглавлении)", "направл(N)". Справочник "заголовокприема" определяет лишь уровень срабатывания, равный 1. Этого достаточно, чтобы компилятор спецификаций создал полноценный прием.

   Число приемов данного типа - 11. Впрочем, в действительности их несколько больше - часть таких приемов была первоначально охарактеризована как приемы общей стандартизации, а коррекция типа пока не выполнена.

2. Стандартизация описателя "класс".

   Приемы специальной стандартизации почти не проработаны. Пока предпринята лишь их предварительная классификация. Это объясняется тем, что решение о применении в том или ином контексте специальной стандартизации принимается не в базе приемов, а в базе теорем, путем анализа имеющихся теорем рассматриваемого ее раздела. Накопленный в базе приемов материал позволяет перейти к работе над обобщением частных случаев специальной стандартизации и автоматизации выработки решений о ней, однако такая работа лишь предстоит.

   (a) Стандартизация описателя "класс".

   Многие приемы, работающие с множествами, определенными через описатель "класс", требуют предварительной стандартизации утверждений под этим описателем. Например, в аналитической геометрии уравнения кривых или поверхностей представляются в виде равенства множества координат их точек описателю класс, под которым располагается равенство нулю некоторого выражения. Это выражение должно иметь вид многочлена от координат, т.е. предварительная стандартизация будет исключать

дроби, группировать все ненулевые члены в одной части и при необходимости изменять знаки всех слагаемых. Стандартизация необходима, так как большинство приемов, использующих уравнения кривых или поверхностей, предполагают работу именно с многочленом от координат.

Таким образом, мы сталкиваемся с целевой установкой, заключающейся в подготовке возможности применения других приемов. Таких типов приемов будет еще много. Здесь же подготовка связана со стандартизирующим эквивалентным преобразованием утверждений под описателем "класс". Пример приема этого типа:

$\forall_{abc}(\neg(b=0) \to \mathrm{set}_x(a/b+c=0 \ \&\ A(x)) = \mathrm{set}_x(a+bc=0 \ \&\ A(x)))$

Предпринимается исключение дробных слагаемых в уравнении под описателем.

Спецификация приема имеет вид "тип(окружность)", "направл(N)", "указатель(...)", "см(...)". Последние два элемента перечисляют фильтры и указатели приема, известные уже на момент создания его теоремы. Они фиксируют конкретный контекст, в котором предпринимается стандартизация, и конкретный способ стандартизации. Эти сведения выходят за рамки компетенции логического ассемблера и связаны с программирующим логическим выводом в базе теорем. Именно там, путем анализа имеющихся в предметной области теорем, должны приниматься решения о целесообразности той или иной специальной стандартизации, и создаваться теоремы для такой стандартизации. Принятые решения фиксируются в так называемых протоколах базы теорем, располагаемых в ней наравне с обычными теоремами. Подробнее об этом будет говориться в следующем томе монографии.

Справочник "заголовокприема" не прорабатывался - пока указывает лишь уровень срабатывания 1.

Число приемов данного типа - 24.

(b) Переход от параметрического задания класса к непосредственному.

Параметрическое задание класса имеет вид $\mathrm{set}_x(\exists_y(x = F(y) \ \&\ A(y)))$. Здесь y - параметр либо набор параметров, участвующих в перечислении элементов. В случае непосредственного задания класса под описателем располагается утверждение без связанных переменных. Переход от одного способа задания класса к другому не является общей стандартизацией. В одних ситуациях полезен один способ, в других - другой. Тем не менее, в определенных контекстах такой переход может рассматриваться как стандартизирующий. Определение этих контекстов - дело процедуры, анализирующей базу теорем. Здесь же, в логическом ассемблере, фиксируем два типа приемов для указанных переходов. Начнем с перехода от параметрического задания класса к обычному. Пример - прием, выполняющий переход от параметрического уравения прямой к обычному:

$\forall_{abcd}(\neg((a,c)=(0,0)) \to \mathrm{set}_{xy}(\exists_t(x=at+b \ \&\ y=ct+d \ \&\ t-\text{число})) = \mathrm{set}_{xy}(cx-ay+ad-bc=0 \ \&\ x-\text{число} \ \&\ y-\text{число}))$

В этом случае прием применяется почти без ограничений (при отсутствии цели, указывающей на необходимость именно параметрического задания), так что компилятор спецификаций будет создавать его правильно.

Спецификация приема имеет вид "тип(константа)", "направл(N)". Могут

добавляться элементы "указатель", "см". Справочник "заголовокприема" определяет уровень срабатывания 1, вводит фильтр "или(тип(доказать) и(не(цель(известны)) не(цель(учетрезультата)) не(цель(вспомпараметр))))" и указатель "заменавхождений". Однако, по существу он не прорабатывался.

Число приемов данного типа - 7.

3. Группировка под описатель "класс".

   Теоретико-множественное выражение, содержащее один или несколько описателей "класс", иногда бывает удобно преобразовать к единственному описателю "класс". Этим достигается либо немедленное сильное упрощение, либо создаются условия для такого упрощения, проистекающие из того, что ранее разбросанные по разным описателям логические условия оказываются собраны в одном контексте. Примеры приемов такого типа:

   $\forall_{an}(\bigcup_{i=1}^{n} \mathrm{set}_x(\text{двнабор}(x,n)\ \&\ x(i)=a) = \mathrm{set}_x(\text{двнабор}(x,n)\ \&\ 0 < \text{колич}(x,a)))$

   Конечное объединение выделено указателем "развертка" и идентифицируется с обычным объединением.

   $\forall_{AB}(\mathrm{set}_x(A(x)) \cap \mathrm{set}_y(B(y)) = \mathrm{set}_x(A(x)\ \&\ B(x)))$

   Предварительно проверяется, что утверждения $A(x), B(x)$ не имеют заголовка "существует".

   $\forall_P(N\ \mathrm{set}_x(x - \text{натуральное}\ \&\ P(x)) = \mathrm{set}_x(x - \text{натуральное}\ \&\ \neg(P(x))))$

   Как правило, в созданных приемах такого типа дополнительные фильтры отсутствуют. Спецификация приема имеет вид "тип(транзитоперанд)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 4. Пока он не прорабатывался.

4. Переход от параметрического задания класса к операции над семейством.

   Так как операции над семействами множеств обычно определяются с помощью описателя "класс", появляется возможность обратного перехода - исключать описатель "класс", переходя от него к операции над семейством. Это позволяет воспользоваться многочисленными приемами, созданными для работы с такими операциями. Примеры:

   $\forall_{cf}(\neg(\mathrm{set}_x c(x) = \emptyset) \to \mathrm{set}_x(\forall_x(c(x) \to a \in f(x))) = \bigcap_{x,c(x)} f(x))$

   $\forall_{fgh}(\mathrm{set}_x(x - \text{число}\ \&\ \exists_n(f(n)\ \&\ g(n) < x\ \&\ x < h(n))) = \bigcup_{n,f(n)}(g(n), h(n)))$

   Оба приема применяются без ограничений. Спецификация приема имеет вид "тип(нормуравнение)", "направл(N)". Справочник "заголовокприема" ограничивается тем, что указывает уровень срабатывания 1. Этого оказывается достаточно для создания полноценного приема.

   Число приемов данного типа - 7.

5. Упрощение выражения под описателем относительно варьируемой переменной.

   Если варьируемая переменная имеет несколько вхождений в выражении, определяющем значение функции, то целесообразна попытка уменьшить количество этих вхождений, в идеале - свести все к единственному вхождению. Такое

преобразование может позволить усмотреть ту или иную стандартную функциональную зависимость и воспользоваться имеющимися для нее приемами. Пример:

$\forall_{abcdefghi}(ha^{dg+f}/ie^{bg+c} = ha^f(a^d/e^b)^g/ie^c)$

Преобразуемое выражение расположено под описателем "отображение" либо "класс", причем выражения a, b, c, d, e, f не содержат переменных связывающей приставки этого описателя, а выражение g - содержит. Таким образом усматривается экспоненциальная зависимость.

Спецификация приема имеет вид "тип(измзнака)", "направл(N)", "переменная(x)", где x - переменная, идентифицируемая с выражением, содержащим переменные связывающей приставки. Справочник "заголовокприема" указывает уровень срабатывания 1 и создает фильтры, проверяющие независимость от связывающей приставки выражений, идентифицируемых с переменными, отличными от x. Вводятся указатели, определяющие дополнительную идентификацию описателя и уточняющие идентификацию x. В нашем примере указатели приема следующие:

"контекст(подчинено(теквхожд x10) символ(x10 класс отображение))", "перечень(x7 пересекаются(x7 связприставка(x10)))".

Фильтры приема: "уровень(1)", "не(заголовок(x7 1))", "не(пересекаются(x1 связприставка(x10)))", "не(пересекаются(x2 связприставка(x10)))","не(пересекаются(x3 связприставка(x10)))","не(пересекаются(x4 связприставка(x10)))","не(пересекаются(x5 связприставка(x10)))","не(пересекаются(x6 связприставка(x10)))".

Число приемов данного типа - 9. Хотя оно и невелико, но имеется много тождеств, уменьшающих число вхождений неизвестной, и все они способны породить приемы данного типа. Генератор приемов создает их вполне приемлемым образом.

6. Определение характеристики отображения.

   (a) Непосредственное определение характеристики отображения.

   Приемы данного типа преобразуют более чем одноместную операцию, одним из операндов которой служит описатель "отображение", в выражение без описателей. Пока с этому типу были отнесены лишь приемы, вычисляющие образ отображения. Например:

   $\forall_{abfn}(a < 0 \ \& \ 0 < b \ \& \ n - \text{rational} \ \& \ \text{числитель}(n) - \text{even} \ \& \ 0 < n \rightarrow \text{образ}(\lambda_x(x^n, f(x)), (a, b)) = [0, \max(a^n, b^n)))$

   Спецификация приема имеет вид "тип(усмпростое)", "направл(N)". Так как обычно приемы применяются без ограничений, справочник "заголовокприема" ограничивается указанием уровня срабатывания 1.

   Число приемов данного типа - 40.

   (b) Операции над семействами.

      i. Непосредственное вычисление операции над семейством.

A. Непосредственное вычисление операции над семейством.
Приемы данного типа преобразуют одноместную операцию над описателем "отображение" в выражение без описателей "отображение". Примеры:

$\forall_{mnk}(m - \text{целое } \& \ n - \text{целое } \& \ k - \text{целое} \rightarrow \sum_{i=0}^{k}(C_n^i C_m^{k-i}) = C_{m+n}^k)$

$\forall_{af}(\prod_{x,f(x)} a = a^{\text{card}(\text{set}_{xf(x)})})$

Спецификация приема имеет вид "тип(эллипсоид)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1.

Число приемов данного типа - 37.

B. Непосредственное вычисление операции над семейством, использующее развертку в заменяющем терме.
В некоторых случаях вычисление операции над семейством можно свести к операции над семейством, отличающемся от данного в конечном числе точек. Тогда в заменяющем терме появляются корректирующие члены, имеющие вид операций над конечными семействами и допускающие развертку в обычные операции. Например:

$\forall_{abcdfn}(0 \le c \ \& \ 0 \le f \ \& \ f = n - d \ \& \ n - \text{натуральное} \rightarrow$
$\sum_{k=c}^{d}(a^k b^{n-k} C_n^k) = (a+b)^n - \sum_{g=0}^{c-1}(a^g b^{n-g} C_n^g) - \sum_{g=0}^{f-1}(a^{n-g} b^g C_n^g))$

Здесь проверяется, что c, f - десятичные константы, и конечные суммы в заменяющей части разворачиваются в обычные суммы.

Спецификация приема имеет вид "тип(названиесимвола)", "направл(N)". К ней могут добавляться элементы "см(целое(...))", указывающие, какие именно переменные должны идентифицироваться с целочисленными константами. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит указатели "развертка(...)" для операций над семействами в заменяющей части, подлежащими развертке.

Число приемов данного типа - 9.

C. Шаг развертки операции над конечным семейством в обычную операцию.
Приемы этого типа выносят из-под операции над конечным семейством единственный член, одновременно обращаясь к нормализатору, выносящему все остальные члены. Предполагается, что область действия операции задана либо в виде дизъюнкции равенств для варьируемой переменной, либо в виде условия принадлежности ее перечню. Примеры:

$\forall_{PQaf}(Q(a) \rightarrow \sum_{i,i=a \ \vee \ P(i),Q(i)} f(i) = f(a) + \sum_{i,P(i),Q(i)} f(i))$

Проверяется, что P(i) - дизъюнкция равенств, фиксирующих значение i. Затем проверяется, что точка a удовлетворяет условию суммирования Q, и член $f(a)$ выносится наружу.

$\forall_{Pabf}(P(a) \ \& \ \neg(a \in \{;b\}) \rightarrow \bigcup_{i,i\in\{a;b\},P(i)} f(i) = f(a) \cup \bigcup_{i,i\in\{;b\},P(i)} f(i))$

Спецификация приема имеет вид "тип(раскрытьскобки)", "направл(N)", "указатель(очевидно(1))", "указатель(единица(истина R))", где R - дополнительное ограничение на область действия операции(в первом примере - Q, во втором - P). Справочник "заголовокприема" указывает уровень срабатывания 0. Для приемов первого типа он также вводит фильтр "не(контекст(вид(фикс(0 1 1 2 1)или(x2 x3))не(контекст(вид(x2 равно(x9 x4))))единица(истина x3)))", проверяющий, что все дизъюнктивные члены суть равенства, фиксирующие значение переменной i.

Число приемов данного типа - 7.

D. Развертка операции над конечным семейством.

В этом случае преобразование операции над конечным семейством в обычную операцию над элементами семейства происходит за один шаг, с использованием указателя приема "развертка". Примеры:

$\forall_{ab}(\mathrm{Val}(a) = \{;b\}\ \&\ l(b) = n \to \bigcup(a) = \bigcup_{i=1}^{n} b(i))$

Первый антецедент идентифицируется с утверждением из контекста. Выражение b имеет заголовок "набор". Это позволяет выделять конечное объединение в заменяющей части указателем "развертка" и преобразовать его в обычное объединение.

$\forall_{an}(\sum_{i,j,i<j,i\in\{1,\ldots,n\},j\in\{1,\ldots,n\}} a(i,j) = \sum_{k=1}^{n}\sum_{l=1}^{k-1} a(l,k))$

Переменная n идентифицируется с натуральной константой, меньшей 4. Обе суммы в заменяющей части выделены указателем "развертка".

Спецификация приема имеет вид "тип(отрицание)", "напрвл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит указатели "развертка" для заменяющей части. Иногда этого достаточно, но ввод ограничений на мощность области суммирования потребует работы доводчика.

Число приемов данного типа - 4.

ii. Попытка сведения вычисления операции над семейством к вычислению операций над другими семействами.

Иногда операцию над семейством можно вычислить, декомпозировав ее в несколько операций над более "простыми" семействами либо проварьировав семейство. Заметим, что термин "семейство" здесь не совсем удачный. Фактически, везде речь идет об отображениях и операциях над ними. Примеры:

$\forall_{abcdfg}(\int_a^b f(x)dx = c\ \&\ \int_z^b g(x)dx = d\ \&\ c - \text{число}\ \&\ d - \text{число} \to \int_a^b (f(x)+g(x))dx = c+d)$

В антецедентах предпринимается попытка вычислить интегралы для слагаемых (проверяется, что c, d не содержат символа "интеграл"), и далее выдается сумма результатов.

$\forall_{af}(0 < f(n)\ \&\ \lim(\lambda_n(\ln(f(n)), n - \text{натуральное})) = a\ \&\ (a - \text{число}\ \vee\ a = \infty\ \vee\ a = -\infty) \to \lim(\lambda_n(f(n), n - \text{натуральное})) = \exp(a))$

Вспомогательные вычисления во всех случаях выполняются задачами на преобразование.

Спецификация приема имеет вид "тип(переменазнака)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры, проверяющие отсутствие символа рассматриваемой операции над отображением в полученных результатах. Во многих случаях этого достаточно для получения приемлемого приема.

Число приемов данного типа - 8.

iii. Занесение внешнего члена под знак операции над семейством.

A. Занесение внешнего члена под знак операции над семейством - случай продолжения ряда значений.

Тип аналогичен предыдущему, но нет необходимости решать задачу на описание, так как известны точки продолжения ряда значений варьируемого параметра. Пример:

$\forall_{amn}(b = a(m-1) \to b + \sum_{i=m}^{n} a(i) = \sum_{i=m-1}^{n} a(i))$

Антецедент сравнивает значения b и $a(m-1)$, используя только нормализаторы общей стандартизации.

Спецификация приема имеет вид "тип(простойцикл)", "направл(N)". Справочник "заголовокприема" указывает только уровень срабатывания 4.

Число приемов данного типа - 4.

iv. Сведение операции над семейством к характеристике класса.

A. Сведение операции над семейством к характеристике класса.

Пример:

$\forall_P(\sum_{x,P(x)}((\mathrm{card}(x))\mathrm{mod}2) = \mathrm{card}(\mathrm{set}_x(P(x)\ \&\ \neg(\mathrm{card}(x) - \mathrm{even}))))$

Спецификация приема имеет вид "тип(нормсигнум)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Хотя в данном примере дополнительных фильтров нет, но в некоторых случаях они нужны и пока автоматически не создаются.

Число приемов данного типа - 4.

v. Переход к операциям над семействами, имеющими более простой вид общего члена.

A. Сведение операции над семейством к операциям над семействами, имеющими более простой вид общего члена.

Обычно прием данного типа выносит наружу не варьируемую часть общего члена либо декомпозирует операцию. Примеры:

$\forall_{afgh}(\sum_{x,f(x)}(ag(x)/h(x) = a\sum_{x,f(x)} g(x)/h(x)))$

$\forall_{fgh}(\prod_{x,h(x)}(f(x)/g(x)) = \prod_{x,f(x)} f(x)/\prod_{x,h(x)} g(x))$

Спецификация имеет вид "тип(фильтррадикалов)", "направл(N)". Справочник "заголовокприема" определяет уровень срабатывания приема 3. В отдельных случаях этого достаточно для получения

полноценного приема, но иногда требуется доработка создания дополнительных фильтров.

Число приемов данного типа - 25.

B. Сведение операции над семейством к операциям над семействами, допускающими непосредственное вычисление.

Пример:

$\forall_{mnpq}(\sum_{i=p}^{q}(iC_{m+1}^{n}) = (n+1)\sum_{i=p}^{q}C_{m+i+1}^{n+1} - (m+1)\sum_{i=p}^{q}C_{m+1}^{n})$

Суммы в заменяющей части обычно легко вычисляются.

Спецификация приема имеет вид "тип(подобныетермы)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. В некоторых случаях требует проработки создание фильтров, уточняющих условия, при которых возможно вычисление.

Число приемов данного типа - 5.

C. Сведение операции над семейством к операциям над подсемействами, на которых упрощается вид общего члена.

Прием выполняет такое подразбиение области определения семейства, при котором сложное подвыражение его общего члена заменяется на более простые частные случаи. Пример:

$\forall_{Pfg}(\iint_{P(x,y)} f(x,y)dxdy = \iint_{P(x,y)\ \&\ 0\le g(x,y)} f(x,y)dxdy + \iint_{P(x,y)\ \&\ g(x,y)\le 0} f(x,y)dxdy)$

Прием имеет указатель "контекст(позиция(x1 фикс(0 1 1 4)вид(x1 модуль(значение(x7 набор(x23 x24))))))", определяющий идентификация подвыражения "$|g(x,y)|$" в общем члене. После разбиение области интегрирования на два подмножества модуль будет устранен.

Спецификация приема имеет вид "тип(началопути)", "направл(N)", "терм(t)", где t - сложное подвыражение, которое будет упрощено после разбиения области. В нашем примере это "модуль(значение(x7 набор(x23 x24)))". Справочник "заголовокприема" указывает уровень срабатывания 2 и создает приведенный выше указатель "контекст(...)". Он требует доработки, хотя в отдельных случаях создает приемлемые приемы.

Число приемов данного типа - 3.

D. Замена переменной для стандартизации вида общего члена семейства.

Пример:

$\forall_{abfg}(f(\lambda_i(g(i+b), i - \text{целое } \&\ a \le i)) = f(\lambda_i(g(i), i - \text{целое } \&\ a+b \le i)))$

Здесь f - одноместная операция, полученная из двуместной ассоциативно-коммутативной операции обобщением на конечное множество операндов. Прием выполняет сдвиг индексации для упрощения общего члена.

Спецификация приема имеет вид "тип(уровеньпосылки)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтр "не(равно(справка(развертка f)0))", проверяющий, что f - обобщение двуместной операции. Он требует доработки.

Число приемов данного типа - 12.

E. Применение специального нормализатора для последующей декомпозиции операции над семейством.

Пример:

$\forall_{fghpq}(g(i)/h(i) = q(i) \to \sum_{i,p(i)} f(i)g(i)/h(i) = \sum_{i,p(i)}(f(i)q(i)))$

Выражение $g(i)$ идентифицируется с произведением всех множителей, представляющих собой многочлены от i. Проверяется, что $h(i)$ - тоже многочлен степени больше первой от i, и далее левая часть антецедента преобразуется к виду суммы простейших дробей с помощью нормализатора "простейшиедроби". Проверяется, что результат $q(i)$ имеет вид суммы. Тогда после преобразования исходная сумма разобьется в несколько более простых сумм. Прием проверяется отсутствие специальных целевых установок, таких, как разложение в ряд Тейлора либо Фурье.

Спецификация имеет вид "тип(блокнеравенства)", "направл(N)". Справочник "заголовокприема" не прорабатывался и определяет лишь уровень срабатывания 2.

Число приемов данного типа - 4.

vi. Сведение операции над семейством к операциям над семействами, имеющими более простую область определения.

A. Переход к операции над семейством меньшего размера.

Пример - вычисление определителя, у которого в первом столбце единственный ненулевой элемент:

$\forall_{Amn}(\mathrm{set}_i(i \in \{1,\ldots,n\}\ \&\ \neg(A(i,1)=0)) = \{m\} \to \det(\lambda_{ij}(A(i,j),$
$i \in \{1,\ldots,n\}\ \&\ j \in \{1,\ldots,n\})) = A(m,1)(-1)^{m+1}\det(\lambda_{ij}((A(i,j+$
$1)$ при $i < m$, иначе $A(i+1,j+1)), i \in \{1,\ldots,n-1\}\ \&\ j \in \{1,\ldots,n-$
$1\})))$

Левая часть антецедента обрабатывается нормализатором "нормненули", определяющим совокупность ненулевых элементов.

Спецификация приема имеет вид "тип(Нижняягрань)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 3. Он не проработан.

Число приемов данного типа - 4.

B. Сведение операции над семейством к операциям над подсемействами, если область определения семейства представлена в виде объединения подобластей. Пример:

$\forall_{PQfn}(\mathrm{set}_{xy}P(x,y) = \bigcup_{i=1}^{n} Q(i)\ \&\ \text{разделены}(Q) \to$
$\iint_{P(x,y)} f(x,y)dxdy = \sum_{i=1}^{n} \iint_{Q(i)} f(x,y)dxdy)$

Первый антецедент выделен указателем "идентификатор". Утверждение $P(x,y)$ явно разрешается относительно x,y задачей на описание, после чего описатель обрабатывается нормализатором "нормкласс". Конечное объединение идентифицируется с обычным.

Спецификация приема имеет вид "тип(номервхождения)", "направл(N)". Справочник "заголовокприема" определяет уровень срабатывания 2, вводит фильтры "натуральное(n)", "не(заголовок(n 1))", а также необходимые указатели "развертка". Он недоработан.

Число приемов данного типа - 7.

C. Группировка членов ассоциативно-коммутативной операции над семейством.

Прием группирует одинаковые элементы семейства. Пример:

$\forall_{Afn}(n = \mathrm{card}(A) \to \sum_{b,b\subseteq A} f(\mathrm{card}(b)) = \sum_{i=0}^{n}(C_n^i f(i)))$

Спецификация приема имеет вид "тип(диффлагранжа)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. Этого достаточно для создания полноценного приема.

Число приемов данного типа - 6.

D. Отбрасывание членов семейства, не изменяющих значение операции.

Отбрасываются элементы семейства, представляющие собой единицу операции. Пример:

$\forall_{Bpq}(\sum_{i,B(i)}(1+(-1)^i)p(i)/q(i) = \sum_{i,B(i),i-\mathrm{even}} 2p(i)/q(i))$

Спецификация приема имеет вид "тип(нормчислосочетаний)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Этого почти достаточно.

Число приемов данного типа - 7.

vii. Частичное сокращение операций над семействами.

Если операция F над конечным семейством является обобщением двуместной ассоциативно-коммутативной операции f, для которой имеется тождество типа сокращения, то это тождество обобщается на случай операции F. Пример:

$\forall_{abcdefg}(0 < f(n)\ \&\ 0 < b-c \to e(\prod_{n=a}^{b} f(n))^d/(g(\prod_{n=a}^{c} f(n))^d) = e(\prod_{n=c+1}^{b} f(n))^d/g)$

Антецеденты обрабатываются проверочными операторами, причем первому антецеденту передается дополнительная посылка $n \in \{a,\ldots,b\}$.

Спецификация приема имеет вид "тип(нормпредел)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3. Он не прорабатывался.

Число приемов данного типа - 3.

viii. Разбиение области изменения параметра для сближения операций над семействами.

Прием подготавливает возможность частичного сокращения либо группировки операций над семействами. Пример:

$\forall_{abcdefg}(0 < e - c \to a\sum_{n=c}^{d} g(n) + b\sum_{n=e}^{f} g(n) = a\sum_{n=e}^{d} g(n) + a\sum_{n=c}^{e-1} g(n) + b\sum_{n=e}^{f} g(n))$

Прием подготавливает возможность приведения подобных членов, выравнивая нижние границы суммирования. Для выравнивания верхних границ создан другой прием. Если очевидно, что d меньше e, то прием не применяется.

Спецификация имеет вид "тип(фильтрпромежутков)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1 и фильтра "условие". Он не прорабатывался.

Число приемов данного типа - 4.

7. Определение характеристики класса.

   (a) Непосредственное определение характеристики класса.

   Пример:

   $\forall_{an}(n-\text{целое} \,\&\, 0 \le n \,\&\, \text{конечное}(a) \to \text{card}(\text{set}_x(x-\text{set} \,\&\, x \subseteq a \,\&\, \text{card}(x) = n)) = C^n_{\text{card}(a)}))$

   Спецификация приема имеет вид "тип(орграф)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. В ряде случаев этого достаточно для создания полноценного приема. Иногда нужна доработка фильтров.

   Число приемов данного типа - 10.

   (b) Стандартизирующая замена переменной при определении характеристики класса.

   Прием делает такую замену переменной, которая сближает описание класса с описаниями, для которых имеется прием непосредственного определения характеристики. Примеры:

   $\forall_{bdf}(b(e) \subseteq f(e) \,\&\, b(e)-\text{set} \,\&\, f(e)-\text{set} \to \text{card}(\text{set}_{ae}(a-\text{set} \,\&\, b(e) \subseteq a \,\&\, a \subseteq f(e) \,\&\, d(a,e))) = \text{card}(\text{set}_{ce}(c \subseteq f(e)(e) \,\&\, c - \text{set} \,\&d(b(e) \cup c, e))))$

   Прием переходит от пары условий вида $A \subseteq X$, $X \subseteq B$ к одному условию вида $X \subseteq C$ на варьируемую переменную X.

   $\forall_{An}(\text{конечное}(A) \,\&\, 0 < \text{card}(A) - 2n \to \text{card}(\text{set}_x(x \subseteq A \,\&\, x - \text{set} \,\&\, n \le \text{card}(x))) = 2^{\text{card}(A)} - \text{card}(\text{set}_x(x \subseteq A \,\&\, x - \text{set} \,\&\, \text{card}(x) \le n - 1)))$

   Спецификация приема имеет вид "тип(внешобрыв)", "направл(N)". Иногда дополнительно вводятся элементы "указатель", определяющие дополнительные посылки проверочных операторов, обрабатывающих антецеденты. Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. Он почти не проработан.

   Число приемов данного типа - 19.

(c) Упрощающий переход к характеристикам других классов.

Пример:

$\forall_{ABPmn}(m = \text{card}B(x)\ \&\ m - \text{целое}\ \&\ n = \text{card}A(x)\ \&\ n \le m \rightarrow$
$\text{card}(\text{set}_{fx}(\text{Отображение}(f, A(x), B(x))\ \&\ \text{взаимнооднозначно}(f)\ \&\ P(x))) =$
$m!\text{card}(\text{set}_x(P(x)))/(m-n)!)$

Имеются фильтры, проверяющие, что m, n не зависят от переменных списка x.

Спецификация приема имеет вид "тип(граф)", "направл(N)". Добавляются элементы "см", указывающие на независимость m, n от переменных связывающей приставки, а также элемент "указатель", определяющий идентификацию x со списком переменных. Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. Он почти не прорабатывался.

Число приемов данного типа - 41.

(d) Попытка свести вычисление характеристики класса к вычислению характеристик других классов.

Приемы данного типа сначала пытаются вычислить характеристики других классов, и лишь в случае успеха реализуют замену. Пример:

$\forall_{AMPQt}(M = \text{set}_i(P(i)\ \&\ i - \text{целое}\ \&\ 0 \le i\ \&\ i \le \text{card}(A))\ \&$
$\text{конечное}(A)\ \&\ t(i) = \text{card}(\text{set}_{xy}(x - \text{set}\ \&\ x \subseteq A\ \&\ \text{card}(x) = i\ \&\ Q(x,y))) \rightarrow$
$\text{card}(\text{set}_{xy}(x - \text{set}\ \&\ x \subseteq A\ \&\ P(\text{card}(x))\ \&\ Q(x,y))) = \sum_{i, i \in M} t(i))$

Предпринимается разбиение семейства подмножеств на подсемейства подмножеств одинаковой мощности. Проверяется, что результат $t(i)$ вычисления числа подмножеств мощности i не содержит символа "класс".

Спецификация приема имеет вид "тип(цикл)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 5, вводит фильтры "условие", "тип(преобразовать)", а также фильтр, проверяющий, что результат определения характеристики другого класса не содержит символа "класс". Он почти не проработан.

Число приемов данного типа - 6.

(e) Разрешение уравнения под описателем "класс" относительно заданной переменной для последующего вычисления.

Прием усматривает уравнение под описателем и разрешает его относительно одной из варьируемых переменных, если оно уже не разрешено относительно нее. Пример:

$\forall_{ABKbfg}(\text{set}_{xyz}(f(x,y,z) = g(x,y,z)\ \&\ A(x,y,z)) = B \rightarrow S(\text{точки}(\text{set}_{xyz}($
$f(x,y,z) = g(x,y,z)\ \&\ A(x,y,z)) \cup b, K)) = S(\text{точки}(B \cup b, K)))$

Утверждения под описателем "класс" в антецеденте разрешаются относительно z с помощью задачи на описание. Затем описатель упрощается нормализатором "нормкласс". Прием используется при определении площадей поверхностей.

Спецификация приема имеет вид "тип(Плюс)", "направл(N)", "переменная(X)", где X - переменная, относительно которой выполняется разрешение. Справочник "заголовокприема" указывает уровень срабатывания 3. В приведенном примере он вводит также следующие фильтры: "тип(преобразовать)", "не(цель(извлекается))", "или(не(заголовок($f(x,y,z)$, z)) входит(z $g(x,y,z)$))", "или(не(заголовок($g(x,y,z)$, z)) входит(z $f(x,y,z)$))", "входит(z фикс(...))", где указатель вхождения "фикс" ссылается на уравнение под описателем. Кроме того, вводится нормализатор "задача(6 тип(описать) полный прямойответ или явное упростить цель(неизвестная(z)))" для утверждений под описателем и нормализатор "нормкласс" для описателя. Требуется дополнительная проработка данного справочника.

Число приемов данного типа - 4.

Приемы обращения к нормализаторам

Приемы этих типов основаны на технических теоремах, не несущих никакой информации кроме того, что определяют последовательность вычислений. Тем не менее, многие из них крайне важны и часто применяются. Обычно эти приемы требуют множества дополнительных фильтров, источники которых связаны с общим анализом алгоритмизации предметной области и пока почти не прорабатывались.

1. Применение нормализатора приведения к заданным заголовкам.

   (a) Попытка применения нормализатора приведения к заданным заголовкам для упрощения выражения.

   Приемы этого типа предпринимают попытку применить к выражению нормализатор приведения к заданным заголовкам в надежде, что он даст сильное упрощение. Если это произойдет, прием выполняет замену. Пример:

   $\forall_{abcd}(d = a/b + c \rightarrow a/b + c = d)$

   Прием применяется к подвыражению условия задачи на преобразование, содержащему сумму с дробями. Антецедент выделен указателем "идентификатор", и его правая часть обрабатывается нормализатором "видумножение", выполняющим сложение дробей и разложение на множители их числителей и знаменателей. Замена происходит, если результат d оказался короче исходного выражения. Прием имеет множество дополнительных фильтров.

   Спецификация приема имеет вид "тип(символ)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2 и вводом фильтра "короче(результат теквхожд)". Он не проработан.

   Число приемов данного типа - 4.

2. Применение нормализатора стандартной формы.

   (a) Попытка упрощения выражения путем применения нормализатора стандартной формы и последующей свертки с помощью других нормализаторов.

Пример:

$\forall_{abcdefg}(g = adb + cd\sqrt{b} + ae\sqrt{b} + ce \ \& \ f - \text{rational} \ \& \ \neg(\text{знаменатель}(f) - \text{even}) \rightarrow (a\sqrt{b} + c)^f (d\sqrt{b} + e)^f = g^f)$

К правой части первого антецедента сначала применяется нормализатор стандартной формы "стандплюс", выполняющий раскрываник скобок. Затем применяется нормализатор "видумножение". Замена выполняется лишь при условии, что заменяющий терм короче заменяемого. Более яркий пример дает алгебра логики, где для упрощения выражений сначала выполняется преобразование к дизъюнктивной нормальной форме с последующими упрощением и сверткой.

Спецификация приема имеет вид "тип(упростить)", "направл(N)", "оператор(A)", где A - список названий последовательно применяемых нормализаторов, начинающийся с нормализатора стандартной формы. Справочник "заголовокприема" определяет уровень срабатывания 3 и фильтры "тип(преобразовать)", "условие", "цель(упростить)", "коммент(длина)", "короче(результат теквхожд)", "не(цель(нормтеорема))". В рассмотренном примере добавляется фильтр "не(цель(стандплюс))". Кроме того, вводятся указатели "попытка(стандплюс теквхожд)", "замечание(стандплюс результат)". Хотя в некоторых случаях этот справочник и дает результаты, близкие к приемлемым, он требует доработки.

Число приемов данного типа - 18.

(b) Попытка применения нормализатора стандартной формы для упрощения выражения.

В отличие от предыдущего случая, для упрощения применяется только нормализатор стандартной формы. Пример - попытка раскрывания скобок для упрощения выражения:

$\forall_{abc}(c = a + b \rightarrow a + b = c)$

Прием применяется в условиях задач на преобразование. Проверяется, что некоторое слагаемое преобразуемой суммы допускается фактическое раскрывание скобок - имеет своим сомножителем сумму либо степень суммы. Правая часть антецедента обрабатывается нормализатором "стандплюс". Проверяется, что результат раскрывания скобок короче исходного выражения, и лишь в этом случае реализуется замена. Имеется ряд дополнительных фильтров.

Спецификация приема имеет вид "тип(Умножение)", "направл(N)", "оператор(P)", где P - название оператора приведения к стандартной форме. Справочник "заголовокприема" ограничивается указанием уровня срабатывания 3 и вводом фильтра "короче(результат теквхожд)". Он не проработан. Более того, так как контексты срабатывания приемов этого типа сильно различаются, сам тип приема, по-видимому, придется разбить на ряд подтипов.

Число приемов данного типа - 5.

3. Применение нормализатора общей стандартизации.

(a) Применение нормализатора общей стандартизации для вычисления в стандартной ситуации.

Прием обращается к нормализатору общей стандартизации, если гарантировано, что он вычислит значение выражения данного типа. Пример:

$\forall_{mn}(m = n! \to n! = m)$

Переменная n идентифицируется с целочисленной константой. Эта константа органичена сверху значением, уточняемым в зависимости от контекста. Возможны варианты 9, 19, 200. Правая часть антецедента обрабатывается нормализатором "нормфакториал".

Спецификация приема - "тип(сложитьдроби)", "направл(N)". Сюда добавляется элемент "см(...)", уточняющий контекст, в котором гарантировано вычисление значения нормализатором. Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2 и вводом фильтра "не(заголовок(m факториал))" (символ варьируется в зависимости от теоремы приема). Кроме того, вводится нормализатор "нормфакториал" правой части антецедента. Справочник недоработан.

Число приемов данного типа - 7.

4. Попытка обращения к нормализатору вычисления.

Прием усматривает выражение, для вычисления которого создан специальный нормализатор (например, предел, интеграл и т.п.), и предпринимает обращение к данному нормализатору. Пример:

$\forall_{abcf}(a = \lim_{x \to b \setminus c} f(x) \to \lim_{x \to b \setminus c} f(x) = a)$

Правая часть антецедента обрабатывается нормализатором "нормпредел". Проверяется, что результат a не имеет заголовка "предел".

Спецификация приема имеет вид "тип(9)", "направл(N)", "оператор(P)", где P - название нормализатора вычисления. Справочник "заголовокприема" определяет уровень срабатывания 0, вводит фильтр, проверяющий невхождение заголовка вычисляемого выражения в результат вычислений, и создает нормализатор обращения к оператору P. Требуется существенная доработка, так как обычно приемы данного типа имеют множество дополнительных фильтров.

Число приемов данного типа - 22.

5. Попытка применения нормализатора, подготавливающего возможность упрощения.

Прием применяет нормализатор для упрощения некоторого надвыражения. Пример:

$\forall_{abcdn}(a + b = cd \to (a + b)c^n = c^{n+1}d)$

К левой части антецедента применяется нормализатор разложения на множители. Проверяется, что выражение d оказалось короче выражения $a + b$.

Спецификация имеет вид "тип(удалениепримечпосылки)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 4. Он не проработан.

Число приемов данного типа - 4.

Сближение подвыражений задачи

1. Задачи на преобразование.

   (a) Переход в задаче на преобразование к уже имевшимся подтермам.

   Прием заменяет выражение на примерно равноценное по своей сложности, но уже встречающееся в условии задачи. Пример:

   $\forall_{abcd}(0 < a \to a^{d \log_b c} = c^{d \log_b a})$

   Имеется указатель, усматривающий в задаче выражение $\log_b a$. Чтобы не возникали конфликты с приемами предыдущих типов, анализируются комментарии (перегруппировка ...).

   Спецификация приема имеет вид "тип(поглощается)", "направл(N)", "исключение(P)", "терм(Q)". Здесь P - исключаемое сложное выражение, Q - заменяющее выражение, уже встречающееся в условии задачи. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "тип(преобразовать)", "условие", "коммент(длина)", "коммент(перегруппировка $\log_b a$)". Также им вводятся указатели "замечание(перегруппировка $\log_b c$)", "контекст(позиция(x5 корень)вид(x5 логарифм(b a)))". В ряде случаев этого достаточно для создания полноценного приема, но некоторая доработка все же нужна.

2. Задачи на описание.

   (a) Преобразование двух неизвестных унифицируемых аргументов, приводящее к единственному неизвестному унифицируемому аргументу.

   Пример:

   $\forall_{abcd}(c = \sin(a - b)\ \&\ d = \sin(a + b) \to \sin a \cos b = (c + d)/2)$

   Оба выражения a, b содержат неизвестные, а хотя бы одно из выражений c, d - не содержит. Каждый содержащий неизвестные тригонометрический аргумент, расположенный вне преобразуемого произведения, совпадает, с точностью до знака, с одним из выражений c, d.

   Спецификация приема имеет вид "тип(нормистина)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "тип(описать)", "условие", "не(известно(a))", "не(известно(b))", "или(известно(c)известно(d))", "контекст(список(x5 c d) тригаргумент(x5 x6) равно(x7 терм(минус(x6))) не(контекст(тригаргумент(корень x8) не(подчинено(x8 фикс(0 1 1))) не(подчинено(x8 фикс(0 1 2))) не(известно(x8)) не(равно(x6 x8)) не(равно(x7 x8)))))". Этого достаточно для создания полноцекнного приема.

   Число приемов данного типа - 3.

(b) Отождествление стандартизируемых операндов неизвестных подвыражений условия задачи на описание.

Пример - тот же, что в предыдущем случае. Однако, здесь либо выражение a не содержит неизвестных, а b - содержит, либо оба эти выражения содержат неизвестные, причем либо a короче, либо длины выражений равны, но a лексикографически предшествует b. В том же самом условии должно иметься выражение вида $\log_a d$.

Спецификация приема имеет вид "тип(нормпринадлежит)", "направл(N)", "терм(t)". Здесь t - в нашем случае $\log_a d$. Спрвочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие", "альтернатива(известно(a) не(известно(b)) и(не(известно(b)) или(короче(a b)и(равнойдлины(a b) лексикопредшествует(a b)))))". Кроме того, вводится указатель "контекст(позиция(x3 корень) вид(x3 логарифм(x1 x4)))". Справочник требует доработки.

Число приемов данного типа - 3.

3. Задачи на доказательство.

(a) Выражение атомарного объекта условия задачи на доказательство через другие атомарные объекты, хотя бы один из которых уже имеется в условии.

i. Выражение атомарного объекта условия задачи на доказательство через другие атомарные объекты, уже имеющиеся в этом условии.

Пример:

$\forall_{ABCDab}$(актив(вектор(AB)) & актив(вектор(BD)) & $D \in$ отрезок(BC) & $al(BD) = bl(CD)$ & $\neg(b = 0) \rightarrow$ вектор(AC) = вектор(AB) + $(a + b)/b \cdot$ вектор(BD))

Заменяемое выражение "вектор(AC)" входит в условие задачи на доказательство, в котором уже имеются выражения "вектор(AB)" и "вектор(BD)".

Спецификация приема имеет вид "тип(связприставка)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(доказать)", "условие", "вхождениетерма(корень вектор(AB))", "вхождениетерма(корень вектор(BD))". Кроме того, введен указатель "заменавхождений". Этого достаточно для автоматического создания приема, совпадающего с созданным вручную.

Число приемов данного типа - 4.

4. Варьирование выражения для получения повторного вхождения. Пример:

$\forall_{abck}(a - b = c \;\&\; 0 \leq c \;\&\; 0 \leq a(\mathrm{mod}\ k) - c \rightarrow b(\mathrm{mod}\ k) = a(\mathrm{mod}\ k) - c)$

В том же терме задачи уже встречается выражение $a(\mathrm{mod}\ k)$. Первый антецедент выделен указателем "идентификатор", два других - обрабатываются проверочными операторами.

Спецификация приема имеет вид "тип(текстТитра)", "направл(N)", "терм(t)". Здесь t - целевое выражение, уже встречающееся в текущем терме. Справочник "заголовокприема" указывает уровень срабатывания 2, вводит фильтр "не(равно(a b))" и указатель "контекст(позиция(x4 корень) вид(x4 $a(\mathrm{mod}\ k)$))". Этого достаточно для создания полноценного приема.

Число приемов данного типа - 5.

Сокращенная переформулировка выражения

1. Сокращенная переформулировка выражений при завершающем редактировании.

   В процессе решения задачи часто используется стандартизация, упрощающая преобразования, но увеличивающая размеры термов. При получении ответа выполняются обратные преобразования, направленные на получение компактной записи. Пример:

   $\forall_{abcd}(a(\sin b)^c/(d(\cos b)^c) = a(\mathrm{tg}\, b)^c/d)$

   Прием применяется в условии задачи. Если это задача на преобразование, то либо должна иметься цель "учетрезультата", либо комментарий "длина". Если задача - на описание, то должна иметься цель "редакция" и отсутствовать цель "редуцирование".

   Спецификация приема имеет вид "тип(левпозиция)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "или(и(тип(преобразовать) или(не(коммент(длина)) цель(учетрезультата)) не(цель(преобразование)) не(цель(быстрпреобр))) и(тип(описать) цель(редакция) не(цель(редуцирование)) не(цель(вспомописание)) не(цель(стандравно))))". Этого достаточно для создания полноценного приема.

   Число приемов данного типа - 34.

2. Корневая свертка условия задачи на преобразование.

   В определенных ситуациях тождество сокращенной перезаписи, применяемое к корневому вхождению в условие задачи на преобразование, может не нарушать целевой направленности действий и приравниваться к общей стандартизации. Пример:

   $\forall_{abcd}(d = a^2 - b^2\ \&\ c - \mathrm{rational}\ \&\ \neg(\text{знаменатель}(c) - \mathrm{even}) \rightarrow (a-b)^c(a+b)^c = d^c)$

   Прием сворачивает в разность квадратов корневое произведение суммы на разность, если нет цели "разложитьнамножители" либо "нормИнтеграл", но есть хотя бы одна из целей "упростить", "длина". Данное преобразование могло бы помешать сокращению дробей, если применялось бы к числителю или знаменателю. Оно иногда нежелательно и в основаниях степеней. Однако, в корневом случае и при отсутствии специальных целевых установок смысла блокировать его нет.

   Спецификация приема имеет вид "тип(внешоперанд)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры

"тип(преобразовать)", "условие", "корень", "или(цель(упростить) цель(длина))". Иногда этого достаточно, но определенная доработка требуется.

Число приемов данного типа - 10.

3. Группировка всех вхождений унифицируемого аргумента в условие задачи на преобразование.

   Пример:

   $\forall_{abcdef}(\neg(d - d\operatorname{tg} a \operatorname{tg} b = 0)\ \&\ \neg(\cos a = 0)\ \&\ \neg(\cos b = 0) \to e(\operatorname{tg} a + \operatorname{tg} b)^c/(f(d - d\operatorname{tg} a \operatorname{tg} b)^c) = e(\operatorname{tg}(a + b))^c/(fd^c))$

   Выражения a, b не являются аргументами тригонометрических функций, расположенных в текущем условии задачи вне преобразуемого выражения.

   Спецификация приема имеет вид "тип(Простоеотношение)", "направл(N)", "символ(s)", "терм(t_1)", ..., "терм(t_n)". Здесь $t_1, \ldots, t_n$ - группируемые унифицируемые аргументы, s - тип этих аргументов (в данном случае - символ "тригаргумент"). Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(преобразовать)", "условие", "конец(не(контекст(тригаргумент(корень x7) не(подчинено(x7 теквхожд)) или(равно(x7 a)равно(x7 b)))))". Этого достаточно для создания полноценного приема.

   Число приемов данного типа - 5.

Преобразование выражений с неизвестными

1. Группировка относительно выражения с неизвестными.

   Приемы данного типа упрощают выражения относительно неизвестных, группируя известные члены. Пример:

   $\forall_{abc}(b \setminus a \cup b \setminus c = b \setminus (a \cap c))$

   Выражение b содержит неизвестные, а выражения a, c не содержат.

   Спецификация приема имеет вид "тип(нормнеизв)", "направл(N)", "неизвестные(X)", где X - список всех переменных, идентифицируемых с содержащими неизвестные выражениями. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "или(тип(описать)тип(преобразовать)тип(исследовать))", "не(известно(корень))", "свободеранд(теквхожд)", "не(известно(b))", "известно(a)", "известно(c)", "не(контекст(ключ(цели независит x4) или(пересекаются(a x4)пересекаются(c x4))))". Этого достаточно для создания полноценного приема.

   Число приемов данного типа - 19.

2. Группировка в одном подвыражении всех неизвестных условия задачи на описание либо посылки задачи на исследование.

   Пример:

   $\forall_{ab}(\sqrt{(}3)a\cos b - a\sin b = 2a\cos(b + \pi/6))$

Выражение b содержит неизвестные, а выражение a - не содержит. Вне заменяемого выражения неизвестные в текущее условие не входят.

Спецификация приема имеет вид "тип(путьвоглавлении)", "направл(N)", "терм(X)", где X - список всех переменных, идентифицируемых с содержащими неизвестные выражениями. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "или(и(тип(описать)условие)тип(исследовать))", "не(известно(b))", "не(сопровождение)", "не(контекст(неизвестная(x3) разряд(корень x3 x4) не(подчинено(x4 фикс(0 1 1))) не(подчинено(x4 фикс(0 1 2)))))", "известно(a)". Этого достаточно для создания полноценного приема.

Число приемов данного типа - 7.

3. Уменьшение глубины неизвестной.

   Пример:

   $\forall_{abcde}(0 < a \to a^{d \log_b c/e} = c^{d \log_b a/e})$

   Переменная c идентифицируется с выражением, содержащим неизвестные, прочие переменные - с выражениями без неизвестных.

   Спецификация имеет вид "тип(точкаокружности)", "направл(N)", "неизвестные(X)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "тип(описать)", "условие", "не(известно(корень))", "свободеранд(теквхожд)", "не(известно(c))", "известно(a)","известно(b)", "известно(d)", "известно(e)". Этого достаточно для создания полноценного приема.

4. Сведение неизвестного подвыражения условия задачи на описание к более простым неизвестным выражениям.

   (a) Сведение неизвестного подвыражения условия задачи на описание либо посылки задачи на исследование к более простым неизвестным выражениям.

   Пример:

   $\forall_a(\sin(|a|) = \sin a \mathrm{sg} a)$

   Спецификация имеет вид "тип(записьприема)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "или(и(тип(описать) условие) тип(исследовать))", "не(известно(теквхожд))". Этого достаточно для синтеза полноценного приема.

   Число приемов данного типа - 3.

5. Использование равенств из посылок для определения неизвестного выражения.

   Пример:

   $\forall_{abm}(m - \text{rational} \ \& \ \text{числитель}(m) - \text{even} \ \& \ |a| = b \to a^m = b^m)$

   Последний антецедент идентифицируется с посылкой задачи на исследование либо на доказательство. Выражение a содержит неизвестные, а выражения b, m - не содержат.

Спецификация приема имеет вид "тип(вставка)", "направл(N)", "антецедент(k)", где k - номер антецедента, идентифицируемого с посылкой. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "или(тип(доказать) тип(исследовать))", "посылка", "не(известно(теквхожд))", "известно(b)", "известно(m)". Этого достаточно для создания полноценного приема.

Число приемов данного типа - 24.

6. Использование равенств из посылок для исключения неизвестных.

   Отличие от предыдущего типа состоит в том, что неизвестные исключаются из преобразуемого выражения лишь частично. Пример:

   $\forall_{abcdmn}(\neg(a = 0)\ \&\ ax/b = cy/d \to mx/(ny) = mbc/(nad))$

   Второй антецедент идентифицируется с посылкой задачи на исследование, имеющей цель "известно", либо задачи на доказательство. Выражения x, y содержат неизвестные; выражения a, b, c, d - не содержат. Остающиеся в заменяющем терме подвыражения m, n могут содержать неизвестные.

   Спецификация приема имеет вид "тип(кратность)", "направл(N)", "антецедент(k)", где k - номер антецедента, идентифицируемого с посылкой. Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "или(тип(доказать) и(тип(исследовать)цель(известно)))", "не(известно(x))", "не(известно(y))", "известно(a)", "известно(b)", "известно(c)", "известно(d)". Этого достаточно для создания полноценного приема.

   Число приемов данного типа - 9.

Преобразования, подготавливающие возможность упрощения

1. Преобразование, приводящее к возможности упрощения надтерма.

   Пример:

   $\forall_a(\sin a/\cos a = \operatorname{tg} a)$

   Проверяется, что преобразуемое выражение - операнд арктангенса либо арккотангенса.

   Спецификация приема имеет вид "тип(плоскоститочки)", "направл(N)", "см(A)", где A - фильтр, определяющий внешний контекст. Справочник "заголовокприема" указывает уровень срабатывания 1 и переносит фильтр A в список фильтров приема. Для данного примера это "контекст(операнд(x2 теквхожд) символ(x2 арктангенс арккотангенс))". Таких действий почти достаточно для создания полноценного приема.

   Число приемов данного типа - 5.

2. Попытка группировки в условии задачи на преобразование для последующего сокращения.

   Пример:

$\forall_{abcd}(d = a^2 - b^2 \ \& \ c - \text{rational} \ \& \ \neg(\text{знаменатель}(c) - \text{even}) \rightarrow (a-b)^c(a+b)^c = d^c)$

Преобразуемое выражение образует пару сомножителей числителя либо знаменателя дроби, у которой противоположная часть делится на d.

Спецификация приема имеет вид "тип(блокфрагментов)", "направл(N)", "см(...)", где последний элемент уточняет внешний контекст. В нашем примере он имеет вид "см(контекст(подтерм(дробь(теквхожд умножение(d x5)))единица(1 x5) дробь))". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(преобразовать)", "условие". Он требует доработки.

Преобразования, связанные с константными выражениями

1. Попытка использовать специальный оператор для упрощения константного выражения.

   Пример:

   $\forall_{abc}(c = a^b \rightarrow a^b = c)$

   Переменная a идентифицируется с десятичным числом, b - с натуральной константой. В зависимости от контекста, на b накладываются различные ограничения сверху. Антецедент выделен указателем "идентификатор" и обращается к реализованному на ЛОСе оператору "натурстепень", вычисляющему значение степени.

   Спецификация приема имеет вид "тип(переучет)", "направл(N)", "оператор(s)", "типданных(a_1, x_1)", "...", "типданных(a_n, x_n)", где последние элементы в нашем примере суть "типданных(десчисло a)", "типданных(натуральное b)". Кроме того, обычно присутствует элемент "едн(x, e)", указывающий, что переменная x не должна принимать единичное значение e. Этот элемент блокирует создание соответствующего указателя "единица", чтобы избежать вырожденных вычислений. Справочник "заголовокприема" указывает уровень срабатывания 0, вводит фильтры "десчисло(a)", "натуральное(b)" и нормализатор обработки антецедента оператором "натурстепень". Он требует доработки.

   Число приемов данного типа - 10.

2. Стандартизация с помощью вчсислений.

   Пример:

   $\forall_{abcdefpq}(p = bf + de \ \& \ q = df \rightarrow ab/(cd) + ae/(cf) = pa/(qc))$

   Переменные b, d, e, f идентифицируются с десятичными константами. Антецеденты выделены указателем "программа" и выполняют вычисления над константными значениями. Прием приводит подобные члены для дробей.

   Спецификация имеет вид "тип(исключприем)", "направл(N)", "см(...)", где последний элемент уточняет типы константных значений. Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "десчисло(b)", "десчисло(d)","десчисло(e)","десчисло(f)". Остальные элементы описания приема вводит компилятор спецификаций, так что в итоге создается полноценный

прием. Заметим, что все элементы спецификации, включая элемент "см", создаются по теореме программой спецификатора, а сама теорема, снабженная обобщающими параметрами, автоматически выводится из обычного тождества сложения дробей. Подробнее об этом будет сказано в последующих томах монографии.

Число приемов данного типа - 32.

3. Усмотрение связи между константами, позволяющей выполнить упрощение.

   Пример:

   $\forall_{abcm}(b = 2m \ \& \ cm^2 + 1 = a \rightarrow \sqrt{a + b\sqrt{c}} = |m\sqrt{c} + 1|)$

   Переменные a, b, c идентифицируются с целочисленными константами. Антецеденты выделены указателем "программа".

   Спецификация приема имеет вид "тип(элементномер)", "направл(N)", "см(...)", где последний элемент указывает типы константных значений. Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "целое(a)", "целое(b)","целое(c)". Этого достаточно для создания полноценного приема.

   Число приемов данного типа - 12.

4. Упрощение с помощью проверки условия на константные подвыражения.

   Отличие от предыдущего типа состоит в том, что условия на константные подвыражения обрабатываются проверочными операторами, а не с помощью непосредственных вычислений.

5. Группировка константных подвыражений, обеспечивающая упрощение с помощью нормализаторов общей стандартизации.

   Пример:

   $\forall_{abcdpqr}((a \log_b c + d)/(p \log_b q + r) = \log_{q^p b^r}(c^a b^d))$

   Переменные a, d, p, r идентифицируются с целочисленными константами, не превосходящими по модулю 4; переменные b, c, q - с десятичными константами либо числовыми дробями. Основание логарифма и выражение под логарифмом в заменяющем терме обрабатываются нормализаторами общей стандартизации, обеспечивающими преобразование их в десятичные константы.

   Спецификация приема имеет вид "тип(коррекцияоткатов)", "направл(N)", "см(...)", где последний элемент перечисляет типы константных значений. Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры, указывающие на типы констант. Этого почти достаточно для создания полноценного приема.

   Число приемов данного типа - 4.

6. Стандартизирующее рекурсивное уменьшение числового параметра.

   Пример:

   $\forall_n(0 < n-1 \to \Gamma(n) = (n-1)\Gamma(n-1))$

   Переменная n идентифицируется с константным выражением произвольного вида. Антецедент обрабатывается проверочным оператором.

   Спецификация приема имеет вид "тип(внешвхождение)", "направл(N)", "см(...)", где последний элемент указывает тип константных выражений. В нашем примере - "см(константа(n))". Справочник "заголовокприема" указывает уровень срабыывания 3 и вводит фильтры, указывающие тип константных значений. Этого достаточно для создания полноценного приема.

   Число приемов данного тип - 3.

7. Использование пакета продукций для приближенного вычисления операции над константными термами.

   Пример:

   $\forall_{Aabn}(A = \lambda_{ij}(a(i,j), i \in \{1,\ldots,n\}\ \&\ j \in \{1,\ldots,n\})\ \&\ (\lambda_{ij}(a(i,j), i \in \{1,\ldots,n\}\ \&\ j \in \{1,\ldots,n\}))^{-1} = b \to A^{-1} = b)$

   Прием применяется в задачах, имеющих явное указание на применение приближенных вычислений. Матрица A идентифицируется в первом антецеденте как константный терм, заданный перечислением строк либо столбцов. Второй антецедент, выделенный указателем "программа", обращается к реализованному на ГЕНОЛОГе пакету продукций "обрматрица". Здесь используется математический сопроцессор.

   Спецификация приема имеет вид "тип(инфимум)", "направл(N)". Добавляются указатели "см(...)" и "указатель(...)", необходимые для создания приема. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "или(и(тип(преобразовать) цель(выч)) и(тип(описать)или(цель(вычисление) Входит(точность списокусловий))))", "условие". Остальные компоненты описания приема берутся из элементов "см", "указатель".

   Число приемов данного типа - 6.

8. Упрощение константного подвыражения посылки с помощью вспомогательной задачи.

   Пример:

   $\forall_{ab}(a = \cos b \to \cos b = a)$

   Преобразуемое выражение константное и содержит внутри себя символ обратной тригонометрической функции. Правая часть антецедента обрабатывается задачей на упрощение, причем проверяется, что результат не содержит символов обратных тригонометрических функций.

Спецификация приема имеет вид "тип(конъюнкциявсех)", "направл(N)". Добавляется элемент "см(...)", уточняющий контекст, в котором целесообразна попытка упрощения. Справочник "заголовокприема" указывае уровень срабатывания 0 и вводит фильтры "посылка", "константа(теквхожд)". Вводится также нормализатор, организующий обращение к задаче на упрощение.

Число приемов данного типа - 3.

Преобразования, связанные с невырожденными числовыми атомами

Напомним, что невырожденные числовые атомы - неоднобуквенные выражения, имеющие численные значения, причем хотя бы один из корневых операндов - нечисленный (например, "масса(A)" либо "угол(ABC)"). Для работы с такими атомами потребовалось создать множество специальных типов приемов.

1. Выражение числового атома через численные параметры.

   (a) Выражение числового атома через численные параметры с помощью равенств в посылках.

      i. Использование равенства из контекста, явно выражающего числовой атом через численные параметры.

      Пример:

      $\forall_{ABp}$(вероятность$(A,B) = p \rightarrow$ вероятность$(A,B) = p)$

      Антецедент идентифицируется с утверждением из контекста, выражение p не содержит невырожденных числовых атомов. Приемы такого типа работают, когда посылка уже имеет достаточно большой вес, а преобразуемое выражение возникло недавно. Тогда общий прием использования равенств для стандартизации термов, инициируемый рассмотрением равенства, не срабатывает.

      Спецификация приема имеет вид "тип(величинаугла)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтр "не(контекст(числовойатом(p x4)не(переменная(x4))))". Этого достаточно, чтобы прием, созданный автоматически, совпал с приемом, введенным вручную.

      Число приемов данного типа - 3.

      ii. Выражение числового атома через численные параметры с помощью равенства из контекста.

      В отличие от предыдущего случая, равенство не дает непосредственного выражения через численные параметры. Пример:

      $\forall_{abcdp}(\neg(p = 0)$ & $q =$ скалумнож$(a,d) -$ скалумнож(a,c) & $pb + c = d \rightarrow$ скалумнож$(a,b) = q/p)$

      Последний антецедент идентифицируется с равенством из контекста. В этом равенстве фигурируют векторное сложение и умножение вектора b на число p. Второй антецедент выделен указателем "идентификатор". Скалярные произведения в нем вычисляются нормализатором

"значскалумнож", так что в итоге выражения p, q не имеют невырожденных числовых атомов.

Спецификация приема имеет вид "тип(очереднойслучай)", "направл(N)", "антецедент(m)", где m - номер антецедента, идентифицируемого с равенством из контекста. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "не(контекст(список(x5 p q) числовойатом(x5 x6) не(переменная(x6))))", "или(не(заголовок(c вектор0)) не(заголовок(p 1)))". Этого достаточно для создания полноценного приема.

Число приемов данного типа - 5.

iii. Выражение числового атома через числовую функцию, заданную равенством из контекста. Пример:

$\forall_{ABabfgx}$(плотнраспред(A, B) $=$ $\lambda_x(f(x), g(x))$ $\rightarrow$ вероятность(прообраз$(A, [a, b]), B) = \int_a^b f(x)dx$)

Антецедент идентифицируется с равенством из контекста. Переменные f, g функциональные.

Спецификация приема имеет вид "тип(неубывает)", "направл(N)", "антецедент(m)", где m - номер антецедента, идентифицируемого с равенством из контекста. Справочник "заголовокприема" указывает уровень срабатывания 3. В зависимости от конкретного вида теоремы приема, могут вводиться дополнительные фильтры и укзатели - например, указатель "развертка", фильтр для невхождения нвырожденных числовых атомов в результаты вспомогательных вычислений и т.п. Иногда этого достаточно.

Число приемов данного типа - 4.

2. Выражение терма с числовыми атомами через численные параметры при помощи соотношений из посылок.

   (a) Использование равенства, выражающего функцию от невырожденного числового атома через численные параметры.

   Пример:

   $\forall_{ab}(\cos a = b \ \& \ 0 \leq a \ \& \ 0 \leq \pi - a \rightarrow \sin a = \sqrt{1 - b^2})$

   Выражение a содержит невырожденные числовые атомы, а выражение b - не содержит.

   Спецификация приема имеет вид "тип(записьпеременной)", "направл(N)", "антецедент(i)", где i - номер антецедента, идентифицируемого с посылкой. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(контекст(числовойатом(x2 x4) не(переменная(x4))))", "контекст(числовойатом(x1 x3) не(переменная(x3)))". Этого достаточно для создания полноценного приема.

   Число приемов данного типа - 4.

3. Приведение подобных членов с числовыми атомами.

   (a) Приведение подобных членов с невырожденными числовыми атомами.

   Пример:

   $\forall_{Kabcdim}(a \cdot \text{крд}(m, K, i)/b + c \cdot \text{крд}(m, K, i)/d = (a/b + c/d)\text{крд}(m, K, i))$

   Выражения a, b, c, d не содержат невырожденных числовых атомов. Заметим, что данный тип приемов можно было бы исключить, создав вместо него единственный универсальный прием, приводящий подобные члены для любых невырожденных числовых атомов. Однако, такой прием привел бы к заметному замедлению системы, предпринимая попытки усмотреть невырожденные числовые атомы в произвольных суммах. Гораздо быстрее реагировать на появление конкретных заголовков таких атомов, хотя за это и придется платить увеличением числа приемов.

   Спецификация приема имеет вид "тип(кривая)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтр "не(контекст(список(x5 x1 x2 x3 x4) числовойатом(x5 x6) не(переменная(x6))))". Этого достаточно для создания полноценного приема.

   Число приемов данного типа - 4.

4. Раскрывание скобок в выражениях с числовыми атомами.

   (a) Раскрывание скобок в выражении с числовым атомом, позволяющее выполнить приведение подобных членов с этим атомом.

   Пример:

   $\forall_{ABabcd}(a(b\text{длина}(A) + c) + d\text{длина}(A) = (ab + d)\text{длина}(A) + ac)$

   Выражения a, b, d не содержат символа "длина".

   Спецификация приема имеет вид "тип(другоевхождение)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "не(входит(длина a))", "не(входит(длина b))", "не(входит(длина d))". Этого достаточно, что созданный автоматически прием совпал с приемом, введенным вручную.

   Число приемов данного типа - 3.

5. Использование пропорциональной линейной комбинации числовых атомов, усматриваемой в посылках.

   (a) Выражение линейной комбинации числовых атомов через численные параметры при помощи посылки, представляющей собой равенство для пропорциональной линейной комбинации тех же атомов.

   Пример:

   $\forall_{ABCDabcdpq}(pl(AB) + ql(CD) = c \;\&\; p = ad \;\&\; q = bd \;\&\; \neg(d = 0) \rightarrow al(AB) + bl(CD) = c/d)$

Первый антецедент идентифицируется с посылкой, второй и третий - выделены указателем "идентификатор". Выражения c, d не содержат невырожденных числовых атомов.

Спецификация приема имеет вид "тип(контрольнормализации)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(контекст(числовойатом(x3 x5) не(переменная(x5))))", "не(контекст(числовойатом(x4 x5) не(переменная(x5))))". Этого достаточно для создания полноценного приема.

Преобразования, связанные с условными выражениями

1. Свертка условного выражения.

   (a) Свертка условного выражения.

   Пример:

   $\forall_a(a - \text{число} \to (a \text{ при } 0 < a, \text{ иначе } -a) = |a|)$

   Указатель "извлечениеварианта" определяет расширенную идентификацию: сворачиваются любые выражения вида $(T(a) \text{ при } 0 < a, \text{ иначе } T(-a))$, причем результатом служит $T(|a|)$.

   Спецификация приема имеет вид "тип(новооператор)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтр "коммент(варианты)". Он требует доработки.

   Число приемов данного типа - 6.

   (b) Свертка условного выражения в условии задачи на преобразование.

   Этот тип уточняет контекст срабатывания для приемов предыдущего типа. При обучении решателя часть сверток условных выражений оказалось возможным применять повсеместно (например, свертка в модуль), а часть - лишь в условиях задач на преобразование. Пример такой свертки:

   $\forall_{ab}((b \text{ при } 0 < b - a, \text{ иначе } a) = \max(a, b))$

   Прием применяется к подвыражению условия задачи на преобразование. Указатель "извлечениеварианта" отсутствует.

   Спецификация приема имеет вид "тип(арккотангенс)", "направл(N)". Справочник "заголовокприема" указывает уровнь срабатывания 1 и вводит фильтры "условие", "тип(преобразовать)". Этого достаточно.

   Число приемов данного типа - 4.

Исключение сложных операций и вычисления

1. Исключение сложной операции.

   (a) Тождественная расшифровка по определению.

i. Тождественная расшифровка по определению для исключения редко используемого понятия.

Если понятие новое и для работы с ним приемов недостаточно, то вполне естественно сразу расшифровать его, используя определение. Примерами могут служить операции симметрической разности, гиперболические функции, и т.п. Вначале решатель сразу же от них избавлялся. Однако, по мере создания приемов для работы с данными операциями, целесообразность немедленного их исключения обычно становится гораздо меньшей. Тогда приемы данного типа обрастают множеством дополнительных ограничений. Например, исключение гиперболических функций теперь предпринимается только при наличии явных экспонент, пересекающихся с аргументами функций по своим параметрам.

Пример:

$\forall_{nx}(n = l(x) \to \text{периметр}(\text{фигура}(x)) = \sum_{i=1}^{n} l(x(i)x(i(\text{mod } n) + 1)))$

Спецификация приема имеет вид "тип(новаргумент)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "или(не(тип(преобразовать)) коммент(длина))", "или(не(тип(описать)) не(цель(редакция)))". Этого почти достаточно.

Число приемов данного типа - 19.

ii. Расшифровка по определению подвыражения условия задачи на описание, содержащего неизвестные.

Пример:

$\forall_x(\text{sh}\, x = (\exp(x) - \exp(-x))/2)$

Преобразуемый терм входит в условие задачи на описание и содержит неизвестные.

Спецификация приема имеет вид "тип(производная)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "условие", "тип(описать)", "не(известно(теквхожд))". Этого достаточно для создания полноценного приема.

Число приема данного типа - 7.

(b) Непосредственное исключение сложной операции.

Пример:

$\forall_a(0 \le a + 1 \ \& \ 0 \le 1 - a \to \cos(3 \arcsin a) = (1 - 4a^2)\sqrt{1 - a^2})$

Спецификация приема имеет вид "тип(стандупорядочение)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтр "не(контекст(подчинено(теквхожд x2) символ(x2 класс отображение) пересекаются(x1 связприставка(x2))))". Смысл такого фильтра - не увеличивать число вхождений варьируемой переменной (в данном примере - a) под описателем. В ручной версии приема этого фильтра не было. Он был подсказан другими приемами данного типа. Автоматически сгенерированная версия приема здесь даже лучше исходной.

Число приемов данного типа - 95.

(c) Исключение сложной операции путем развертки.

Пример:

$\forall_{mn}(m - \text{натуральное} \ \& \ n - \text{натуральное} \rightarrow C_n^m = \prod_{k=0}^{m-1}(n-k)/m!)$

Переменная m идентифицируется с натуральной константой, меньшей 5. Конечное произведение разворачивается в обычное с помощью нормализатора "нормпроизведениевсех".

Спецификация приема имеет вид "тип(числкоэффициент)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 4. Он не прорабатывался.

Число приемов данного типа - 4.

(d) Исключение сложной операции с помощью кванторного тождества из контекста.

i. Непосредственное исключение сложной операции с помощью кванторного тождества, имеющегося в контексте.

Это - частный случай предыдущего типа, когда значение сложной операции сразу задается кванторным тождеством. Пример:

$\forall_{ABCPQpt}(\forall_x(C(x) \rightarrow \text{вероятность}(A(x), B(x)) = p(x)) \ \& \ (A(t), B(t)) = (P, Q) \ \& \ C(t) \rightarrow \text{вероятность}(P, Q) = p(t))$

Первый антецедент идентифицируется с утверждением из контекста, причем выражение $p(x)$ не содержит символа "вероятность". Переменные A, B, C, p функциональные. Второй антецедент выделен указателем "идентификатор", третий - обрабатывается оператором "очевидно".

Спецификация приема имеет вид "тип(дискретнаяматематика)", "направл(N)", "см(...)", где последний элемент содержит ограничение на заменяющую часть кванторного тождества. Справочник "заголовокприема" определяет уровень срабатывания 2 и вводит фильтр "условие". В сочетании с элементом "см" этого достаточно.

Число приемов данного типа - 3.

(e) Исключение сложной операции при установлении независимости вспомогательного терма от заданных параметров.

Пример:

$\forall_{ABijknp}(l(A) = n \ \& \ \text{незавсобытия}(A, B) \ \& \ j \in \{0, \ldots, n\} \ \& \ \text{вероятность}($ $A(k), B) = p \rightarrow \text{вероятность}(\text{слойсемейства}(A, \text{элемсобытия}(B), j), B) = C_n^j p^j (1-p)^{n-j})$

Переменная k выбирается приемом как новая. Четвертый антецедент выделен указателем "идентификатор". Его левая часть обрабатывается вспомогательной задачей на преобразование. Этой задаче передается дополнительная посылка $k \in \{1, \ldots, n\}$. Проверяется, что результат p не содержит переменной k.

Спецификация приема имеет вид "тип(неизв)", "направл(N)", "конст(x, y)", где y - вспомогательная переменная, от которой не должно зависеть выражение x. В нашем случае $x = p, y = k$. Добавляется элемент "посылки(...)", определяющий посылки для переменной y. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтр "не(входит($k\ p$))". Он также создает нормализатор, обрабатывающий левую часть четвертого антецедента вспомогательной задачей. Однако, справочник недоработан.

Число приемов данного типа - 7.

(f) Исключение сложной операции с помощью тождеств из контекста.

Пример:

$\forall_{knp}(n(\mathrm{mod}k) + p = 0 \to [n/k] = (n + p)/k)$

Антецедент идентифицируется с утверждением из контекста.

Спецификация приема имеет вид "тип(упрощение)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Этого достаточно.

Число приемов данного типа - 7.

(g) Попытка исключения сложной операции, заключенной внутри терма, идентифицированного с теоремной переменной.

Пример:

$\forall_{abcd}(d = a^2 - b^2\ \&\ 0 \le a - b\ \&\ 0 \le a + b \to (a - b)^c(a + b)^c = d^c)$

Хотя бы одно из выражений a, b имеет вид $p \cdot q^{m/2n}$, где p - десятичная константа. После перехода к разности квадратов двойка в знаменателе показателя степени исключается.

Спецификация приема имеет вид "тип(функционально)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(преобразовать)". Он не прорабатывался.

Число приемов данного типа - 6.

2. Попытка вычисления.

(a) Попытка использовать заданную последовательность промеуточных вычислений.

Это весьма часто встречающийся тип приема, где некоторые заранее созданные вычислительные возможности используются для исключения сложной операции. Примеры:

$\forall_{abcd}(b \le a\ \&\ a \le c\ \&\ d = [b]\ \&\ 0 < d - c + 1 \to [a] = d)$

Первые два антецедента при помощи синтезаторов "нижняяоценка" и "верхняяоценка" определяют константные значения a, b. Третий антецедент вычисляет целую часть константы b, четвертый - проверяет, что она подходит в качестве результата.

$\forall_{abcdfgh}(f = \lambda_x(g(x), h(x))$ & стационарныеточки$(f) = a$ & особыеточки$(f) = b$ & граница$(\text{set}_x(h(x))) = c$ & критическиеточки$(f) = a \cup b \cup d)$

Первые три антецедента идентифицируются с посылками, причем второй и третий были выведены заранее согласно общей схеме нахождения критических точек для внутренности области определения функции. Четвертый и пятый антецеденты выделены указателем "идентификатор". Сначала нормализатор "нормграница" определяет границу рассматриваемой области, затем вспомогательная задача на преобразование находит критические точки этой границы.

Спецификация приема имеет ви "тип(извлечение)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 4. Он не прорабатывался.

Число приемов данного типа - 8.

(b) Попытка вспомогательных вычислений, позволяющих перейти к сложной операции с более простыми операндами.

Пример:

$\forall_{ABfgh}(B = \text{образ}(\lambda_x(g(x), h(x)), A) \to \text{образ}(\lambda_x(f(g(x)), h(x)), A) = \text{образ}(\lambda_x(f(x), \text{одз}(f(x))), B))$

Антецедент выделен указателем "идентификатор". Его правая часть обрабатывается нормализатором "нормобраз", после чего выражение B символа "образ" не содержит. Терм $g(x)$ отличен от x.

Спецификация приема имеет вид "тип(равноудалена)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 3. Он не прорабатывался.

Число приемов данного типа - 6.

3. Вычисления по рекурсии.

(a) Вычисление с помощью префиксной рекурсии.

Пример:

$\forall_{abp}(p = (a \in \{; b\}) \to \text{card}\{a; b\} = \text{card}\{; b\} + (0 \text{ при } p, \text{ иначе } 1))$

Антецедент выделен указателем "идентификатор" и его правая часть обрабатывается задачей на описание. В качестве неизвестной выступает некоторый параметр выражения a. Напомним, что $\{a; b\}$ обозначает терм "перечень(префикс(x1 x2))", $\{; b\}$ - терм "перечень(x2)".

Спецификация приема имеет вид "тип(номера)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 3. Он недоработан.

Число приемов данного типа - 10.

4. Переход к сложной операции, имеющей более простые операнды.

(a) Переход к сложной операции, имеющей более простые операнды.

Пример:

$\forall_{ab}(\sup\{a;b\} = \max(a, \sup\{;b\}))$

Оценка сложности супремума больше, чем максимума. Операнд супремума упрощается.

Спецификация приема имеет вид "тип(описатель)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит нормализатор, обращающийся к задаче на упрощение. Обычно этого достаточно.

Число приемов данного типа - 53.

(b) Декомпозиция сложной операции.

Пример:

$\forall_{abc}(0 \le a \ \& \ 0 \le b \to \log_c(ab) = \log_c a + \log_c b)$

Спецификация приема имеет вид "тип(сопровождтерм)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "или(посылка и(или(не(тип(описать)) не(цель(редакция)) цель(редуцирование)) ии(не(тип(преобразовать)) коммент(длина)) или(тип(доказать)и(не(цель(свертка)) или(известно(x3) не(известно(x1)) не(известно(x2)))))))", "не(контекст(подчинено(теквхожд x4) символ(x4 класс отображение) пересекаются(связприставка(x4) x3) не(пересекаются(связприставка(x4)x1)) не(пересекаются(связприставка(x4)x2))))". Обычно этого достаточно.

Число приемов данного типа - 32.

(c) Вынесение наружу операции над семейством из-под сложного понятия.

Пример:

$\forall_{ABCn}(\text{несовместны}(\lambda_i(A(i), B(i)), C) \to \text{вероятность}(\bigcup_{i,B(i)} A(i), C) = \sum_{i,B(i)} \text{вероятность}(A(i), C))$

Спецификация приема имеет вид "тип(усмделит)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Этого почти достаточно.

Число приемов данного типа - 6.

(d) Решение вспомогательной задачи для представления множества в виде объединения и вынесение этого объединения из-под сложной операции.

Множество, заданное подтермом сложной операции, выражается как описатель "класс", примененный к условию принадлежности этому множеству. Если после явного разрешения такого условия относительно варируемой переменной и упрощения описателя "класс" возникает объединение нескольких множеств, то далее оно выносится из-под сложной операции. Пример:

$\forall_{AKQ}(\text{set}_{xyz}((x,y,z) \in A) = \bigcup_{i=1}^{n} Q(i)$ & разделены$(Q) \rightarrow$
объем(точки$(A,K)) = \sum_{i=1}^{n}$ объем(точки$(Q(i),K))$)

Предполагается, что выражение A имеет заголовок "класс". Условие принадлежности в первом антецеденте разрешается относительно z вспомогательной задачей на описание. Далее испрльзуется нормализатор "нормкласс". Конечные сумма и объединение разворачиваются в обычные.

Спецификация приема имеет вид "тип(контрольглубины)", "направл(N)". Добавляется элемент "см(...)", уточняющий контекст срабатывания. Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2 и вводом фильтра "условие".

Число приемов данного типа - 3.

5. Группировка сложных операций.

   (a) Группировка сложных операций.

   Пример:

   $\forall_{abcde}(0 < a \;\&\; 0 < b \;\&\; c = 1 - ab \;\&\; c < 0 \;\&\; d = a + b \rightarrow e \cdot \operatorname{arctg} a + e \cdot \operatorname{arctg} b = e \cdot \pi + e \cdot \operatorname{arctg}(d/c))$

   Прием применяется в условиях задач. Либо a, b константные, причем решается задача на преобразование, либо происходит редактирование ответа.

   Спецификация приема имеет вид "тип(логзамена)", "направл(N)". справочник "заголовокприема" ограничивается указанием уровня срабатывания 3 и вводом фильтра "условие". Он не прорабатывался.

   Число приемов данного типа - 9.

   (b) Подготовка к группировке сложных операций.

   Пример:

   $\forall_{ab}(0 < a \;\&\; b = 1/\sqrt{a^2+1} \rightarrow \operatorname{arctg} a = \arccos b)$

   Имеет место линейная комбинация текущего арктангенса с арксиунусом либо с арккосинусом. Модули коэффициентов этой комбинации совпадают.

   Спецификация приема имеет вид "тип(ребра)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "условие", "тип(преобразовать)". Он не прорабатывался.

   Число приемов данного типа - 3.

6. Упрощение контекста сложной операции.

   (a) Перегруппировка с перенесением сложных операций на более "удобные" вхождения.

   Пример:

   $\forall_{abdeghipq}(p = hg^{i/2} - e \;\&\; q = g^i h^2 - e^2 \;\&\; \neg(q = 0) \;\&\; b - \text{rational} \;\&\; \neg(\text{знаменатель}(b) - \text{even}) \rightarrow d/(a(e + hg^{i/2})^b) = dp^b/(aq^b))$

Прием переносит иррациональность из знаменателя в числитель. Предварительный анализ на уровне базы теорем приводит к выводу, что появление иррациональности в числителе удобнее, чем в знаменателе: при сложении дробных выражений иррациональности в знаменателе "размножаются", а в числителе - нет.

Спецификация приема имеет вид "тип(вычет)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "условие", "тип(преобразовать)", "цель(упростить)", "коммент(длина)". Он не прорабатывался.

Число приемов данного типа - 8.

7. Подготовка выражения к исключению сложной операции.

   (a) Преобразование сложной операции, ориентированное на последующее ее вычисление.

   Пример:

   $\forall_{ABCDEF}(\text{ромб}(ABCD)\ \&\ E \in \text{отрезок}(AB)\ \&\ F \in \text{отрезок}(AD)\ \&\ \text{актив}(S(\text{фигура}(BEC))) \rightarrow S(\text{фигура}(AECF)) = S(\text{фигура}(ABCD)) - S(\text{фигура}(BEC)) - S(\text{фигура}(CDF)))$

   Вычисление площади четырехугольника $AECF$ сводится к вычислению площадей ромба и двух треугольников, обеспеченному известными формулами.

   Спецификация приема имеет вид "тип(нормрасстояний)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Он не прорабатывался.

   Число приемов данного типа - 11.

   (b) Преобразование, подготавливающее возможность исключения сложного понятия, расположенного в терме, идентифицированном с переменной.

   Пример:

   $\forall_{ab}(\sin(a+b) = \sin a \cos b + \cos a \sin b)$

   Выражение a имеет вид kA либо $kA/2$, где заголовком A служит обратная тригонометрическая функция.

   Спецификация приема имеет вид "тип(началоразбора)", "направл(N)". К этому добавляются элемент "см(...)", уточняющий контекст. Справочник "заголовокприема" указывает уровень срабатывания 3 и переносит фильтры из элемента "см".

   Число приемов данного типа - 6.

8. Реализация специальной целевой установки задачи на преобразование.

   Пример:

$\forall_{abflpq}(l = q - p \ \& \ 0 < l \ \&$ нечетнаяфункция$(\lambda_x(f(x), x -$ число$)) \ \&$
$a = \lambda_i(2/l \int_p^q f(x)\sin(2\pi ix/l)dx, i -$ целое$) \rightarrow f(x) = \sum_{i=1}^{\infty}(a(i)\sin(2\pi ix/l)))$

Текущая задача на преобразование имеет цель (рядфурье $x\ p\ q$), указывающую на необходимость разложения определяемой условием задачи функции от x в ряд Фурье на отрезке $[p, q]$.

Спецификация приема имеет вид "тип(теорияграфов)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 5, вводит фильтр "тип(преобразовать)" и указатель "корень". Он не прорабатывался.

Число приемов данного типа - 3.

9. Упрощающее нетождественное преобразование при наличии специальной цели.

   Пример:

   $\forall_n((n \rightarrow \infty) \ \& \ m(n) -$ целое $\& \ \lim_{n\rightarrow\infty}\{m(n)\} = \infty \rightarrow$
   $m(n)! = \sqrt{2\pi m(n)}m(n)^{m(n)}\exp(-m(n)))$

   Текущая задача на преобразование имеет цель "асимптоценка". Напомним, что фигурные скобки под знаком предела указывают на предел последовательности.

   Спецификация приема имеет вид "тип(движвлево)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(преобразовать)", "условие". Он не прорабатывался.

   Число приемов данного типа - 7.

Использование тождеств для специальной стандартизации

К этим типам относятся приемы, реализующие ту специальную стандартизацию выражений, решение о которой принимается путем анализа имеющихся теорем предметной области. Это уровень более высокий, нежели логический ассмеблер, которому должна передаваться вся информация о фильтрах и указателях в уже готовом виде. Он будет рассмотрен в томах монографии, посвященных базе теорем.

1. Специальная стандартизация.

   Пример:

   $\forall_{abcde}(b = cd + e \ \& \ 0 < c \rightarrow \sin(a + \pi b/d) = (-1)^c\sin(a + \pi e/d))$

   Используется одна из формул приведения. Переменные b, d идентифицируются с натуральными константами.

   Спецификация приема имеет вид "тип(автоклавиатура)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Он не проработан.

   Число приемов данного типа - 118.

2. Применение нормализатора приведения к заданным заголовкам для контекстной стандартизации.

   Пример:

   $\forall_{abcdef}(f = b + c \ \& \ 0 \leq b + c \rightarrow a(b + c)^d/e = af^d/e)$

   Прием предпринимает попытку разложения на множители числителя дробного выражения в условии задачи на преобразование, имеющей цель "упростить". Это - обычная стандартизация дробных выражений (включающая также разложение на множители знаменателей), применяемая до завершающей сокращенной переформулировки ответа. Правая часть первого антецедента обрабатывается нормализатором "видумножение".

   Спецификация приема имеет вид "тип(стандарт)", "направл(N)", "оператор(P)", где P - применяемый нормализатор приведения к заданным заголовкам. Справочник "заголовокприема" ограничивается указанием уровня срабатывания 3. Он не прорабатывался.

   Число приемов данного типа - 29.

3. Обращение к нормализатору стандартной формы для контекстной стандартизации.

   Пример:

   $\forall_{abcdefn}(f = (a + b)^n c/d + e \rightarrow \sin((a + b)^n c/d + e) = \sin f)$

   Реализуется переход к сумме под синусом. Это обычная стандартизация, вызванная тем, что есть формула синуса суммы, но нет формул синуса произведения либо степени. Переменная n идентифицируется с натуральной константой, меньшей 4. Хотя бы одно из выражений c, d, n отлично от 1.

   Спецификация приема имеет вид "тип(Прямоепроизведение)", "направл(N)", "см(...)", где последний элемент уточняет контекст срабатывания. Справочник "заголовокприема" указывает уровень срсбатывания 2 и переносит фильтры из элемент "см". Заметим, что система уже сейчас порождает как теорему приема, так и элемент спецификации "см" самостоятельно. Тем не менее, автоматическое создание приемов данного типа нуждается в доработке.

   Число приемов данного типа - 13.

Тождественные замены, связанные с координатами

1. Выражение координат объекта через невырожденные числовые атомы.

   Пример:

   $\forall_{ABCDEK}(K = (A, B, C, E) \ \& \ D \in \text{прямая}(AB) \ \& \ \text{точкалуча}(A, B, D) \rightarrow$
   $\text{коорд}(D, K) = (l(AD)/l(AB), 0, 0))$

   Исключение делается для посылок, представляющих собой равенство координаты координатному набору.

Спецификация приема имеет вид "тип(добавлениеветви)", "направл(N)". Справочник "заголовокприема" указывает уровни срабатывания 1,4 и 5. Он также вводит фильтр "или(условие не(контекст(подтерм(равно(теквхожд x1)) символ(x1 набор)корень)))". Этого достаточно.

Число приемов данного типа - 11.

2. Выражение координат объекта через координаты других объектов и вычисление последних с помощью нормализаторов.

   Пример:

   $\forall_{ABKabcdef}$(коорд$(A,K)=(a,b,c)$ & коорд$(B,K)=(d,e,f)\rightarrow$
   коорд(вектор$(AB),K)=(d-a,e-b,f-c)$)

   Левые части антецедентов обрабатываются нормализатором "нормкоорд", вычисляющим координаты концов вектора.

   Спецификация приема имеет вид "тип(восстановлениеменю)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))". Этого достаточно для создания полноценного приема.

   Число приемов данного типа - 3.

3. Выражение координат объекта через указанные в посылках координаты другого объекта.

   Пример:

   $\forall_{ABCKabcd}$(коорд(вектор$(AC),K)=(a,b)$ & коорд(вектор$(CB),K)=(c,d)\rightarrow$
   коорд(вектор$(AB),K)=(a+c,b+d)$)

   Антецеденты идентифицируются с посылками.

   Спецификация приема имеет вид "тип(Набор)", "направл(N)", "антецедент(...)". Последний элемент перечисляет номера непосредственно идентифицируемых антецедентов. Справочник "заголовокприема" ограничивается указанием уровня срабатывания 3. Он не прорабатывался.

   Число приемов данного типа - 8.

4. Выражение числового атома через координаты.

   (a) Выражение числового атома посылки через координаты.

   Пример:

   $\forall_{Kabc}$(горизплосквект(a,K) & крд$(a,K,1)=b$ & крд$(a,K,2)=c\rightarrow$
   длина$(a)=\sqrt{b^2+c^2}$)

   Два последних антецедента идентифицируются с посылками.

   Спецификация приема имеет вид "тип(типы)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры

"посылка", "или(тип(доказать)тип(исследовать))". Этого почти достаточно.

Число приемов данного типа - 18.

(b) Выражение числового атома посылки через параметры уравнения для координат множества объектов.

Пример:

$\forall_{ABEKabcde}$(прямкоорд(K) & фокус(A, E) & фокус(B, E) & разныеточки(A, B) & коорд(E, K) = $\text{set}_{xy}(ax^2 + bx + cy^2 + dy + e = 0$ & $x -$ число & $y -$ число) & $\neg(a = 0)$ & $\neg(c = 0) \rightarrow l(AB) = \sqrt{|(c - a)(4ace - b^2c - d^2a)|}/(ac))$

Первые три антецедента и пятый антецедент идентифицируются с посылками.

Спецификация приема имеет вид "тип(знач)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(корень актив))", "не(равно(A B))". Этого почти достаточно.

Число приемов данного типа - 4.

(c) Выражение числового атома условия задачи на преобразование через координаты.

Пример:

$\forall_{ABCKPQRS}$(параллелепипед(A) & прямоугольный(A) & грань(фигура(P QRS), A) & прямкоорд(K) & вверх(вектор(PQ), K) $\rightarrow$ верхнийуровень(A, K) = крд($Q, K, 3$))

Спецификация приема имеет вид "тип(Копия)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "условие", "тип(преобразовать)". Иногда этого достаточно.

Число приемов данного типа - 5.

5. Определение характеристики множества, заданного через координаты своих элементов.

Пример:

$\forall_{KPQf}$(прямкоорд(K) & P = точки($\text{set}_{xyz}(z = f(x, y)$ & $Q(x, y)), K) \rightarrow S(P) = \iint_{Q(x,y)} \sqrt{1 + (df(x, y)/dx)^2 + (df(x, y)/dy)^2} dxdy$)

Второй антецедент выделен указателем "идентификатор". Он обращается к оператору "смточки", пытающемуся при помощи равенств из посылок усмотреть представление области P как множества точек с заданными кординатами.

Спецификация приема имеет вид "тип(просмотртерма)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтр "условие". Иногда этого достаточно.

Число приемов данного типа - 14.

6. Выражение координат множества объектов через описатель "класс".

   Пример:

   $\forall_{ABCDKP}(P = \text{Круг}(ABC)$ & прямкоорд(K) & вертплосквект(вектор$(AB), K)$ & вертплосквект(вектор$(AC), K) \rightarrow$ коорд$(P, K) = \text{set}_{xyz}(x -$ число & $z -$ число & $(x - (\text{крд}(A, K, 1)))^2 + (z - (\text{крд}(A, K, 3)))^2 \leq l(AB)^2$ & $y = \text{крд}(A, K, 2)))$

   Спецификация приема имеет вид "тип(внешконтекст)", "напрввл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Иногда этого достаточно.

   Число приемов данного типа - 6.

7. Задачи на преобразование либо на описание, имеющие цель "класс". Напомним, что ответ задачи на преобразование с такой целью не должен содержать описателей. В случае задачи на описание цель (класс A B) решается для переформулировки условия принадлежности некоторому классу набора значений переменных списка A, ориентированной на исключение в задании класса переменных списка B.

   (a) Исключение вспомогательных параметров в условии задачи на преобразование, имеющей цель "класс".

   Пример:

   $\forall_{ABKabc}(\text{коорд}(A, K) = (a, b)$ & коорд$(B, K) = (c, b)$ & $0 < c - a \rightarrow \text{set}_X(X -$ точка & $\exists_x(x -$ число & коорд$(X, K) = (x, b)$ & $a \leq x)) = \text{луч}(AB))$

   Прием применяется в задаче на преобразование, имеющей цель "класс". Хотя бы одна переменная, выделенная комментарием "вспомпараметр", входит в выражения a, b, K. Переменные A, B не выделены как вспомогательные параметры. Заменяемый терм не расположен под квантором либо описателем. Заметим, что некоторые приемы данного типа могут не устранять описатель "класс", а лишь избавляться от вспомогательных переменных.

   Спецификация приема имеет вид "тип(нормравно)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "условие", "тип(преобразовать)", "цель(класс)", "контекст(список(х4 a b K)входит(х5 параметры(х4)) не(коммент(вспомпараметр переменная(х5))))", "не(контекст(подчинено(теквхожд х4) символ(х4 существует класс)))", "переменная(х26)", "переменная(х27)", "коммент(вспомпараметр переменная(х26))", "коммент(вспомпараметр переменная(х27))", "не(равно(х1 х3))". Этого почти достаточно.

   Число приемов данного типа - 11.

   (b) Переход от координатного задания множества к бескоординатному в задаче на преобразование, имеющей цель "класс".

   Отличие от предыдущего случая состоит в том, что исключения вспомогательных параметров не происходит. Пример:

$\forall_{ABCDEKax}$(прямкоорд(K) & $K = (A, B, C)$ & $D \in$ прямая(AC) & $E \in$ прямая(AC) & разныеточки(D, E) $\rightarrow$ set$_X$($X -$ точка & $\exists_x$($x -$ число & коорд(X, K) $= (x, a)$)) = перпендикуляр(прямая(DE), тчкоорд($K, (0, a)$)))

Спецификация приема имеет вид "тип(общаяточка)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "условие", "тип(преобразовать)", "цель(класс)". "не(равно(D E))". Этого почти достаточно.

Число приемов данного типа - 8.

(c) Декомпозиция координатного задания множества в задаче на преобразование, имеющей цель "класс".

Пример:

$\forall_{Kabcdfpr}$(прямкоорд(K) & $0 < a$ & $p = -d/a + (b^2 + c^2)/(4a^2)$ & $0 < p$ & $r = \sqrt{p} \rightarrow$ set$_A$($A -$ точка & $\exists_{xy}$($x -$ число & $y -$ число & коорд(A, K) $= (x, y)$ & $f(x, y)$ & $0 < ax^2 + ay^2 + bx + cy + d$)) = set$_A$($A -$ точка & $\exists_{xy}$($x -$ число & $y -$ число & коорд(A, K) $= (x, y)$ & $f(x, y)$)) \ круградиуса(тчкоорд($K, (-b/(2a), -c/(2a))), r$))

Спецификация приема имеет вид "тип(сохранениеменю)", "направл(N)". Справочник "заголовокприема" указывает уровень срсбатывания 3 и вводит фильтры "условие", "тип(преобразовать)", "цель(класс)". Этого почти достаточно.

Число приемов данного типа - 4.

8. Использование посылки для перехода к бескоординатному описанию множества в задаче на исследование, имеющей цель "точки".

Цель "точки" указывает на задачу, в которой требуется получить бескоординатное описание множества точек, стартуя с его координатного задания. Пример приема:

$\forall_{Kabcd}$($a =$ точки(b, K) & $a = c \rightarrow a \cup d = c \cup d$)

Оба антецедента идентифицируются с посылками. Происходит следующее. Некоторое множество точек, заданное множеством b их координат в системе координат K, обозначено переменной a. В процессе решения задачи для a было найдено выражение c, не содержащее символа "точки" и не являющееся переменной. Хотелось бы просто заменить a на c. Однако, на ГЕНОЛОГе трудно это сделать так, чтобы точка привязки находилась в преобразуемом терме задачи - не понятно, по какому символу выполнить привязку. Поэтому в данном приеме предполагается, что a - операнд теоретико-множественной операции, именно - объединения. Для пересечения и разности созданы еще два приема.

Спецификация приема имеет вид "тип(ограничено)", "направл(N)", "антецедент(...)", где последний элемент перечисляет номера непосредственно идентифицируемых антецедентов. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "тип(исследовать)", "цель(точки)", "не(входит(точки x3))", "не(переменная(x3))", "переменная(x1)". Этого почти достаточно.

Число приемов данного типа - 3.

14.2.3 Приемы эквивалентной замены

Общая стандартизация одного утверждения

1. Безусловная общая стандартизация одного утверждения.

   Приемы общей стандартизации утверждений, не имеющие существенных посылок. Пример:

   $\forall_{ab}(b-\text{rational} \,\&\, \neg(\text{числитель}(b)-\text{even}) \,\&\, \neg(\text{знаменатель}(b)-\text{even}) \to 0 < a^b \leftrightarrow 0 < a)$

   Спецификация приема имеет вид "тип(нормэкв)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания, который в зависимости от теоремы приема варьируется от 0 до 3. Обычно этого достаточно.

   Число приемов данного типа - 266.

2. Условная общая стандартизация одного утверждения.

   Приемы общей стандартизации утверждений, имеющие существенные посылки. Пример:

   $\forall_{abc}(\text{непересек}(a,b) \to b \subseteq c \leftrightarrow b \subseteq a \cup c)$

   Замена выполняется справа налево. Антецедент обрабатывается проверочным оператором.

   Спецификация приема имеет вид "тип(числооперандов)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2 либо 3. Иногда вводит ряд дополнительных фильтров, блокирующих усложнение утверждения относительно неизвестных, переменных связывающей приставки либо неконстантных термов. Обычно получается приемлемый прием.

   Число приемов данного типа - 324.

3. Элементарная переформулировка с исключением сложного понятия.

   Пример:

   $\forall_{Afxy}(\text{группа}(A) \,\&\, f = \text{операция}(A) \,\&\, x \in \text{носитель}(A) \,\&\, y \in \text{носитель}(A) \to \text{коммутатор}(x,y,A) = \text{единица}(f) \leftrightarrow f(x,y) = f(y,x))$

   Спецификация приема имеет вид "тип(обрыв)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. Иногда вводит ряд дополнительных фильтров, блокирующих ухудшение ситуации в прочих отношениях (усложнение относительно неизвестных и т.п.). Обычно этого достаточно.

   Число приемов данного типа - 6.

4. Дизъюнктивно-конъюнктивная декомпозиция элементарного утверждения.

   (a) Конъюнктивная декомпозиция элементарного утверждения.

i. Конъюнктивная декомпозиция элементарного утверждения.

Это - общий случай конъюнктивной декомпозиции, выполняемой безотносительно к наличию специальной цели. Фильтры приемов данного типа и его подтипов лишь блокируют действия, идущие вразрез с текущими целями. Выбор между данным типом и подтипами предоставляется доводчику. Пример:

$\forall_{abc}(c \subseteq a \setminus b \leftrightarrow c \subseteq a \ \& \ \text{непересек}(b, c))$

Спецификация приема имеет вид "тип(огрсверху)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "или(посылка не(тип(описать)) и(или(и(антецедент(теквхожд) не(цель(редакция))) и(или(не(цель(редакция)) цель(нормантецеденты)) не(цель(свертка)) или(не(известно(корень)) не(цель(соединение))))) или(цель(развертка) и(или(не(неизвестная(c)) не(цель(редакция))) или(известно(c) не(известно(a)) не(известно(b)))))))", "или(и(тип(доказать) условие) не(контекст(операнд(x5 теквхожд) символ(x5 не))))", "или(константа(c)не(константа(a))не(константа(b)))", "не(контекст(подчинено(теквхожд x4) символ(x4 длялюбого существует класс отображение) входит(c связприставка(x4))не(входит(c a)) не(входит(c b))))". Хотя процесс обучения справочника продолжается, уже в этом виде он обычно дает приемлемые приемы.

Число приемов данного типа - 76.

ii. Конъюнктивная декомпозиция элементарной посылки.

Здесь декомпозируется посылка, что приводит к расформированию ее в группу других посылок. Уровень срабатывания у приемов данного типа меньше, чем у предыдущего типа. Пример - теорема из предыдущего пункта. Вместо длинного списка фильтров ее сопровождают лишь два - "посылка" и "корень". Уровень срабатывания приема здесь равен 0, причем прием преобразует даже утверждения, сопровождающие по о.д.з.

Спецификация приема имеет вид "тип(трансlзамена)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 0, вводит фильтры "посылка" и "корень", а также указатель "сопровождение". Иногда создаются дополнительные фильтры. Обычно этих возможностей достаточно.

Число приемов данного типа - 22.

(b) Дизъюнктивная декомпозиция элементарного утверждения.

i. Дизъюнктивная декомпозиция элементарного утверждения.

Пример:

$\forall_{abc}(a \in b \cup c \leftrightarrow a \in b \ \vee \ a \in c)$

Спецификация приема имеет вид "тип(сверткавариантa)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "или(посылка не(тип(описать)) и(не(цель(свертка)) или(не(известно(корень)) и(не(цель(соединение)) или(не(цель(редакция)) не(цель(и))))) или(цель(развертка) и(или(не(неизвестная(x1))

не(цель(редакция))) или(известно(a) не(известно(b)) не(известно(c)))))))", "конец(или(не(корень) и(условие или(не(тип(преобразовать)) не(цель(посылки))))))", "или(константа(a) не(константа(b)) не(константа(c)))", "не(контекст(подчинено(теквхожд x4) символ(x4 длялюбого существует класс отображение) входит(a связприставка(x4)) не(входит(a b)) не(входит(a c))))". Этого почти достаточно.

Число приемов данного типа - 12.

(c) Дизъюнктивно-конъюнктивная декомпозиция элементарного утверждения.

i. Дизъюнктивно-конъюнктивная декомпозиция элементарного утверждения.

Пример:

$\forall_{abcd}(a \times b \subseteq c \times d \leftrightarrow a \subseteq c \ \& \ b \subseteq d \ \vee \ a = \emptyset \ \vee \ b = \emptyset)$

Спецификация приема имеет вид "тип(множество)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтр "или(посылка не(тип(описать))и(не(цель(свертка)) или(не(известно(корень)) не(цель(соединение))))))". В других примерах список фильтров пополняется дополнительными элементами. Создаваемые приемы в большинстве случаев уже приемлемы.

Число приемов данного типа - 12.

ii. Дизъюнктивно-конъюнктивная декомпозиция в условии задачи на описание.

Вариант общего случая дизъюнктивно-конъюнктивной декомпозиции, выбор которого определяется примеркой на задачах. Пример:

$\forall_{ab}(0 \leq a \ \& \ 0 \leq b \rightarrow a + b = 0 \leftrightarrow a = 0 \ \& \ b = 0)$

Декомпозируемое равенство - условие задачи на описание либо расположено под корневым отрицанием в условии. Выражение a имеет своим сомножителем степенное выражение с неконстантным основанием. Данная попытка декомпозиции чаще оказывается успешной в условиях, нежели в посылках; этим и обусловлен выбор данного типа приема.

Спецификация приема имеет вид "тип(дескриптор)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "условие", "тип(описать)", "отрицание". Он требует доработки.

Число приемов данного типа - 6.

iii. Дизъюнктивно-конъюнктивная декомпозиция в условии задачи на доказательство.

Некоторые случаи декомпозиции имеют смысл, главным образом, в условиях задач на доказательство. Пример:

$\forall_{ABCD}$(параллелограмм($ABCD$) $\leftrightarrow$ прямая(AB) $\parallel$ прямая(CD) & прямая(BC) $\parallel$ прямая(AD))

В случае посылки вместо данной эквивалентной замены работает вывод следствий.

Спецификация приема имеет вид "тип(констдробь)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3, вводит фильтры "условие", "тип(доказать)"и указатель "сопровождение". Этого почти достаточно.

Число приемов данного типа - 27.

5. Ориентация равенства.

   (a) Ориентация равенства, приводящая к исключению понятия.

   Прием ориентирует равенство, выражающее сложное понятие через более простые, таким образом, чтобы общий прием применения этого равенства для стандартизации обозначений исключил данное понятие. Пример:

   $\forall_{ab}(a = \text{card}(b) \leftrightarrow \text{card}(b) = a)$

   Перестановка частей равенства при идентификации заблокирована. Выражение a не содержит символа "мощность" и не содержит невырожденных числовых атомов. Преобразуемое утверждение представляет собой посылку задачи. В приемах данного типа замена всегда происходит слева направо.

   Спецификация приема имеет вид "тип(Равно)". Иногда имеются сопровождающие элементы, уточняющие вид заменяющего терма:

   i. числпарам. Проверяется отсутствие в заменяющем терме невырожденных числовых атомов.
   ii. символы($s_1, \ldots, s_n$). Заменяющий терм не содержит символов $s_1, \ldots, s_n$.
   iii. параметр. Заменяемый терм содержит несущественную неизвестную, а заменяющий - не содержит наких неизвестных.
   iv. заголовок($s_1, \ldots, s_n$). Перестановка выполняется несмотря на блокирующий ее комментарий "ориентацияравенства", если заменяющий терм будет иметь своим заголовком один из символов $s_1, \ldots, s_n$.
   v. переменная. Заменяющий терм представляет собой переменную.
   vi. неизвестная. Заменяющий терм представляет собой неизвестную.

   Справочник "заголовокприема" указывает уровень срабатывания 0, вводит фильтры "посылка", "корень" и дополнительные фильтры, соотвественно указаниям на вид заменяющего терма. Кроме того, вводятся указатели "коммутативно(фикс(0 1))" и "примечание(ориентацияравенства)". Последний комментарий блокирует действия общего приема, ориентирующего равенства из своих соображений. Указанных действий справочника обычно достаточно.

   Число приемов данного типа - 39.

   (b) Ориентация равенства, приводящая к использованию понятия.

   Прием ориентирует равенство таким образом, чтобы стандартизация с его помощью обеспечивала появление понятий, на которые ориентированы прочие приемы решателя. Пример:

   $\forall_{ABa}(\text{отрезок}(AB) = a \leftrightarrow a = \text{отрезок}(AB))$

Переменная a идентифицируется с переменной. Прием применяется к посылке задачи на исследование.

Спецификация приема имеет вид "тип(возрастаниедлин)". Иногда имеются сопровождающие элементы, уточняющие вид заменяемого терма:

i. переменная. Заменяемый терм - переменная, не входящая в заменяющий терм.
ii. вспомпараметр. Заменяемый терм содержит переменную, выделенную комментарием "вспомпараметр".
iii. числовойатом(n). Находится n-й операнд заменяющего терма. Как он, так и заменяемый терм должны содержать невырожденный числовой атом, а прочие операнды заменяющего терма - не должны.

Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "тип(исследовать)", "корень", а также дополнительные фильтры согласно условиям на вид заменяемого терма. Кроме того, вводятся указатели "коммутативно(фикс(0 1))" и "примечание(ориентацияравенства)". Обычно этого достаточно.

Число приемов данного типа - 8.

6. Усиление утверждения.

(a) Усиление утверждения.

Вообще говоря, целесообразность усиления утверждения зависит от контекста. Если это утверждение требуется доказать, то усиление его может лишь усложнить задачу; если это утверждение - посылка, то задача будет упрощена. В данном типе собраны случаи, когда усиление, по-видимому, безопасно в любом контексте. Пример:

$\forall_{abx}(a \le x \ \& \ x - \text{целое} \ \& \ b - a = 1 \to x < b \leftrightarrow x = a)$

Переменные a, b идентифицируются с целочисленными константами. Первый антецедент идентифицируется с утверждением из контекста.

Спецификация приема имеет вид "тип(занесениеусловия)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. Он требует доработки.

Число приемов данного типа - 5.

(b) Усиление посылки.

Пример:

$\forall_n(n - \text{целое} \to 0 < n \leftrightarrow 0 \le n - 1)$

Заменяемое утверждение расположено в посылке, причем не внутри подтерма с заголовком "не". Существует посылка с заголовком "целое", "натуральное" либо "четное". Антецедент обрабатывается проверочным оператором.

Спецификация приема имеет вид "тип(явное)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтр

"или(посылка контекст(подчинено(теквхожд x1)символ(x1 отображение)) и(тип(описать) контекст(ключ(цели серия x1)пересекаются(x1 параметры(теквхожд)))))". Он требует доработки. Заметим, что указанный фильтр допускает применение приема не только в посылках, но и в ряде других контекстов.

Число приемов данного типа - 20.

7. Константные выражения.

   (a) Общая стандартизация, использующая вычисления с константными термами.

   Пример:

   $\forall_{kmnp}(p = \text{нок}(m,n) \to m|k \ \& \ n|k \leftrightarrow p|k)$

   Переменные m, n идентифицируются с натуральными константами. Антецедент выделен указателем "программа".

   Спецификация приема имеет вид "тип(титр)", "направл(N)", "см(...)", где последний элемент фиксирует типы константных значений переменных. Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры, задающие типы константных значений. Вводится также указатель "программа" на обработку антецедента. Этого достаточно.

   Число приемов данного типа - 9.

8. Общая стандартизация с исключением квантора.

   (a) Общая стандартизация с исключением квантора.

   К этому типу отнесены приемы, которые не просто исключают квантор, но еще и не переходят к сложным понятиям. Пример:

   $\forall_m(m - \text{натуральное} \to \forall_n(n - \text{натуральное} \to m|n) \leftrightarrow m = 1)$

   Спецификация приема имеет вид "тип(пример)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтр "или(посылка не(тип(описать)) не(цель(развертка)))". Обычно этого достаточно.

   Число приемов данного типа - 44.

   (b) Общая стандартизация с частичным исключением квантора.

   К этому типу относятся преобразования типа общей стандартизации, уменьшающие число кванторов либо число переменных связывающей приставки. Пример:

   $\forall_{ABm}(m - \text{натуральное} \to \forall_k(k - \text{натуральное} \ \& \ m \leq k \to \exists_x(0 \leq k + A(x) \ \& \ B(x))) \leftrightarrow \exists_x(0 \leq m + A(x) \ \& \ B(x)))$

   Спецификация приема имеет вид "тип(хэшзадачи)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1. Этого почти достаточно.

   Число приемов данного типа - 5.

(c) Развертка квантора существования в дизъюнкцию.

Пример:

$\forall_{fmkpx}(p = k - m \ \& \ k - \text{целое} \ \& \ m - \text{целое} \to \exists_n(n - \text{целое} \ \& \ f(n) \ \& \ m \leq n \ \& \ n \leq k) \leftrightarrow \exists_l(l \in \{0, \ldots, p\} \ \& \ f(m + l)))$

Первый антецедент выделен указателем "идентификатор". После упрощения p оказывается десятичным числом, не превосходящим 8. Заменяющий квантор существования выделен указателем "или", разворачивающим его в дизъюнкцию.

Спецификация приема имеет вид "тип(провменьше)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Он не прорабатывался.

Число приемов данного типа - 9.

(d) Развертка квантора общности в конъюнкцию.

Пример:

$\forall_{Pkmn}(k = n - m + 1 \to \forall_i(i \in \{m, \ldots, n\} \to P(i)) \leftrightarrow \forall_j(j \in \{1, \ldots, k\} \to P(m + j - 1)))$

Переменные k, m, n идентифицируются с целочисленными константами, причем k меньше 9.

Спецификация приема имеет вид "тип(таблица)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Он не прорабатывался.

Число приемов данного типа - 5.

9. Общая стандартизация с исключением описателя.

Пример:

$\forall_{abc}(\text{card}(\text{set}_x(x - \text{число} \ \& \ ax^2 + bx + c = 0)) = 2 \leftrightarrow 0 < b^2 - 4ac)$

Спецификация приема имеет вид "тип(списокзадач)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Иногда этого достаточно.

Число приемов данного типа - 7.

10. Свертка дизъюнкции.

Пример:

$\forall_{ab}(a = b \ \vee \ a < b \leftrightarrow a \leq b)$

Спецификация приема имеет вид "тип(полный)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтр "или(не(корень)и(условие не(тип(описать)))комментусловия(разборслучаев))". Вводится также указатель "дизъюнктоперанд". Этого достаточно.

Число приемов данного типа - 8.

11. Свертка конъюнкции.

    Пример:

    $\forall_{ab}(\neg(a = b)\ \&\ a \leq b \leftrightarrow a < b)$

    Спецификация приема имеет вид "тип(блоктеорем)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. Этого почти достаточно.

    Число приемов данного типа - 10.

Общая стандартизация группы утверждений

1. Общая стандартизация группы посылок.

   Пример:

   $\forall_{ab}(\neg(a = b)\ \&\ a \leq b \leftrightarrow a < b)$

   Спецификация приема имеет вид "тип(заменатермов)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтр "посылка". Обычно этого достаточно.

   Число приемов данного типа - 8.

2. Общая стандартизация группы условий задачи на описание.

   Хотя общую стандартизацию группы условий или группы посылок может задавать одна и та же теорема, но заголовки приемов здесь должны быть разными. Поэтому и типы приемов различны. Пример:

   $\forall_{abc}(\neg(b \in \{;c\}) \rightarrow \neg(a = b)\ \&\ a \in \{b;c\} \leftrightarrow a \in \{;c\})$

   Прием применяется к паре условий задачи на описание. Антецедент обрабатывается проверочным оператором.

   Спецификация приема имеет вид "тип(заменаусловия)". "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "условие", "тип(описать)". Этого достаточно.

   Число приемов данного типа - 11.

Сокращенная переформулировка

1. Дизъюнктивно - конъюнктивная свертка в условии задачи на свертку.

   Пример:

   $\forall_{abc}(a \in b \cup c \leftrightarrow a \in b\ \vee\ a \in c)$

   Замена выполняется справа налево. Заменяемая дизъюнкция входит в условие задачи на описание, либо имеющей цель "свертка", либо - цель "соединение"; в последнем случае дизъюнкция не должна содержать неизвестных. Переменные дизъюнкции не связаны внешними кванторами и описателями.

Спецификация приема имеет вид "тип(сборка)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "или(цель(свертка)и(цель(соединение)известно(a)известно(b) известно(c)))", "свобоперанд(фикс(0 2 1))", "свобоперанд(фикс(0 2 2))". Этого достаточно.

Число приемов данного типа - 27.

2. Сокращенная переформулировка группы условий задачи на свертку.

   Пример:

   $\forall_{abc}(\text{непересек}(c, a \cup b) \leftrightarrow \text{непересек}(a, c)\ \&\ \text{непересек}(b, c))$

   Замена выполняется справа налево, причем конъюнктивные члены заменяемой части идентифицируются с условиями задачи на описание, имеющей цель "свертка" либо "соединение".

   Спецификация приема имеет вид "тип(соединение)", "направл(N)". Справочник "заголовокприема" указывает уровни срабатывания 1,3 и 5, а также вводит фильтры "условие", "тип(описать)", "или(цель(свертка) и(цель(соединение) известно(a) известно(b) известно(c)))". Этого достаточно.

   Число приемов данного типа - 22.

3. Свертка группы известных условий задачи на описание при редактировании ответа.

   Пример:

   $\forall_{ab}(a + b < 0\ \&\ 0 < a - b \leftrightarrow b + |a| < 0)$

   Конъюнктивные члены левой части идентифицируются с условиями задачи на описание, имеющей цель "редакция". Они не содержат неизвестных и не используются для сопровождения по о.д.з. Левая часть первого из них не имеет вида "$AX + B$", где A, B - константы, X - переменная.

   Спецификация приема имеет вид "тип(комментариийусловия)", "направл(N)". Справочник "заголовокприема" указывает уровни срабатывания 2,4,6 и вводит фильтры "условие", "тип(описать)", "цель(редакция)", "не(сопровождение)", "не(цель(или))", "известно(a)", "известно(b)". Он недоработан.

   Число приемов данного типа - 6.

4. Упрощение известного условия для параметров при редактировании ответа задачи на описание.

   Пример:

   $\forall_{abcmn}(a + bc^{m/n} = 0 \leftrightarrow a^n + b^n c^m = 0)$

   Заменяемый терм находится под корневым отрицанием в условии задачи на описание, имеющей цель "редакция". Он не содержит неизвестных. Переменные m, n идентифицируются с натуральными константами, причем n - с нечетной

константой. Проверяется, что после стандартизации заменяющий терм короче заменяемого. Замена выполняется, даже если преобразуемое условие используется для сопровождения по о.д.з.

Спецификация приема имеет вид "тип(равнойдлины)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(описать)", "условие", "цель(редакция)", "известно(теквхожд)". Он недоработан.

Число приемов данного типа - 10.

Кванторные свертки и расшифровки

1. Кванторная свертка.

   Приемы этого типа преобразуют кванторные утверждения в элементарные. Пример:

   $\forall_{ab}(a \subseteq b \leftrightarrow \forall_c(c \in a \rightarrow c \in b))$

   Преобразование применяется справа налево. Прием имеет множество фильтров, так как данное преобразование не всегда полезно. Практически все эти фильтры синтезируются автоматически (см. ниже).

   Спецификация приема имеет вид "тип(кванторнаясвертка)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "или(не(тип(описать)) и(или(посылка и(или(не(цель(редакция)) не(цель(попыткапараметризации))) не(цель(учетвответа)) не(цель(вычисление)) комментусловия(параметризация) комментусловия(серия))) не(цель(развертка))))", "коммент(кванторнаясвертка)", "или(и(условие не(тип(описать))) комментусловия(кванторнаясвертка))", "или(не(тип(преобразовать)) заголовок(теквхожд длялюбого) не(цель(посылки)))", "или(и(не(тип(преобразовать)) не(тип(доказать))) не(посылка) заголовок(корень длялюбого) не(корень))". Для почти всех приемов данного типа этого достаточно.

   Число приемов данного типа - 51.

2. Кванторная расшифровка.

   Приемы данного типа преобразуют бескванторное утверждение (возможно, с описателями) в утверждение с кванторами. Имеется множество частных случаев данного типа, приводимых ниже. Иногда они вводятся доводчиком, если прием общего вида приводит к нежелательным явлениям. Иногда - дополняют прием общего вида, но имеют другой уровень срабатывания. Пример приема данного типа:

   $\forall_{afg}(\neg(\mathrm{set}_x f(x) = \emptyset) \rightarrow a \in \bigcap_{x,f(x)} g(x) \leftrightarrow \forall_x(f(x) \rightarrow a \in g(x)))$

   Прием имеет множество фильтров, однако практически все они снтезируются автоматически.

   Спецификация приема имеет вид "тип(теквхожд)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры

"или(тип(доказать) и(тип(описать) или(цель(развертка) и(не(цель(редакция)) или(посылка не(известно(теквхожд)))))))", "или(посылка не(тип(описать)) цель(развертка) и(или(не(неизвестная(a)) не(цель(редакция))) или(известно(a) не(известно(фикс(0 1 2))))))", "не(контекст(подчинено(теквхожд x2) символ(x2 длялюбого существует класс отображение) входит(a связьприставка(x2)) не(входит(a фикс(0 1 2)))))". Вводится также указатель "замечание(кванторная свертка)". Почти всегда этого достаточно.

Число приемов данного типа - 7.

3. Кванторная расшифровка в условии задачи на доказательство.

Появление квантора в условии задачи на доказательство обычно сразу же приводит к его устранению. Поэтому приемы, выполняющие кванторную расшифровку в данном контексте, имеют определенный приоритет и выделяются в отдельный тип. Обычно приемы данного типа имеют большой уровень срабатывания, так как во многих случаях доказать утверждение удается без расшифровки. Пример:

$\forall_{ab}(a \subseteq b \leftrightarrow \forall_c(c \in a \rightarrow c \in b))$

Прием применяется к подутверждению условия задачи на доказательство - корневому либо расположенному под корневым отрицанием. В первом случае уровень срабатывания равен 6, иначе - 7.

Спецификация приема имеет вид "тип(развертка)", "направл(N)". Справочник "заголовокприема" указывает уровни срабатывания 6 и 7. Вводятся фильтры "условие", "тип(доказать)", "отрицание", "альтернатива(корень уровень(6) уровень(7))" и указатель "замечание(кванторнаясвертка)". Обычно этого достаточно.

Число приемов данного типа - 74.

4. Кванторная расшифровка посылки, приводящая к квантору общности.

Пример:

$\forall_{AGaf}(f = \text{операция}(G)\ \&\ a \in \text{носитель}(G)\ \&\ A \subseteq \text{носитель}(G)\ \&\ \text{группа}(G) \rightarrow \text{порождэлемент}(a, A, G) \leftrightarrow a \in A\ \&\ \forall_x(x \in A \rightarrow \exists_n(n{-}\text{целое}\ \&\ x = \text{алгстепень}(a, f, n))))$

Прием применяется к посылке. Уровень срабатывания равен 4. Фактически, это просто расшифровка по определению. Такого рода приемы создаются "на первое время", пока не накоплены средства для работы с понятием, не прибегающие к его расшифровке. В дальнейшем либо увеличивается уровень срабатывания этих приемов, либо они вообще удаляются.

Спецификация приема имеет вид "тип(задача)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 8, вводит фильтры "посылка", "корень" и указатели "примечание(кванторнаясвертка)", "примечание(контрольвывода)". Он недоработан.

Число приемов данного типа - 3.

5. Кванторная расшифровка посылки, приводящая к квантору существования.

   Примеры:

   $\forall_{mn}(m|n \leftrightarrow \exists_k(k - \text{целое } \& \ n = mk))$

   Прием применяется к посылке задачи на доказательство, либо на исследование, либо задачи на описание, не имеющей цели "прямойответ". Вводится комментарий "кванторнаясвертка". Уровень срабатывания равен 6.

   $\forall_{Ax}(A \subseteq \mathbb{R} \ \& \ x - \text{число} \rightarrow \text{граничнточка}(x, A) \leftrightarrow \forall_b(b - \text{число} \ \& \ 0 < b \rightarrow \exists_y(y \in A \ \& \ |x - y| < b) \ \& \ \exists_y(y - \text{число} \ \& \ \neg(y \in A) \ \& \ |x - y| < b)))$

   Аналогично предыдущему, но преобразуемое утверждение расположено под корневым отрицанием посылки.

   Спецификация приема имеет вид "тип(облнорм)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 6, вводит фильтры "посылка", "корень" либо (для второго примера) "отр", "или(тип(доказать)тип(исследовать) и(тип(описать)не(цель(прямойответ))))" и указатель "примечание(кванторнаясвертка)". Этого почти достаточно.

   Число приемов данного типа - 17.

6. Кванторная расшифровка утверждения с описателями.

   (a) Кванторная расшифровка утверждения с описателями.

   Пример:

   $\forall_{Pb}(\text{нижняягрань}(b, \text{set}_x(P(x))) \leftrightarrow b - \text{число} \ \& \ \forall_c(P(c) \rightarrow b \leq c))$

   Спецификация приема - "тип(точкапрямой)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "или(посылка не(тип(описать)) и(не(цель(свертка)) или(цель(развертка) и(или(не(неизвестная(b))не(цель(редакция))) или(известно(b) не(известно(фикс(0 1 2))))))))", "не(контекст(подчинено(теквхожд x1) символ(x1 для-любого существует класс отображение) входит(b связприставка(x1)) не(входит(b фикс(0 1 2)))))". Вводятся также указатели "примечание(условие(или(посылка тип(описать)) кванторнаясвертка))", "примечание(условие(или(посылка тип(описать)) смантецеденты))". В большинстве случаев этого достаточно.

   Число приемов данного типа - 19.

7. Кванторная расшифровка под квантором.

   (a) Кванторная расшифровка под квантором.

   Пример:

   $\forall_{Aa}(A - \text{set} \ \& \ A \subseteq \mathbb{R} \rightarrow \text{предельнточка}(a, A) \leftrightarrow a - \text{число} \ \& \ \forall_x(x - \text{число} \ \& \ 0 < x \rightarrow \exists_y(y \in A \ \& \ \neg(y = a) \ \& \ |a - y| < x)))$

Преобразуемое утверждение расположено внутри квантора общности либо существования. Уровень срабатывания равен 7.

Спецификация приема имеет вид "тип(нормчислитель)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 4, вводит фильтр "контекст(подчинено(теквхожд x2) символ(x2 длялюбого существует))" и указатель "замечание(кванторнаясвертка)". Этого почти достаточно.

Число приемов данного типа - 9.

8. Кванторная расшифровка в условии задачи на описание, не имеющей неизвестных.

Задачи на описание без неизвестных решаются для установления истинности либо ложности конъюнкции своих условий. Для типичных проверяемых утверждений кванторная расшифровка может оказаться естественным первым шагом. Пример:

$\forall_{Af}$(бинарнаяоперация$(f, A) \rightarrow$ правсократимо$(f) \leftrightarrow \forall_{xyz}(x \in A \ \& \ y \in A \ \& \ z \in A \ \& \ f(x, z) = f(y, z) \rightarrow x = y)$)

Прием применяется в условии задачи на описание, не имеющей неизвестных, но имеющей цель "проверка" либо "развертка". Задача не должна иметь цель "исследовать". Антецедент использует пакетный синтезатор для определения области A операции f.

Спецификация приема имеет вид "тип(неравенства)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "условие", "тип(описать)", "не(цель(исследовать))", "свобоперанд(теквхожд)", "не(Входит(неизвестные цели))", "или(цель(проверка)цель(развертка))". Вводится также указатель "замечание(кванторнаясвертка)". Этого достаточно.

Число приемов данного типа - 12.

9. Кванторная расшифровка в режиме развертки.

Пример:

$\forall_{ab}(a \subseteq b \leftrightarrow \forall_c(c \in a \rightarrow c \in b))$

Прием применяется в условии задачи на описание, имеющей цель "развертка".

Спецификация приема имеет вид "тип(контрольбуфера)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие", "цель(развертка)". Этого достаточно.

Число приемов данного типа - 9.

Применение параметрического описания

1. Попытка использования явного параметрического описания при получении частичного ответа.

   Использование параметрических описаний целесообразно в задачах на поиск примера либо на получение именно параметрического описания. В обычных задачах, где требуется найти все значения неизвестных, они, как правило, лишь затрудняют решение. Пример:

   $\forall_{ab}(a \subseteq b \leftrightarrow \exists_c(c - \text{set } \& \ b = a \cup c))$

   Прием имеет заголовок "параметризация". Он применяется в условиях задач на описание, имеющих цель "пример" либо "параметризация". Переменная b - неизвестная, причем она не входит в выражение a.

   Спецификация приема имеет вид "тип(параметризация)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "условие", "тип(описать)", "или(цель(пример) цель(параметризация))", "неизвестная(b)", "не(входит(b a))". Этого обычно достаточно.

   Число приемов данного типа - 18.

2. Попытка использования неявного параметрического описания при получении частичного ответа.

   Аналогично предыдущему, но параметрическое описание неявное. Пример:

   $\forall_{ab}(\neg(\text{непересек}(a, b)) \leftrightarrow \exists_c(c \in a \ \& \ c \in b))$

   Прием применяется к подутверждению содержащего неизвестные условия задачи на описание, имеющей цель "пример" либо "параметризация".

   Спецификация приема - "тип(попыткапараметризации)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "условие", "тип(описать)", "или(цель(пример)цель(параметризация))", "не(известно(корень))". Этого достаточно.

   Число приемов данного типа - 3.

Преобразование утверждений с неизвестными

1. Разрешение относительно неизвестных.

   (a) Разрешение условия либо посылки относительно заданных неизвестных.

      i. Разрешение условия задачи на описание либо посылки задачи на исследование относительно заданных неизвестных.

         К этому типу относятся лишь наиболее простые случаи разрешения относительно неизвестных. Для прочих случаев разрешение предпринимается лишь в условии задачи на описание. Пример приема данного типа:

         $\forall_{abc}(0 < c \rightarrow ac \leq b \leftrightarrow a \leq b/c)$

Переменная a идентифицируется с непустым произведением всех содержащих неизвестные сомножителей. Выражение b не содержит неизвестных. Имеется ряд дополнительных фильтров.

Спецификация приема имеет вид "тип(глуб)", "направл(N)", "неизвестные(X)", где X - набор переменных, идентифицируемых с содержащими неизвестные выражениями. Справочник "заголовокприема" указывает уровни срабатывания 0 и 1. Он также вводит фильтры "или(и(тип(описать)условие) тип(исследовать))", "альтернатива(или(и(тип(исследовать)цель(известно)) заголовок(корень или)) уровень(0) уровень(1))", "не(цель(редакция))", "внешзнак(и или существует)", "не(известно(a))", "известно(b)", "или(не(заголовок(корень или)) комментусловия(случаи))", "не(контекст(ключ(цели независит x4) пересекаются(x4 c)))", "или(не(контекст(подчинено(теквхожд x4) символ(x4 существует)не(парамописание(x4)))) не(комментусловия(серия)))". Вводится указатель "перечень(a не(известно(a)))". Как правило, действий этого справочника достаточно. В отдельных случаях требуется доработка.

Число приемов данного типа - 36.

ii. Разрешение условия задачи на описание относительно заданных неизвестных.

A. Непосредственное разрешение условия задачи на описание относительно заданных неизвестных.

Пример:

$\forall_{abcde}(e = b^2 + 4ac \rightarrow ad^2 + bd = c \leftrightarrow \neg(a = 0)\ \&\ 0 \leq e\ \&\ (d = (\sqrt{e} - b)/(2a)\ \vee\ d = -(\sqrt{e} + b)/(2a))\ \vee\ bd = c\ \&\ a = 0)$

Прием решения квадратного уравнения, применяемый к подутверждению условия задачи на описание. Выражение d содержит неизвестные, выражения a, b, c - не содержат.

Спецификация приема имеет вид "тип(Делители)", "направл(N)", "неизвестные($x_1 \ldots x_k$)". Переменные $x_1, \ldots, x_k$ идентифицируются с выражениями, содержащими неизвестные. Остальные переменные - с известными выражениями. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(описать)", "условие", "или(неизвестные(2) не(контекст(новоеусловие(x6)не(известно(x6))равно(x7 сложнреш(x6))равно(x8 сложнреш(теквхожд)) или(и(равно(число(x7)число(x8)) меньше(x7 10)короче(x6 корень)) меньше(x7 x8)))))" (нет другого уравнения, более предпочтительного для разрешения относительно той же неизвестной), "не(цель(редакция))", "внешзнак(и или существует длялюбого)", "не(известно(d))", "одинквантор(теквхожд)" (уравнение не расположено внутри двух различных кванторов), "известно(a)", "известно(b)", "известно(c)", "не(контекст(подчинено(теквхожд x6) символ(x6 длялюбого) не(контекст(подчинено(теквхожд последнийоперанд(x6))))))", "не(контекст(список(x6 a b) ключ(цели независит x7) пересекаются(x7 x6)))", "или(не(контекст(подчинено(теквхожд x6) символ(x6 существует) не(парамописание(x6)))) не(ком-

ментусловия(серия)))", "или(не(цель(пример)) и(свобоперанд(теквхожд) независит(теквхожд)))". В большинстве случаев этого достаточно, причем создается даже больше фильтров, чем в приемах, введенных вручную.

Число приемов данного типа - 76.

B. Непосредственное разрешение условия задачи на описание относительно заданных неизвестных - корневой случай.

В отдельных случаях (как правило, при существенном усложнении утверждения после разрешения относительно неизвестной) прием применяется только к самому условию, а не к его подтерму. Пример:

$\forall_{abcdepqr}(d = b^2 + 4ac \ \& \ p = \sqrt{d} \ \& \ q = (-b - p)/(2a) \ \& \ r = (-b + p)/(2a) \to c \le ae^2 + be \leftrightarrow a < 0 \ \& \ 0 \le d \ \& \ (p < 0 \ \& \ q \le e \ \& \ e \le r \ \vee \ 0 \le p \ \& \ r \le e \ \& \ e \le q) \ \vee \ 0 < a \ \& \ (d < 0 \ \vee \ 0 \le d \ \& \ (p < 0 \ \& \ (e \le r \ \vee \ q \le e) \ \vee \ 0 \le p \ \& \ (e \le q \ \vee \ r \le e))) \ \vee \ a = 0 \ \& \ c \le be)$

Прием разрешает квадратное неравенство - условие задачи на описание. Выражение e содержит неизвестные, выражения a, b, c - не содержат.

Спецификация приема имеет вид "тип(фильтрудвоения)", "направл(N)", "неизвестные($x_1 \dots x_k$)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "тип(описать)", "условие", "не(цель(редакция))", "корень", "не(известно(e))", "известно(a)", "известно(b)", "известно(c)", "не(контекст(список(x6 a b) ключ(цели независит x7) пересекаются(x7 x6)))". Почти всегда этого достаточно.

Число приемов данного типа - 26.

C. Непосредственное разрешение условия задачи на описание относительно заданных неизвестных, если прочие операнды суть константные выражения.

Приемы такого типа создаются в случае особо сложных выражений, появляющихся при разрешении. Пример:

$\forall_{Aabcdpqrx}(\neg(a = 0) \ \& \ p = (3ac - b^2)/(3a^2) \ \& \ q = (2b^3 - 9abc - 27a^2d)/(27a^3) \ \& \ r = q^2/4 + p^3/27 \ \& \ 0 < r \to ax^3 + bx^2 + cx = d \leftrightarrow x = \sqrt[3]{-q/2 + \sqrt{r}} + \sqrt[3]{-q/2 - \sqrt{r}} - b/(3a))$

Прием для решения кубического уравнения применяется к условию задачи на описание. Выражение x содержит неизвестные, выражения a, b, c, d константные.

Спецификация приема имеет вид "тип(последнийсимвол)", "направл(N)", "неизвестные($x_1 \dots x_k$)". Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "тип(описать)", "условие", "корень", "не(известно(x))", "константа(a)", "константа(b)", "константа(c)", "константа(d)". Этого достаточно. Заметим, что для решения кубических уравнений с неконстантными коэффициентами тоже можно было бы создать прием, например,

предшествующего типа, но зато с гораздо большим уровнем срабатывания.

Число приемов данного типа - 4.

D. Усмотрение независимости истинности условия от неизвестной.

Пример:

$\forall_{abcde}(d = b^2 + 4ac \ \& \ d < 0 \to c < ae^2 + be \leftrightarrow 0 < a)$

При отрицательном дискриминанте истинность квадратного неравенства оказывается не зависящей от значения неизвестной. Прием применяется к подутверждению задачи на описание. Выражение e содержит неизвестные, выражения a, b, c - не содержат.

Спецификация приема имеет вид "тип(невозрастает)", "направл(N)", "неизвестные($x_1, \ldots, x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "тип(описать)", "условие", "не(цель(редакция))", "корень", "не(известно(e))", "известно(a)", "известно(b)", "известно(c)", "не(контекст(список(х6 a b) ключ(цели независит х7) пересекаются(х7 х6)))". Он требует доработки.

Число приемов данного типа - 4.

E. Разрешение условия задачи на описание, имеющей цель "функционально".

Цель "функционально" указывает, что необходимо установить лишь однозначную определимость значений неизвестных по известным параметрам, а сами эти значения несущественны. Соответственно, становится допустимым использование различных искусственных понятий, которые в обычных ситуациях не должны появляться в ответе. Например, понятие "элемент(A)" - выбор какого-то неопределенного элемента множества A. Пример приема:

$\forall_{Afx}(\neg(A = \emptyset) \to f = \text{конст}(A, x) \leftrightarrow x = f(\text{элемент}(A)))$

Прием применяется к условию задачи на описание, имеющей цель "функционально". Выражение x содержит неизвестные, выражения A, f - не содержат.

Спецификация приема имеет вид "тип(список)", "направл(N)", "неизвестные($x_1, \ldots, x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие", "корень", "цель(функционально)", "не(известно(x))", "известно(f)", "известно(A)". Этого достаточно.

Число приемов данного типа - 4.

F. Условие задачи на описание преобразуется в явное параметрическое описание неизвестных.

Пример:

$\forall_{ab}(\sin a = b \leftrightarrow 0 \le b + 1 \ \& \ 0 \le 1 - b \ \& \ (\exists_n(n - \text{целое} \ \& \ a = \arcsin b + 2\pi n) \ \vee \ \exists_n(n - \text{целое} \ \& \ a = -\arcsin b + \pi + 2\pi n)))$

Прием применяется к условию задачи на описание. Выражение a содержит неизвестные, а b - не содержит.

Спецификация приема имеет вид "тип(серия)", "направл(N)", "неизвестная(x)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(описать)", "условие", "корень", "не(Входит(известно цели))", "не(известно(a))", "известно(b)". Вводится также указатель "примечание(серия)". Справочник требует доработки.

Число приемов данного типа - 57.

G. Условие задачи на описание преобразуется в явное параметрическое описание функциональных неизвстных.

Пример:

$\forall_{adkmprsy}(\neg(a(m{+}1)=0)\ \&\ p=\sum_{j=1}^{m+1}a(j)n^{j-1}\ \&$ Базисрешений(p,d) $\&\ d=\{;r\}\ \&\ l(r)=k\to\forall_n(n-$ натуральное $\&\ s\le n\to$ $\sum_{j=1}^{m+1}a(j)y(n+j-1)=0)\to\exists_c(\forall_n(n-$ натуральное $\&\ s\le n\to$ $y(n)=\sum_{i=1}^{k}c(i)r(i))\ \&\ \forall_i(i\in\{1,\ldots,k\}\to c(i)-$ число$)))$

Прием применяется к условию задачи на описание. Переменная y идентифицируется с неизвестной, переменная m - с натуральной константой. Конечные суммы разворачиваются в обычные, последний квантор общности - в конъюнкцию.

Спецификация приема имеет вид "тип(заголовок)", "направл(N)", "неизвестная(y)". Могут добавляться элементы "см" и "указатель". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(описать)", "условие", "корень", "не(Входит(известно цели))", "неизвестная(y)". Он требует доработки.

Число приемов данного типа - 3.

H. Использование нормализатора вычислений для явного разрешения относительно неизвестных условия задачи на описание.

Пример:

$\forall_{Aabc}($собствзначение$(A,a,b)=((a,b)\in c)\to$ собствзначение(A,a,b) $\leftrightarrow(a,b)\in c)$

Прием применяется к условию задачи на описание. Хотя бы одно из выражений a,b содержит неизвестные. Выражение A не содержит неизвестных и имеет заголовок "строки" либо "столбцы", т.е. в явном виде определяет матрицу. Левая часть антецедента обрабатывается нормализатором "собствзначения", определяющим собственные значения матрицы. Он выдает утверждение вида $(a,b)\in c$, где c - конечный список пар (собственное значение - его кратность).

Спецификация приема имеет вид "тип(нормсуммавсех)", "направл(N)", "неизвестные($x_1,\ldots,x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(описать)", "условие", "корень", "или(не(известно(a))не(известно(b)))", "известно(A)". Он требует доработки.

Число приемов данного типа - 7.

I. Разрешение кванторного условия задачи на описание относительно неизвестных с помощью ввода вспомогательного известного описателя.

Пример:

$\forall_{ABy}(\forall_x(A(x)\ \&\ \neg(x \in y) \to B(x)) \leftrightarrow \mathrm{set}_x(A(x)\ \&\ \neg(B(x))) \subseteq y)$

Прием применяется к условию задачи на описание. Выражение y содержит неизвестные, выражения $A(x), B(x)$ - не содержат.

Спецификация приема имеет вид "тип(набороперандов)", "направл(N)", "неизвестные($x_1, \ldots, x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "условие", "тип(описать)", "корень", "не(известно(y))", "известно($A(x)$)", "известно($B(x)$)". Этого почти достаточно.

Число приемов данного типа - 4.

J. Переход к более простому описателю в условии задачи на описание и попытка явного разрешения этого условия.

Пример:

$\forall_{Aafghpqrs}(a = \lambda_{xy}(f(x,y), g(x,y))\ \&\ \mathrm{Max}(\lambda_y(f(h(y),y), g(h(y),y)), \mathrm{set}_v(A(v)), p, q) = (p = r\ \&\ q = s) \to \mathrm{Max}(a, \mathrm{set}_{zv}(z = h(v)\ \&\ A(v)), p, q) \leftrightarrow q = s\ \&\ p = \mathrm{set}_{zv}(z = h(v)\ \&\ v \in r))$

Прием применяется к подутверждению условия задачи на описание. Выражения p, q содержат неизвестные, выражения a, $h(v)$, $A(v)$ - не содержат. Прием усматривает, что двумерная область, по которой берется максимум - линия, и сводит задачу к одномерной.

Спецификация приема имеет вид "тип(транскоммент)", "направл(N)", "неизвестные($x_1, \ldots, x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "тип(описать)", "условие", "не(известно(p))", "не(известно(q))", "известно(a)", "известно($h(v)$)", "известно($A(v)$)". Этого достаточно.

Число приемов данного типа - 3.

K. Переход к более простому описателю в условии задачи на описание, явное разрешение этого условия и получение параметрического описания.

Пример:

$\forall_{abfguvw}(\mathrm{Extr}(\lambda_x(f(x), g(x)), u, c, w) = b \to \mathrm{Extr}(\lambda_x(f(x)+a, g(x)), u, v, w) \leftrightarrow \exists_c(b\ \&\ v = c + a))$

Прием применяется к условию задачи на описание. Выражения u, v, w содержат неизвестные, выражения a, $f(x)$, $g(x)$ - не содержат. При поиске экстремумов отбрасывается константное слагаемое a, которое затем учитывается посредством параметрического описания.

Спецификация приема имеет вид "тип(определено)", "направл(N)", "неизвестные($x_1, \ldots, x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "тип(описать)",

"условие", "не(известно(u))", "не(известно(v))", "не(известно(w))", "известно(a)", "известно($f(x)$)", "известно($g(x)$)". Вводятся также указатели "примечание(серия)", "примечание(фильтрсерии)".

Число приемов данного типа - 11.

iii. Разрешение посылки задачи на доказательство либо задачи на исследование, имеющей цель "известно", относительно заданных неизвестных.

Пример:

$\forall_{ax}(0 \le \pi + x \ \& \ 0 \le \pi - x \to \cos x = a \leftrightarrow (x = \arccos a \ \vee \ x = -\arccos a) \ \& \ 0 \le 1 + a \ \& \ 0 \le 1 - a)$

Прием применяется к посылке задачи на доказательство либо задачи на исследование, имеющей цель "известно". Выражение x содержит неизвестные, выражение a - не содержит. Наличие антецедентов обусловлено тем, что в задачах с целью "известно" аргументы тригонометрических функций обычно заключены в конечных пределах, и работа с полной серией корней нецелесообразна.

Спецификация приема имеет вид "тип(цели)", "направл(N)", "неизвестные($x_1 \dots x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "или(тип(доказать)и(тип(исследовать)цель(известно)))", "посылка", "корень", "не(известно(x))", "известно(a)". Иногда этого достаточно, но чаще нужна доработка.

Число приемов данного типа - 13.

(b) Разрешение группы утверждений относительно заданных неизвестных.

i. Разрешение группы условий задачи на описание относительно заданных неизвестных.

Пример:

$\forall_{abcdefpqrxy}(p = ae - bd \ \& \ q = ce - bf \ \& \ r = af - cd \ \& \ \neg(p = 0) \to ax + by = c \ \& \ dx + ey = f \leftrightarrow x = q/p \ \& \ y = r/p)$

Прием применяется к паре условий задачи на описание. Выражения x, y содержат неизвестные, а выражения a, b, c, d, e, f - не содержат.

Спецификация приема имеет вид "тип(область)", "направл(N)", "неизвестные($x_1, \dots, x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(описать)", "условие", "не(известно(x))", "не(известно(y))", "известно(a)", "известно(b)", "известно(c)", "известно(d)", "известно(e)", "известно(f)", "или(не(заголовок(a 1)) не(заголовок(b 1)) не(заголовок(d 1))не(заголовок(e 1)). Этого почти достаточно.

Число приемов данного типа - 15.

ii. Группа условий задачи на описание преобразуется в явное параметрическое описание неизвестных.

Пример:

$\forall_{abc}(\neg(\cos b = 0) \to a \le \operatorname{tg} b \ \& \ \operatorname{tg} b \le c \leftrightarrow \exists_n(n - \text{целое} \ \& \ \pi n + \operatorname{arctg} a \le b \ \& \ b \le \pi n + \operatorname{arctg} c))$

Прием применяется к паре условий задачи на описание. Выражение b содежит неизвестные, выражения a и c - не содержат.

Спецификация приема имеет вид "тип(о)", "направл(N)", "неизвестные($x_1 \dots x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(описать)", "условие", "не(известно(b))", "известно(a)", "известно(c)". Вводится указатель "примечание(серия)". Иногда этого достаточно, но во многих случаях требуется доработка.

Число приемов данного типа - 8.

iii. Разрешение группы посылок задачи на исследование относительно неизвестных.

Пример:

$\forall_{abcdefpxy}(p = bd - ae \ \& \ \neg(p = 0) \rightarrow ax + by = c \ \& \ dx + ey = f \leftrightarrow x = (bf - ce)/p \ \& \ y = (cd - af)/p)$

Прием применяется к паре посылок задачи на исследование, имеющей цель "известно". Выражения x, y содержат неизвестные, выражения a, b, c, d, e, f - не содержат.

Спецификация приема имеет вид "тип(предпоследсимвол)", "направл(N)", "неизвестные($x_1, \dots, x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(исследовать)", "цель(известно)", "посылка", "не(известно(x))", не(известно(y))", "известно(a)", "известно(b)", "известно(c)", "известно(d)", "известно(e)", "известно(f)", "или(не(заголовок(a 1)) не(заголовок(b 1)) не(заголовок(c 1)) не(заголовок(d 1)))". Этого достаточно.

Число приемов данного типа - 3.

iv. Использование пакета продукций для явного разрешения относительно неизвестных группы условий задачи на описание.

Пример:

$\forall_{abnxy}(\lambda_{ij}(a(i,j), i \in \{1, \dots, n\} \ \& \ j \in \{1, \dots, n\})y = \lambda_i(b(i), i \in \{1, \dots, n\}) \rightarrow \forall_i(i \in \{1, \dots, n\} \rightarrow \sum_{j=1}^{n}(a(i,j)x(j)) = b(i)) \leftrightarrow \forall_j(j \in \{1, \dots, n\} \rightarrow x(j) = y(j)))$

Прием применяется к группе условий задачи на описание, представляющих собой линейные уравнения с численными коэффициентами. Указатель "контрольвывода" инициирует попытку применения приема при усмотрении уравнения вида $cd + e = f$, где d - неизвестная, c - известно. Проверяется, что каждое слагаемое остаточной суммы e тоже линейно относительно неизвестной. Список переменных x идентифицируется с перечнем неизвестных, входящих в уравнения задачи. Переменной n присваивается длина списка x. Указатель "коэфф(a)" определяет идентификацию переменной a с прямоугольной матрицей; проверяется, что ее элементы константные. Антецедент выделен указателем "программа". Он обращается к пакету продукций "линсист", решающему систему линейных уравнений с численными коэффициентами стандартным методом. Предварительно элементы матрицы a представляются в машинном формате "с плавающей запятой".

Спецификация приема имеет вид "тип(внешнеизв)", "направл(N)", "неизвестные($x_1 \dots x_n$)". Добавляется элемент "указатель(...)", выделяющий антецедент для обработки пакетом продукций. Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1 и вводом фильтров "условие", "тип(описать)", "неизвестная(d)". Он не прорабатывался.

Число приемов данного типа - 3.

(c) Выражение одних неизвестных подтермов задачи на описание либо на исследование через другие.

i. Выражение неизвестной задачи на описание либо на исследование через другие неизвестные.

A. Выражение неизвестной задачи на описание либо на исследование через другие неизвестные.

Приемы данного типа, несмотря на свою привлекательность, бывают полезны лишь в редких случаях. Например, при решении систем линейных уравнений. В прочих ситуациях подстановка громоздкого выражения одной неизвестной через другие способна загнать в тупик решение оставшихся уравнений. Поэтому приемов данного типа создано совсем немного, и они имеют множество ограничений. Пример:

$\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \rightarrow a = b + c \leftrightarrow b = a - c)$

Прием применяется к условию задачи на описание либо посылке задачи на исследование, имеющих более одной неизвестной. Переменная b - неизвестная. Выражение c содержит неизвестные, но не содержит b. Выражение a неизвестных не содержит. Уровни срабатывания - 0,1 и 5. Дополнительно проверяются ограничения:

Либо решается задача на описание и текущий уровень равен 5, либо a равно 0, а число неизвестных преобарзуемого условия равно 2, либо это условие - линейное уравнение.

В случае задачи на описание текущий уровень не равен 0, иначе - не равен 1.

Либо преобразуемое уравнение имеет ровно две неизвестных, либо решается задача на описание, имеющая не более одного нелинейного уравнения, либо решается задача на исследование, a равно 0, а преобразуемое уравнение имеет ровно три неизвестных.

Если решается задача на описание, причем b имеет целочисленные значения, то c содержит не более двух неизвестных.

Спецификация приема имеет вид "тип(новоперанд)", "направл(N)", "неизвестные($x_1 \dots x_n$)", "переменная(y)". Здесь y - неизвестная, относительно которой предпринимается разрешение; $x_1, \dots, x_n$ - все переменные, идентифицируемые с содержащими неизвестные выражениями. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "или(и(тип(описать)условие) тип(исследовать))", "корень", "не(цель(редакция))". "неизвестные(

2)", "неизвестная(x2)", "не(известно(x3))", "известно(x1)", "не(входит(x2 x3))", "не(контекст(ключ(цели независит x4)список(x5 x1 x3) пересекаются(x4 x5)))". Он требует существенной доработки.

Число приемов данного типа - 3.

B. Выражение неизвестной задачи на описание через другие неизвестные.

Замечание то же, что для предыдущего типа. Пример:

$\forall_{bcx}(x + b = c \leftrightarrow x = c - b)$

Здесь имеется в виду сложение векторов. Прием применяется к подутверждению условия задачи на описание. Переменная x - неизвестная, не встречающаяся в выражениях b, c. Выражение c не является неизвестной.

Спецификация приема имеет вид "тип(стандплюс)", "направл(N)", "неизвестная(x)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "условие", "тип(описать)", "корень", "неизвестная(x)", "не(входит(x ;b))", "не(входит(x c))". Он требует доработки.

Число приемов данного типа - 4.

C. Выражение неизвестной задачи на исследование, имеющей цель "известно", через другие неизвестные.

Пример:

$\forall_{abcx}(\neg(b = 0) \rightarrow a + bx = c \leftrightarrow x = (c - a)/b)$

Прием применяется к посылке задачи на исследование, имеющей цель "известно". Переменная x идентифицируется с неизвестной, переменная c - не с неизвестной. Имеется множество дополнительных ограничений, причем уровень срабатывания достаточно высок.

Спецификация приема имеет вид "тип(прогрвыражение)", "направл(N)", "неизвестные($x_1, \ldots, x_n$)", "переменная(x)". Здесь x - неизвестная, относительно которой предпринимается разрешение; x_1, $\ldots, x_n$ - отличные от нее переменные, которым разрешено идентифицироваться с содержащими неизвестные выражениями. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(исследовать)", "цель(известно)", "корень", "неизвестная(x)", "не(входит(x a))", "не(входит(x b))", "не(входит(x c))", "конец(не(контекст(список(x4 a b c)числовойатом(x4 x5) не(переменная(x5)))))". Он требует доработки.

Число приемов данного типа - 5.

(d) Преобразование условия задачи на описание либо посылки задачи на исследование к виду, допускающему явное разрешение относительно заданного выражения с неизвестными.

i. Преобразование условия задачи на описание либо посылки задачи на исследование к виду, допускающему явное разрешение относительно заданного выражения с неизвестными.

$\forall_{abcdefghijk}(0 < a \ \& \ b - 2c = f \ \& \ d = ae \ \& \ g - 2c = h \rightarrow ie^b + jd^c + ka^g = 0 \leftrightarrow ie^f((e/a)^c)^2 + j(e/a)^c = -ka^h)$

Прием применяется к условию задачи на описание либо посылке задачи на исследование. Выражения b, c, g содержат неизвестные, а выражения f, h, i, j, k - не содержат. Если h отлично от нуля, то a не содержит неизвестных. Если f отлично от нуля, то e не содержит неизвестных. Заменяющее утверждение обарабатывается нормализатором решения квадратных уравнений "квадруравн".

Спецификация приема имеет вид "тип(нормуравн)", "направл(N)", "неизвтермы(t)", "неизвестные($x_1 \ldots x_n$)", норм(s)". Здесь t - выражение, относительно которого должно разрешаться уравнение, $x_1, \ldots, x_n$ - переменные, идентифицируемые с содержащими неизвестные выражениями, s - нормализатор разрешения уравнений. В нашем случае t - "$(e/a)^c$"; $x_1, \ldots, x_n$ - b, c, g. Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "или(и(тип(описать)условие) тип(исследовать))", "корень", "не(известно(b))", "не(известно(c))", "не(известно(g))", "известно(f)", "известно(h)", "известно(i)", "известно(j)", "известно(k)", "или(известно(a)заголовок(h 0))", "или(известно(e) заголовок(f 0))". Вводится также нормализатор "квадруравн". Этого достаточно.

Число приемов данного типа - 9.

ii. Преобразование условия задачи на описание либо посылки задачи на исследование к виду, допускающему явное разрешение относительно неизвестной.

Пример:

$\forall_{abx}(a(\sin x)^2 = b\cos(2x) \leftrightarrow (a + 2b)\cos(2x) = a)$

Прием применяется к условию задачи на описание либо посылке задачи на исследование. Выражение x содержит неизвестные, выражения a, b - не содержат. Так как заменяющее утверждение устойчиво к общей стандартизации, немедленной обработки нормализатором разрешения уравнений не требуется.

Спецификация приема имеет вид "тип(блокприемов)", "направл(N)", "неизвестные(x)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "или(и(тип(описать) условие) тип(исследовать))", "корень", "не(известно(x))", "известно(a)", "известно(b)". Этого почти достаточно.

Число приемов данного типа - 3.

iii. Преобразование условия задачи на описание к виду, допускающему явное разрешение относительно заданного выражения с неизвестными.

Пример:

$\forall_{ABCDEabpq}(pa^2 = A \ \& \ 2apb = B \ \& \ aq = C - pb^2 \ \& \ qb = D \rightarrow Ax^4 + Bx^3 + Cx^2 + Dx = E \leftrightarrow p(ax^2 + bx)^2 + q(ax^2 + bx) = E)$

Прием применяется к условию задачи на описание. Выражение x содержит неизвестные; выражения A, B, C, D, E идентифицируются с

целочисленными константами. Антецеденты выделены указателем "программа". Они реализуют перебор целочисленных делителей для выделения квадратного трехчлена относительно двучлена второй степени.

Спецификация приема имеет вид "тип(вычерк)", "направл(N)", "неизвтермы(t)", "неизвестные($x_1 \ldots, x_n$)", "норм(s)". Все это аналогично рассмотренному выше типу, где допускались как задачи на описание, так и задачи на исследование. Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие", "корень", "не(известно(x))", "известно(p)", "известно(q)", "известно(E)". Учет этим справочником условий целочисленности не прорабатывался. Однако, при отсутствии таких условий он создает приемлемые приемы.

Число приемов данного типа - 7.

iv. Преобразование условия задачи на описание к виду, допускающему явное разрешение относительно неизвестной.

Пример:

$\forall_{abc}(0 < d \to a(\sin c)^d + b(\sin c)^d = 0 \leftrightarrow \neg(a = 0)\ \&\ (\operatorname{tg} c)^d = -b/a\ \vee\ a = 0\ \&\ b\cos c = 0)$

Прием применяется к условию задачи на описание. Выражение c содержит неизвестные, а выражения a, b, d не содержат. Выражение d отлично от двойки.

Спецификация приема имеет вид "тип(высотаполосы)", "направл(N)", "неизвестные($x_1 \ldots x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(описать)", "условие", "корень", "не(известно(c))", "известно(a)", "известно(b)", "известно(d)". Он требует доработки.

Число приемов данного типа - 27.

v. Свертка дизъюнктивного условия задачи на описание, приводящая к разрешимому относительно неизвестного подвыражения подслучаю.

Пример:

$\forall_{ab}(\sin a = 0\ \vee\ \cos a = 0\ \vee\ b \leftrightarrow \sin(2a) = 0\ \vee\ b)$

Прием применяется к условию задачи на описание. Выражение a содержит неизвестные.

Спецификация приема имеет вид "тип(дизъюнкциявсех)", "направл(N)", "неизвестные($x_1 \ldots x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "условие", "корень", "тип(описать)", "не(известно(a))". Этого достаточно.

Число приемов данного типа - 3.

(e) Преобразование условия задачи на описание, разрешенного относительно неизвестной.

i. Преобразование разрешенного относительно неизвестной условия задачи на описание в дизъюнкцию, подслучай которой дает явное выражение для этой неизвестной.

Пример:

$\forall_{abc}(a \in \{b;c\} \leftrightarrow a = b \ \vee \ a \in \{;c\})$

Прием применяется к условию задачи на описание. Выражение a содержит неизвестные, выражения b, c - не содержат. Либо a - не переменная, либо существует другое условие с переменной a, отличное от указателя типа значения a.

Спецификация приема имеет вид "тип(копия)", "направл(N)", "неизвестная(x)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "условие", "корень", "тип(описать)", "не(известно(a))", "известно(фикс(0 1 2))", "или(не(переменная(a)) контекст(новоеусловие(x4) входит(x1 x4) равно(x6 справка(родобъекта начало(x4))) заголовок(x6 0)))". Создается также указатель "примечание(разборслучаев)". Этого почти достаточно.

Число приемов данного типа - 3.

ii. Разгруппировка условия принадлежности неизвестного элемента известному множеству для последующей расшифровки.

Пример:

$\forall_{abc}(a \in b \cup c \leftrightarrow a \in b \ \vee \ a \in c)$

Прием применяется к подутверждению условия задачи на описание. Выражение a содержит неизвестные, выражения b и c не содержат. Либо b, либо c содержит логический символ, отличный от символов "пересечение", "объединение", "разность", "номера", "набор", "перечень", "область" и не расположенный внутри операций "перечень" либо "номера". Если a переменная, то она должна встречаться еще в каком-то условии, отличном от принадлежности либо непринадлежности этой переменной известному множеству и не являющемся указателем типа значения a.

Спецификация приема имеет вид "тип(заменанеизвестной)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "условие", "тип(описать)", "не(известно(a))", "известно(b)", "известно(c)", "или(контекст(подчинено(x4 b)логсимвол(x4)не(символ(x4 набор объединение пересечение разность перечень номера область))не(контекст(подчинено(x4 x5)символ(x5 перечень номера)))) контекст(подчинено(x4 c)логсимвол(x4)не(символ(x4 набор объединение пересечение разность перечень номера область))не(контекст(подчинено(x4 x5)символ(x5 перечень номера)))))", "или(не(переменная(a)) контекст(новоеусловие(x4) входит(a x4) не(контекст(вид(x4 принадлежит(a x5)) известно(x5))) не(контекст(вид(x4 не(принадлежит(a x5))) известно(x5))) или(не(длинатекста(x4 2)) не(равно(справка(родобъекта начало(x4))1)))))". Этого достаточно.

Число приемов данного типа - 3.

iii. Разрешенное относительно неизвестной условие задачи на описание преобразуется в явное параметрическое описание для исключения данной неизвестной из других условий.

Пример:

$\forall_{ABa}(a \in A \times B \leftrightarrow \exists_{xy}(x \in A \ \& \ y \in B \ \& \ a = (x, y)))$

Прием применяется к условию задачи на описание, не имеющей цели (класс ...). Либо a - неизвестная, либо задача имеет цель вида "(серия $\ldots a \ldots$)". Существует еще условие, содержащее a.

Спецификация приема имеет вид "тип(лексикопредшествует)", "направл(N)", "неизвестная(a)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "корень", "или(неизвестная(a) контекст(цель(x2) заголовок(x2 серия) входит(a x2)))", "контекст(новоеусловие(x2) входит(a x2))". Создаются также указатели "примечание(серия)" и "примечание(фильтрсерии)". Этого почти достаточно.

Число приемов данного типа - 6.

iv. Усиление условия задачи на описание, явно разрешенного относительно неизвестной.

Пример:

$\forall_{nx}(n - \text{целое} \ \& \ x - \text{целое} \rightarrow n < x \leftrightarrow n + 1 \leq x)$

Прием применяется к условию задачи на описание. Переменная x - неизвестная, выражение n неизвестных не содержит.

Спецификация приема имеет вид "тип(делители)", "направл(N)", "неизвестная(x)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(описать)", "условие", "корень", "неизвестная(x)", "известно(n)". Этого почти достаточно.

Число приемов данного типа - 6.

v. Конъюнктивная декомпозиция явно разрешенного относительно неизвестной условия для учета прочих ограничений на эту неизвестную.

Пример:

$\forall_{abx}(x \in [a, b] \leftrightarrow x - \text{число} \ \& \ a \leq x \ \& \ x \leq b)$

Прием применяется к условию задачи на описание. Выражение x содержит неизвестные. Либо x - не переменная, либо имеется еще какое-то другое условие с x. Достаточно высокий уровень срабатывания приема гарантирует, что попытки объединения явно разрешенных относительно x условий в одно условий уже не состоялись.

Спецификация приема имеет вид "тип(конечное)", "направл(N)", "неизвестная(x)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "условие", "тип(описать)", "не(известно(x))", "или(не(переменная(x))контекст(новоеусловие(x3) входит(x x3) не(контекст(вид(x3 принадлежит(x x4)) известно(x4))) не(контекст(вид(x3 не(принадлежит(x x4)))известно(x4))) не(контекст(вид(x3 не(равно(x x4))) известно(x4))) или(не(заголовок(x3 число)) не(длинатекста(x3 2)))))". Этого достаточно.

Число приемов данного типа - 3.

(f) Усмотрение возможности выразить условие задачи на описание через заданное неизвестное подвыражение и разрешение его относительно этого подвыражения.

Пример:

$\forall_{afghxy}((y - \text{число } \& \ a \leq f(y)) = g(y) \rightarrow a \leq f(\sin(h(x)) + \cos(h(x))) \leftrightarrow g(\sin(h(x)) + \cos(h(x))))$

Прием применяется к условию задачи на описание, имеющему единственную неизвестную. Выражение a неизвестных не содержит. Указатель "контекст" определяет идентификацию неизвестной x и содержащего x подвыражения текущего условия, имеющего вид $\sin(h(x)) + \cos(h(x))$. Для идентификации $f(\sin(h(x)) + \cos(h(x))$ используется указатель "новаргумент(f x извлечение)".

Спецификация приема имеет вид "тип(Облвхожд)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие", "корень", "равно(числонеизвестных(корень)1)". Он не прорабатывался.

Число приемов данного типа - 4.

2. Дизъюнктивно-конъюнктивные декомпозиции.

   (a) Дизъюнктивно-конъюнктивная декомпозиция условия задачи на описание либо посылки задачи на исследование относительно неизвестных.

      i. Попытка декомпозиции условия задачи на описание либо посылки задачи на исследование с помощью нормализатора приведения к заданным заголовкам.

         A. Попытка декомпозиции условия задачи на описание либо посылки задачи на исследование с помощью нормализатора приведения к заданным заголовкам.

            Пример:

            $\forall_{abc}(c = a - b \ \& \ a - \text{число } \& \ b - \text{число} \rightarrow a = b \leftrightarrow c = 0)$

            Прием применяется к условию задачи на описание, не имеющей цели "редакция", либо к посылке задачи на исследование. Выражение a отлично от переменной и содержит неизвестные. Либо b отлично от 0, либо a не имеет своим заголовком один из символов "умножение", "степень", "дробь". Отсутствует цель "известно ...". Правая часть первого антецедента обрабатывается нормализатором разложения на множители "видумножение". Прием является одним из наиболее часто применяемых в элементарной алгебре и имеет множество дополнительных фильтров. Созданы две его версии, срабатывающие на различных уровнях.

            Спецификация приема имеет вид "тип(декомпозиция)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "или(и(условие тип(описать) не(цель(редакция))) тип(исследовать))", "корень", "не(известно(корень))", "не(Входит(известно цели))", "не(подобныетермы(c копия($a-b$)))". Вводится нормализатор "видумножение" для обработки первого антецедента. Требуется существенная доработка справочника.

            Число приемов данного типа - 4.

ii. Дизъюнктивно-конъюнктивная декомпозиция условия задачи на описание либо посылки задачи на исследование относительно неизвестных.

A. Дизъюнктивно-конъюнктивная декомпозиция условия задачи на описание относительно неизвестных.

Пример:

$\forall_{ab}(0 \le ab \leftrightarrow a \le 0 \ \&\ b \le 0 \ \vee \ 0 \le a \ \&\ 0 \le b)$

Прием применяется к условию задачи на описание, содержащему неизвестные. Оно не используется для сопровождения по о.д.з.

Спецификация приема имеет вид "тип(нормнок)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие", "корень", "не(известно(корень))", "не(Входит(или списокусловий))". Требуется небольшая доработка.

Число приемов данного типа - 17.

B. Дизъюнктивно-конъюнктивная декомпозиция условия задачи на описание относительно заданных неизвестных, использующая конечное перечисление относительно константных параметров.

Пример:

$\forall_{nxy}(x - \text{целое} \ \&\ y - \text{целое} \rightarrow xy = n \leftrightarrow \exists_{pq}(n = pq \ \&\ x = p \ \&\ y = q))$

Прием применяется к условию задачи на описание. Переменная n идентифицируется с ненулевой целочисленной константой. Выражения x, y содержат неизвестные. Задача имеет целочисленные неизвестные. Есть ограничение сверху на число делителей n. Квантор существования выделен указателем "или", определяющим его развертку в дизъюнкцию при перечислении разложений n в произведение двух целочисленных множителей p, q.

Спецификация приема имеет вид "тип(ключ)", "неизвестные($x_1 \ldots x_m$)", "направл(N)", "см(...)". Последний элемент уточняет тип константных значений. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "тип(описать)", "условие", "корень", "не(известно(x))", "не(известно(y))", "целое(n)". Создаются также указатели "или(...)" и "примечание(разборслучаев)". Требуется доработка.

Число приемов данного типа - 3.

iii. Дизъюнктивно-конъюнктивная декомпозиция условия задачи на описание либо посылки задачи на исследование путем анализа экстремальных значений.

A. Дизъюнктивно-конъюнктивная декомпозиция условия задачи на описание либо посылки задачи на исследование путем анализа экстремальных значений.

Пример:

$\forall_{abc}(b - \text{натуральное} \rightarrow (\cos a)^b \cdot c = 1 \leftrightarrow (\cos a)^b = 1 \ \&\ c = 1 \ \vee (\cos a)^b = -1 \ \&\ c = -1)$

Прием применяется к условию задачи на описание либо посылке задачи на исследование, содержащим неизвестные. Каждый сомножитель выражения c представляет собой степень синуса либо косинуса с положительным показателем. Отбрасываются случаи, в которых возможно решение без пересечения нескольких различных серий корней.

Спецификация приема имеет вид "тип(нормвнутренность)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "или(и(тип(описать) условие) тип(исследовать))", "корень", "не(известно(фикс(0 1)))". Он не проработан.

Число приемов данного типа - 7.

B. Дизъюнктивно-конъюнктивная декомпозиция условия задачи на описание путем анализа экстремальных значений.

Пример:

$\forall_{abcde}(|a| \le b \ \& \ |c| \le d \ \& \ bd - e = 0 \to ac = e \leftrightarrow |a| = b \ \& \ |c| = d \ \& \ 0 \le ac)$

Прием применяется к условию задачи на описание, содержащему неизвестные. Выражение e неизвестных не содержит. Первые два антецедента обрабатываются синтезатором "верхняяоценка", определяющим верхние оценки b, d. Третий антецедент, выделенный указателем "идентификатор", сравнивает произведение этих оценок с выражением e.

Спецификация приема имеет вид "тип(нормрасстояние)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие", "корень", "не(известно(корень))". Он правильно расставляет указатели обработки антецедентов. Требуется доработка.
Число приемов данного типа - 17.

iv. Декомпозиция условия задачи на описание с разрешением получаемых подутверждений.

Пример:

$\forall_{MNRSabcdfgmnpqrs}(\mathrm{Max}(\lambda_x(f(x), x - \text{число}), a, m, n) = (m = M \ \& \ n = N) \ \& \ \mathrm{Max}(\lambda_y(g(y), y - \text{число}), b, r, s) = (r = R \ \& \ s = S) \to \mathrm{Max}(\lambda_{xy}(f(x) + g(y), p(x) \ \& \ q(y)), a \times b, c, d) \leftrightarrow c = M \times R \ \& \ d = N + S)$

Прием применяется к условию задачи на описание. Антецеденты выделены указателем "идентификатор". Их левые части разрешаются относительно соответствующих неизвестных (m, n либо r, s) задачами на описание.

Спецификация приема имеет вид "тип(отбор)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие". Он требует доработки.
Число приемов данного типа - 6.

(b) Эквивалентное преобразование условия задачи на описание, обеспечивающее последующую его декомпозицию.

i. Эквивалентное преобразование условия задачи на описание, обеспечивающее последующую его декомпозицию.

Пример:

$\forall_{abcdxy}(x - \text{целое } \& \ y - \text{целое } \& \ \neg(a = 0) \rightarrow axy + bx + cy = d \leftrightarrow (ax + c)(ay + b) = ad + bc)$

Прием применяется к условию задачи на описание. Выражения x, y содержат неизвестные. Переменные a, b, c, d идентифицируются с целочисленными константами. Дальнейшая декомпозиция использует перечисление всевозможных способов разложения на два целочисленных множителя константы в правой части.

Спецификация приема имеет вид "тип(свобвхождение)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие", "корень", "не(цель(редакция))". Он не проработан.

Число приемов данного типа - 4.

ii. Использование нормализатора преобразования к заданному заголовку для последующей декомпозиции уравнения.

Пример:

$\forall_{acd}(a = d \rightarrow d \subseteq c \leftrightarrow a \subseteq c)$

Прием применяется к подутверждению условия задачи на описание. Выражение d отлично от переменной, содержит неизвестные и не имеет заголовка "объединение". Правая часть антецедента обрабатывается нормализатором "видобъединение". Результат a либо имеет заголовок "объединение". либо короче исходного выражения d. Хотя выше уже рассматривался почти такой же тип приема, но отличие между ними имеется: здесь источником приема служит протокол "нормразделение(...)", а ранее источником была теорема либо близкий к ней квазипротокол.

Спецификация приема имеет вид "тип(соотвпозиция)", "направл(N)", "см(...)", где последний элемент фактически содержит все необходимые фильтры. Он генерируется спецификатором по протоколу "нормразделение(...)" автоматически; в нашем случае этот элемент имеет вид "см(не(известно(d))не(переменная(d)) не(заголовок(d объединение)) или(короче(a d) заголовок(a объединение)) длина(контрольглубины))". Справочник "заголовокприема" указывает уровень срабатывания 2, вводит фильтры "тип(описать)", "условие", "не(цель(редакция))" и добавляет фильтры из элемента "см(...)". Этого достаточно.

Число приемов данного типа - 10.

3. Свертка нескольких условий задачи на описание в одно условие.

(a) Свертка группы явно разрешенных относительно неизвестных условий в одно, тоже явно разрешенное относительно неизвестных.

Пример:

$\forall_{abc}(a \cup b \subseteq c \leftrightarrow a \subseteq c \ \& \ b \subseteq c)$

Прием применяется к паре условий задачи на описание (замена выполняется справа налево). Переменная c - неизвестная, выражения a и b неизвестных не содержат.

Спецификация приема имеет вид "тип(кн)", "направл(N)", "неизвестные(c)". Справочник "заголовокприема" указывает уровни срабатывания 0,3,5 и вводит фильтры "условие", "тип(описать)", "неизвестная(c)", "известно(a)", "известно(b)", "или(уровень(5) не(цель(пример)))". Этого почти достаточно.

Число приемов данного типа - 30.

(b) Свертка нескольких неизвестных условий в одно условие, допускающее непосредственное разрешение относительно неизвестного подвыражения. Пример:

$\forall_a(\sin(2a) = 0 \ \& \ \neg(\sin a = 0) \leftrightarrow \cos a = 0)$

Прием применяется к паре условий задачи на описание. Выражение a содержит неизвестные.

Спецификация приема имеет вид "тип(однасторона)", "направл(N)", "неизвестные(a)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "не(известно(a))". Этого почти достаточно.

Число приемов данного типа - 4.

(c) Свертка в одно условие нескольких условий с неизвестными, используемых только при проверке.

Пример:

$\forall_a(\neg(\sin a = 0) \ \& \ \neg(\cos a = 0) \leftrightarrow \neg(\sin(2a) = 0))$

Прием применяется к паре условий задачи на описание, не используемых для сопровождения по о.д.з. Выражение a содержит неизвестные. Имеется условие, представляющее собой равенство тригонометрической функции от содержащего неизвестные выражения известному выражению. Отсутствуют равенства, выражающие $\sin a$ либо $\cos a$ через терм, не содержащий неизвестных.

Спецификация приема имеет вид "тип(начало)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "не(известно(a))", "не(сопровождение)". Он не прорабатывался.

Число приемов данного типа - 4.

(d) Свертка группы неизвестных условий в одно условие, явно разрешенное относительно неизвестного выражения.

Пример:

$\forall_{ab}((a = b \ \vee \ a = -b) \ \& \ 0 \leq a \leftrightarrow a = |b|)$

Прием применяется к паре условий задачи на описание. Выражение a содержит неизвестные, выражение b - не содержит. Заметим, что в отличие от ранее рассмотренного в этом подразделе типа, переменная a не обязательно идентифицируется с неизвестной.

Спецификация приема имеет вид "тип(усмконечное)", "направл(N)", "неизвестные($x_1 \dots x_n$)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "условие", "тип(описать)", "не(известно(a))", "известно(b)". В ряде случаев требуется доработка.

Число приемов данного типа - 12.

4. Упрощение утверждений относительно неизвестных.

   (a) Неизвестные подвыражения со сложным заголовком.

      i. Преобразование условия либо посылки, исключающее сложное выражение с неизвестными.

         A. Преобразование условия задачи на описание либо посылки задачи на исследование, исключающее сложное выражение с неизвестными.

            Пример:

            $\forall_{abcd}(0 < b \to ab^d = c \leftrightarrow \neg(b-1 = 0)\ \&\ (0 < c\ \&\ 0 < a\ \&\ d+\log_b(a) = \log_b(c)\ \vee\ c < 0\ \&\ a < 0\ \&\ d + \log_b(-a) = \log_b(-c))\ \vee\ a = 0\ \&\ c = 0\ \vee\ b - 1 = 0\ \&\ a = c)$

            Прием применяется к условию задачи на описание. Выражение d содержит неизвестные, выражение b - не содержит. Каждый содержащий неизвестные сомножитель выражений a, c представляет собой степень с известным основанием. Выражение b - либо десятичная константа, либо самое короткое основание неизвестной степени среди сомножителей выражений a, c. Прием логарифмирует показательное уравнение, причем после упрощений логарифмы с неизвестными не возникают.

            Спецификация приема имеет вид "тип(прообраз)", "направл(N)", "см(...)". Последний элемент содержит необходимые фильтры. Он создается автоматически спецификатором по характеристике "неизвоценка(...)", которая, в свою очередь, создается автоматически процедурой программирующего вывода, получившей теорему. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры, извлекаемые из элемента "см "тип(описать)", "условие", "корень", "не(известно(d))", "или(и(известно(b)не(контекст(сомножитель(x6 a)не(известно(x6)) не(контекст(вид(x6 степень(x7 x8)) известно(x7))))) не(контекст(сомножитель(x9 c)не(известно(x9)) не(контекст(вид(x9 степень(x5 x6)) известно(x5)))))) и(не(известно(b))заголовок(a 1) не(контекст(сомножитель(x9 c) не(заголовок(x9 1)) не(контекст(вид(x9 степень(b x5)) единица(1 x5)))))))". Хотя этого и не совсем достаточно, но автоматически сгенерированная версия охватывает более широкий класс случаев -

учитывается возможность вхождения неизвестных в основание степени b.

Приведенный пример хорошо иллюстрирует тот факт, что иногда фильтры приема зарождаются еще в процессе вывода его теоремы. Если это учитывать, то станет возможным автоматически генерировать многие необъяснимые иными средствами фильтры приемов.

Число приемов данного типа - 11.

B. Преобразование подутверждения условия задачи на описание либо посылки задачи на исследование, исключающее сложное выражение с неизвестными.

Отличие от предыдущего случая лишь в том, что разрешается некорневое применения преобразования. Пример:

$\forall_{abc}(b-\text{число} \,\&\, \neg(b=0) \to a/b < c \leftrightarrow 0 < b \,\&\, a, bc \vee b < 0 \,\&\, bc < a)$

Прием применяется к подутверждению условия задачи на описания либо посылки задачи на исследование. Хотя бы одно из выражений a, b содержит неизвестные, выражение c - не содержит. В качестве надтермов преобразуемого терма допускаются только конъюнкции либо дизъюнкции.

Спецификация приема имеет вид "тип(Узелприема)", "направл(N)", "терм(...)", ..., "терм(...)", где последние элементы перечисляют исключаемые сложные выражения. В нашем случае - "a/b". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "или(и(тип(описать)условие) тип(исследовать))", "не(известно(фикс(0 1 1)))". Он требует доработки.

Число приемов данного типа - 5.

C. Преобразование условия задачи на описание, исключающее сложное выражение с неизвестными.

Пример:

$\forall_{abcdp}(0 \le d \to ad^{p/2} + b = c \leftrightarrow a^2 d^p - b^2 + 2bc = c^2 \;\&\; 0 \le (c-b)a)$

Прием применяется к условию задачи на описание для исключения содержащего неизвестные радикала. Переменная p идентифицируется с натуральной константой. Выражение d содержит неизвестные, c - не содержит. Выражение p не имеет слагаемых, сомножителем которых служит неизвестный радикал. Имеется множество дополнительных фильтров.

Спецификация приема имеет вид "тип(дизъюнктоперанд)", "направл(N)", "терм(...)", ..., "терм(...)", где последние элементы перечисляют исключаемые сложные выражения. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(описать)", "условие", "корень", "не(известно(фикс(0 1 1 1 2)))", "известно(c)". Он не прорабатывался.

Число приемов данного типа - 18.

D. Преобразование посылки задачи на доказательство либо на исследование, исключающее сложное выражение с неизвестными.

Пример:

$\forall_{abc}(0 \le a \ \& \ 0 \le b \ \& \ 0 \le c \ \& \ 0 \le d \to a\sqrt{b} = c\sqrt{d} \leftrightarrow a^2b - c^2d = 0)$

Прием применяется к посылке задачи на доказательство либо задачи на исследование, имеющей цель "известно". Выражения b, d содержат неизвестные, выражения a и c константные.

Спецификация приема имеет вид "тип(натуральные)", "направл(N)", "терм(...)", ..., "терм(...)", где последние элементы перечисляют исключаемые сложные выражения. Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "посылка", "или(тип(доказать)и(тип(исследовать)цель(известно)))", "корень", "не(известно(b))", "не(известно(d))". Он не проработан.

Число приемов данного типа - 12.

ii. Преобразование условия либо посылки к виду, позволяющему исключить сложное выражение с неизвестными.

A. Преобразование условия задачи на описание к виду, позволяющему исключить сложное выражение с неизвестными.

Пример:

$\forall_{abcdefq}(q = a^2e - c^2 + 2cd - b^2f + 2b(d-c)\sqrt{f} \to a\sqrt{e} + b\sqrt{f} = d \leftrightarrow q = d^2 \ \& \ 0 \le (d - c - b\sqrt{f})a)$

Прием применяется к условию задачи на описание. Выражения e, f содержат неизвестные, выражение d - не содержит. Выражение c не имеет своими слагаемыми одночленов с квадратными радикалами. После перехода от двух радикалов к одному появляется перспектива полного исключения радикалов.

Спецификация приема имеет вид "тип(исключнеизв)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1 и вводом фильтров "тип(описать)", "корень", "условие", "известно(d)". Он не прорабатывался.

Число приемов данного типа - 15.

B. Применение нормализатора приведения к заданным заголовкам к подвыражению условия задачи на описание для последующего эквивалентного преобразования, исключающего сложную операцию.

Пример:

$\forall_{abcde}(e = a/b + c \to a/b + c = d \leftrightarrow e = d)$

Прием применяется к условию задачи на описание. Дробь и выражение c содержат неизвестные. Выражение d неизвестных не содержит. Правая часть антецедента обрабатывается нормализатором "видумножение", обеспечивающим сложение дробных выражений с неизвестными. Для последующего исключения из уравнения таких выражений будет применяться прием, домножающий

обе части на знаменатель. Прием имеет множество дополнительных фильтров. Обращение к нормализатору, в зависимости от контекста, сопровождается множеством комментариев.

Спецификация приема имеет вид "тип(корректформ)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1 и вводом фильтров "тип(описать)", "корень", "условие". Он не прорабатывался.

Число приемов данного типа - 3.

iii. Переход в условии задачи на описание к альтернативным сложным выражениям с неизвестными.

Пример:

$\forall_{abcde}(0 < a \ \& \ 0 < b \ \& \ \neg(b-1=0) \to ca^d + e = 0 \leftrightarrow e < 0 \ \& \ 0 < c \ \& \ d\log_b(a) - \log_b(-e) = -\log_b(c) \ \vee \ 0 < e \ \& \ c < 0 \ \& \ d\log_b(a) - \log_b(e) = -\log_b(-c) \ \vee \ c = 0 \ \& \ e = 0)$

Прием применяется к условию задачи на описание. Некоторое ее уравнение уже содержит выражение вида $\log_b(A)$,где A содержит неизвестные, b - не содержит. Выражения a, c, e не содержат неизвестных логарифмов. Выражение e - не сумма. Прием не изменяет текущей задачи, а лишь предпринимает попытку решить ее копию, в которой реализована замена. Вводятся комментарии, блокирующие обратный переход от логарифмов к показательным выражениям.

Спецификация приема имеет вид "тип(упрощминусвект)", "направл(N)", "терм(t)", где последний элемент указывает дополнительно идентифицируемый терм t. В нашем примере это $\log_b(A)$. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(описать)", "условие", "корень", "не(известно(фикс(0 1 1 1 2)))", "не(известно(x8))", "известно(x2)". Создаются указатели "попытказамены", "контекст(условие(x7)позиция(x8 x7)вид(x8 логарифм(b A)))". Требуется доработка.

Число приемов данного типа - 15.

iv. Применение нормализатора, стандартизирующего неизвестные подвыражения условия задачи на описание со сложным заголовком.

Пример:

$\forall_{abcde}(c = a \ \& \ e = d/2 \to a < b \leftrightarrow \neg(\cos e = 0) \ \& \ c < b \ \vee \ \cos e = 0 \ \& \ a < b)$

Прием применяется к условию задачи на описание, содержащему подтерм "cos(d)". Правая часть первого антецедента обрабатывается нормализатором "половинныйугол", выражающим тригонометрические функции от d через тангенс половинного угла. Предварительно проверяется ряд требований. В частности, наличие в текущем условии синуса d либо тангенса e и отстутствие в нем неизвестных тригонометрических операций не от d либо не являющихся тангенсом e.

Спецификация приема имеет вид "тип(равнозначны)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня сра-

батывания 6 и вводом фильтров "тип(описать)", "условие", "корень". Он не прорабатывался.

Число приемов данного типа - 7.

(b) Расшифровка неизвестного подутверждения условия задачи на описание.

Пример:

$\forall_A(A \subseteq \mathbb{R} \to \text{замкнутое}(A) \leftrightarrow \forall_x(\text{предельныточка}(x, A) \to x \in A))$

Прием применяется к содержащему неизвестные подутверждению условия задачи на описание - корневому либо расположенному под корневым отрицанием. Уровень срабатывания достаточно высок, чтобы убедиться в отсутствии возможностей учета данного условия без его расшифровки.

Спецификация приема имеет вид "тип(содержится)", "направл(N)". Справочник "заголовокприема" указывает уровень срсбатывания 5 и вводит фильтры "тип(описать)", "условие", "отрицание", "не(цель(редакция))", "не(цель(исследовать))", "не(известно(теквхожд))". Этого почти достаточно.

Число приемов данного типа - 24.

(c) Группировка дизъюнктивных членов относительно неизвестных.

Пример:

$\forall_{ab}(b \leq |a| \leftrightarrow b \leq a \ \vee \ b \leq -a)$

Прием применяется к подутверждению условия задачи на описание. Замена выполняется справа налево. Выражение b содержит неизвестные, выражение a - не содержит.

Спецификация приема имеет вид "тип(имп)", "направл(N)", "неизвестные($x_1 \ldots x_n$)". Справочник "заголовокприема" указывает уровни срабатывания 1,3,5 и вводит фильтры "условие", "тип(описать)", "не(цель(пример))", "не(известно(b))", "известно(a)". Он требует доработки.

Число приемов данного типа - 4.

(d) Группировка неизвестных членов в одной части условия либо посылки задачи.

i. Группировка всех неизвестных членов в одной части условия задачи на описание либо посылки задачи на исследование.

Пример:

$\forall_{ab}(a - \text{число} \to a = b \leftrightarrow a - b = 0)$

Прием применяется к подутверждению условия задачи на описание либо посылки задачи на исследование, не имеющих цели "известно". Допускаются только надтермы с заголовками "существует", "и", "или". Выражения a, b содержат неизвестные. Если одно из них - неизвестная, то она входит в другое. Имеется множество дополнительных фильтров.

Спецификация приема имеет вид "тип(неизвестные)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 4 и

вводит фильтры "или(и(тип(описать)условие) тип(исследовать))", "не(цель(редакция))", "внешзнак(и или существует)", "не(известно(a))", "не(известно(b))", "не(контекст(вид(a значение(x3 x4))неизвестная(x3) не(входит(x3 b)) переменная(x4) контекст(цель(x5)заголовок(x5 связка) входит(x4 x5))))", "или(не(тип(исследовать)) и(не(контекст(вид(a значение(x3 x4)) неизвестная(x3) известно(x4) не(вхождениетерма(b a))))не(контекст(вид(b значение(x3 x4)) неизвестная(x3) известно(x4) не(вхождениетерма(a b))))))", "не(контекст(вид(b значение(x3 x4))неизвестная(x3) не(входит(x3 a)) переменная(x4) контекст(цель(x5)заголовок(x5 связка) входит(x4 x5))))", "или(не(неизвестная(a))входит(a b))", "или(не(неизвестная(b))входит(b a))". Требуется доработка.

Число приемов данного типа - 4.

ii. Группировка нескольких неизвестных членов в одной части уравнения задачи на исследование для приведения подобных членов относительно неизвестного подвыражения.

Пример:

$\forall_{abcdepq}(ab/p + c = db/q + e \leftrightarrow (a/p - d/q)b + c = e)$

Прием применяется к посылке задачи на исследование, имеющей цель "известно". Выражение b содержит неизвестные; выражения a, d, p, q - не содержат.

Спецификация приема имеет вид "тип(последнийтерм)", "направл(N)", "неизвестная(x)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(исследовать)", "цель(известно)", "посылка", "корень", "не(известно(b))", "известно(a)", "известно(d)", "известно(p)", "известно(q)". Требуется доработка.

Число приемов данного типа - 6.

(e) Применение нормализатора стандартной формы к неизвестному подвыражению.

i. Применение нормализатора стандартной формы для разгруппировки неизвестного подвыражения условия задачи на описание.

Пример:

$\forall_{abcdfg}(f = a(b + c)^d \to a(b + c)^d = g \leftrightarrow f = g)$

Прием применяется к условию задачи на описание. Сумма $b + c$ содержит неизвестные. Показатель степени d идентифицируется с натуральной константой, меньшей 6. Допускается вырожденный случай показателя единица, но тогда a отлично от единицы. Выражение g отлично от нуля и не содержит неизвестных. Правая часть антецедента обрабатывается нормализатором раскрывания скобок "стандплюс" и нормализатором выражений с неизвестными "уравнплюс". Либо число неизвестных более одной, либо после раскрывания скобок глубина вхождений неизвестных уменьшается, либо $d = 1$, а каждое из выражений a, $b + c$ линейно относительно некоторой неизвестной, либо $b + c$ имеет дробное слагаемое с неизвестным знаменателем. Имеется множество других фильтров.

Спецификация приема имеет вид "тип(таблзначение)", "направл(N)", "оператор(s)", где s - название нормализатора стандартной формы. Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие", "корень". Он не прорабатывался.

Число приемов данного типа - 6.

ii. Применение нормализатора стандартной формы для разгруппировки неизвестного подвыражения посылки задачи на исследование.

Пример:

$\forall_{abcefg}(f = a(b+c)+e \to a(b+c)+e = g \leftrightarrow f = g)$

Прием применяется к посылке задачи на исследование, имеющей цель "известно". Левая часть преобразуемого равенства содержит более одной неизвестной, а результат f обработки ее нормализатором "стандплюс" содержит единственную неизвестную. Отсутствуют невырожденные числовые атомы.

Спецификация приема имеет вид "тип(суп)", "направл(N)", "оператор(s)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2 и вводом фильтров "тип(исследовать)", "цель(известно)", "корень". Он не прорабатывался.

Число приемов данного типа - 3.

(f) Упрощение относительно неизвестных одной посылки задачи на исследование с помощью другой.

i. Преобразование одной посылки задачи на исследование с помощью другой для получения уравнения относительно неизвестных величин внешней задачи на описание.

Пример:

$\forall_{abnpqrsx}(a/b = x \to pa^n/q = rb^n/s \leftrightarrow psx^n = qr)$

Прием применяется к посылке задачи на исследование, имеющей цель "известно". Переменная n идентифицируется с натуральной константой. Выражения p, q, r, s не содержат неизвестных. Выражение x содержит неизвестные, но только такие, которые являются неизвестными внешней задачи на описание. Хотя бы одно из выражений a, b содержит неизвестные, не являющиеся неизвестными внешней задачи на описание.

Спецификация приема имеет вид "тип(цепьзадач)", "направл(N)", "неизвестные($x_1 \ldots x_n$)", "антецедент(k)". Здесь k указывает номер антецедента, идентифицируемого со второй посылкой. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(исследовать)", "корень", "цель(известно)", "или(не(внешнеизв(a))не(внешнеизв(b)))", "или(не(известно(a)) не(известно(b)))", "внешнеизв(x)", "известно(n)", "известно(p)", "известно(q)", "известно(r)", "известно(s)". Вводится также указатель "коммутативно(фикс(1))", указывающий, что при идентификации первого антецедента перестановка частей равенства не допускается. Этого почти достаточно.

Число приемов данного типа - 3.

ii. Преобразование одной посылки задачи на исследование с помощью другой для упрощения относительно неизвестных подвыражений.

Пример:

$\forall_{abcdexy}(ax^2+b=c\ \&\ x+y=e \to ay^2+b=d \leftrightarrow a(x-y)e=c-d)$

Прием применяется к посылке задачи на исследование. Выражения x, y содержат неизвестные, выражения a, c, d, e не содержат.

Спецификация приема имеет вид "тип(перечислцелые)", "направл(N)", "неизвестные(x_1)", ..., "неизвестные(x_n)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(исследовать)", "корень", "цель(известно)", "не(известно(x))", "не(известно(y))", "известно(a)", "известно(c)", "известно(d)", "известно(e)". Этого почти достаточно.

Число приемов данного типа - 6.

(g) Упрощение условия задачи на описание с помощью другого условия либо посылки.

i. Эквивалентное преобразование одного условия с помощью другого для получения условия более простого типа.

Пример:

$\forall_{abcdefgh}(ag+b=c\ \&\ \neg(g=0)\ \&\ f=bh-dg-ch+eg \to ah+d=e \leftrightarrow f=0)$

Прием применяется к условию задачи на описание, имеющей более одной неизвестной. Первый антецедент идентифицируется с другим условием. Выражение a содержит неизвестные; выражения g, h не содержат. Правая часть последнего антецедента обрабатывается нормализатором раскрывания скобок "стандплюс". Проверяется, что число слагаемых левой части заменяемого терма, имеющих своим сомножителем радикал с неизвестными, больше такого числа для левой части заменяющего терма.

Спецификация приема имеет вид "тип(конкатенация)", "направл(N)", "неизвестные(...)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2 и вводом фильтров "тип(описать)", "условие", "корень", "не(известно(корень))", "неизвестные(2)", "не(цель(редакция))", "не(Входит(независит цели))", "не(Входит(известно цели))". Он не прорабатывался.

Число приемов данного типа - 6.

ii. Эквивалентное преобразование одного условия с помощью другого для обеспечения декомпозиции уравнения.

Пример:

$\forall_{abcdxy}(\neg(a=0)\ \&\ ax+b=0\ \&\ ad-bc=yp \to cx+d=0 \leftrightarrow y=0 \vee p=0)$

Прием применяется к условию задачи на описание. Второй антецедент идентифицируется с другим условием. Переменные x, y - различные

неизвестные; выражения a, c неизвестных не содержат. Существует такая неизвестная, что каждое слагаемое выражений b, d имеет своим сомножителем ее либо ее степень. Левая часть третьего антецедента обрабатывается нормализатором упрощенного разложения на множители "факторизация". Выражение p линейно относительно неизвестных, а выражение d - нелинейно.

Спецификация приема имеет вид "тип(замечание)", "направл(N)", "неизвестные(...)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(описать)", "условие", "корень","не(известно(корень))", "неизвестные(2)", "не(цель(редакация))", "не(Входит(независит цели))", "не(Входит(известно цели))". Он не прорабатывался.

Число приемов данного типа - 4.

Преобразование посылки к виду дизъюнкции

1. Переход к дизъюнкции в посылке задачи на доказательство или задаче на исследование, имеющей цель "противоречие".

   Пример:

   $\forall_{abc}(a \in b \cup c \leftrightarrow a \in b \ \vee \ a \in c)$

   Прием применяется к посылке задачи на доказательство либо задачи на исследование, имеющей цель "противоречие". Выведенная дизъюнкция снабжается комментарием "разборслучаев".

   Спецификация приема имеет вид "тип(минимакс)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) и(тип(исследовать) цель(противоречие)))","корень". Создается указатель "примечание(разборслучаев)". Этого достаточно.

   Число приемов данного типа - 6.

Упрощение утверждений с описателями

1. Исключение содержащего неизвестные описателя с помощью кванторов.

   Пример:

   $\forall_{Pa}(\sup(\mathrm{set}_x(P(x))) \le a \leftrightarrow \forall_x(P(x) \to x \le a))$

   Прием применяется к условию задачи на описание. Утверждение $P(x)$ содержит неизвестные.

   Спецификация приема имеет вид "тип(Угол)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "условие", "тип(описать)", "корень", "не(известно(фикс(0 1 1 1 2)))". Этого достаточно.

   Число приемов данного типа - 4.

2. Использование эквивалентности для упрощения описателя.

   Пример:

   $\forall_{abcdef}(\text{образ}(\lambda_x(-f(x), x - \text{число}), a) \subseteq [b, c] \leftrightarrow \text{образ}(\lambda_x(f(x), x - \text{число}), a) \subseteq [-c, -b])$

   Здесь d, e - указатели типа конца промежутка.

   Спецификация приема имеет вид "тип(посылка)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2. Этого достаточно.

   Число приемов данного типа - 7.

3. Разрешение относительно переменной, связанной внешним описателем.

   (a) Явное разрешение утверждения относительно переменной, связанной внешним описателем.

   Пример:

   $\forall_{ikmn}(k - \text{целое} \rightarrow k - i \in \{m, \ldots, n\} \leftrightarrow i \in \{k - n, \ldots, k - m\})$

   Переменная i связана внешним описателем. Все переменные выражений k, m, n - свободные.

   Спецификация приема имеет вид "тип(параллели)", "направл(N)", "переменная(i)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "переменная(i)", "контекст(подчинено(теквхожд x2) символ(x2 класс отображение) входит(i связприставка(x2)))", "свобоперанд(фикс(0 1 1 1))", "свобоперанд(фикс(0 1 2 1))", "свобоперанд(фикс(0 1 2 2))". Этого достаточно.

   Число приемов данного типа - 3.

   (b) Явное разрешение относительно варьируемой переменной при вычислении операции над семейством.

   Данный тип представляет собой версию рассмотренного выше типа "Явное разрешение утверждения относительно переменной, связанной внешним описателем", имеющую чуть более сильную мотивацию срабатывания. Пример:

   $\forall_{abci}(b < 0 \rightarrow a + bi < c \leftrightarrow (c - a)/b < i)$

   Переменная i входит в связывающую приставку внешнего описателя "отображение", расположенного непосредственно под одноместной операцией. Выражения a, b, c не содержат переменных этой связывающей приставки.

   Спецификация приема имеет вид "тип(указательвхождения)", "направл(N)", "неизвестная(i)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "переменная(i)", "контекст(подчинено(теквхожд x4) символ(x4 отображение) входит(i связприставка(x4)) контекст(операнд(x5 x4) равно(количествооперандов(x5)1)) не(пересекаются(a связприставка(x4))) не(пересекаются(b связприставка(x4))) не(пересекаются(c связприставка(x4))))". Этого достаточно.

   Число приемов данного типа - 5.

(c) Группировка элементов описание класса, явно разрешенных относительно переменной связывающей приставки.

Пример:

$\forall_{abx}(x \le a \ \& \ x \le b \leftrightarrow x \le \min(a,b))$

Переменная x входит в связывающую приставку внешнего описателя "класс". Выражения a, b не содержат переменных этой приставки.

Спецификация приема имеет вид "тип(десделение)", "направл(N)", "переменная(x)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "переменная(x)", "не(входит(x a))", "не(входит(x b))", "контекст(подчинено(теквхожд x3) символ(x3 класс) входит(x связприставка(x3)))". Создается также указатель "дизъюнктоперанд". Требуется доработка.

Число приемов данного типа - 13.

Упрощение кванторов

1. Декомпозиция кванторной импликации в конъюнкцию нескольких кванторных импликаций и элементарных утверждений.

   Пример:

   $\forall ABCDa(\forall_x(A(x) \to B(x, \text{индикатор}(C,D,a)(x))) \leftrightarrow \forall_x(A(x) \ \& \ x \in D \to B(x,a)) \ \& \ \forall_x(A(x) \ \& \ \neg(x \in D) \to B(x,0)))$

   Рассматриваемое вхождение терма "индикатор(...)" не расположено внутри квантора либо описателя, размещенного в консеквенте заменяемой импликации и связывающего какие-либо переменные этого терма.

   Спецификация приема имеет вид "тип(Примечпосылки)", "направл(N)". В данном примере добавляется элемент "указатель(вхождение(B))". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. Он не прорабатывался.

   Число приемов данного типа - 4.

2. Упрощение квантора за счет перехода к новым переменным.

   Пример:

   $\forall_{ABn}(\exists_{xy}(A(x^n) \ \& \ x - \text{число} \ \& \ 0 < x \ \& \ B(y)) \leftrightarrow \exists_{xy}(A(x) \ \& \ x - \text{число} \ \& \ 0 < x \ \& \ B(y)))$

   Спецификация приема имеет вид "тип(нормодз)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1 и вводом ряда необходимых указателей. Он требует доработки.

   Число приемов данного типа - 14.

Упрощение проверяемого условия

1. Расшифровка по определению проверяемого условия.

   Пример:

   $\forall_{ABf}$(биекция$(f, A, B) \leftrightarrow \mathrm{Dom}(f) = A$ & $\mathrm{Val}(f) = B$ & взаимнооднозначно(f))

   Прием применяется к подутверждению условия задачи на доказательство либо задачи на описание, имеющей цель "проверка".

   Спецификация приема имеет вид "тип(вставкафрагментов)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "условие", "или(тип(доказать) и(тип(описать) цель(проверка)))". Этого достаточно.

   Число приемов данного типа - 7.

2. Преобразование условия задачи на доказательство, исключающее сложное выражение.

   Пример:

   $\forall_{acde}(\neg(c = 0) \rightarrow c\sup(a) + d = e \leftrightarrow$ верхняягрань$((e - d)/c, a)$ & $\forall_x(x < (e - d)/c$ & $x -$ число $\rightarrow \exists_y(y \in a$ & $x < y)))$

   Прием применяется к условию задачи на доказательство. В этом условии отсутствуют выражения с заголовками "суп", "инф", более длинные, чем выражение $\sup(a)$. Условие не имеет выражений с большей оценкой сложности, чем данное выражение.

   Спецификация приема имеет вид "тип(исключение)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "условие", "тип(доказать)", "корень". Он не прорабатывался.

   Число приемов данного типа - 9.

3. Преобразование условия задачи на доказательство к виду, позволяющему исключить сложное выражение.

   Пример:

   $\forall_{abcde}(0 \le b$ & $0 \le d$ & $a \le 0$ & $c \le 0$ & $0 \le e \rightarrow 0 \le a\sqrt{b} + c\sqrt{d} + e \leftrightarrow 0 \le e^2 - a^2b - c^2d - 2ac\sqrt{b}\sqrt{d})$

   Прием применяется к условию задачи на доказательство. Хотя бы одно из выражений b, d имеет своим сомножителем сумму либо является суммой. Остаточная сумма e не имеет слагаемых, среди сомножителей которых встречаются неконстантные радикалы. После преобразования остается единственный радикал, от которого далее можно будет избавиться.

   Спецификация приема имеет вид "тип(усмчетное)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "условие", "тип(доказать)", "отрицание". Он не прорабатывался.

   Число приемов данного типа - 4.

4. Применение нормализатора приведения к заданным заголовкам для декомпозиции условия задачи на доказательство.

   Пример:

   $\forall_{cd}(d = c \to 0 \leq c \leftrightarrow 0 \leq d)$

   Прием применяется к условию задачи на доказательство. Правая часть антецедента обрабатывается нормализатором разложения на множители "видумножение". Выражение c представляет собой сумму, а выражение d - нет. Созданы две версии приема. В первой, срабатывающей на уровне 1, предполагается, что c имеет дробное слагаемое. Это гарантирует невырожденный результат применения нормализатора. Во второй версии, срабатывающей на уровне 5, такое предположение отсутствует.

   Спецификация приема имеет вид "тип(нормарктангенс)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 5 и вводом фильтров "тип(доказать)", "условие", "корень". Он не прорабатывался.

   Число приемов данного типа - 4.

Специальная стандартизация

Как и в случае тождеств, приемы данного раздела реализуют ту специальную стандартизацию утверждений, решение о которой принимается путем анализа имеющихся теорем предметной области. Этим должны будут заниматься процедуры алгоритмизации предметной области, развитие которых находится лишь на начальной стадии. По-видимому, большинство типов приемов данного раздела подлежат разгруппировке в подтипы, преследующие более конкретные цели. Фактически, раздел просто является временным хранилищем приемов, типы которых будут уточняться.

1. Группировка в одной части двуместного отношения всех операндов ассоциативно-коммутативной операции.

   Пример:

   $\forall_{ab}(a - \text{число} \ \& \ b - \text{число} \to a = b \leftrightarrow a - b = 0)$

   Ни одна из частей равенства не является нулевой. Либо равенство расположено в условии задачи, либо является посылкой задачи на исследование, либо расположено в посылке, имеющей вид отрицания данного равенства. Впрочем, имеется более десятка других фильтров, ограничивающих применение данного приема. Например, будет заблокирована попытка "испортить" равенство неизвестной известному выражению. Как иногда случается, приемы с простыми теоремами требуют крайне сложного управления. Это - как раз такой случай.

   Спецификация приема имеет вид "тип(группировка)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "или(не(тип(описать)) и(или(не(цель(редакция)) коммент(стандменьше)) не(Входит(известно цели))))", "не(символ(a 0))", "не(символ(b 0))", "или(отр

условие и(тип(исследовать) корень))", "или(и(тип(доказать) условие свобоперанд(теквхожд)) не(контекст(перестановка(a b x3 x4) числовойатом(x3 x5) равно(x3 x5) не(переменная(x5)) не(константа(x5)) не(вхождениетерма(x4 x3)))))", "не(контекст(перестановка(a b x3 x4) переменная(x3) подчинено(теквхожд x5) символ(x5 длялюбого существует класс отображение) входит(x3 связьприставка(x5))))", "или(тип(доказать) и(или(и(известно(теквхожд) не(контекст(вид(теквхожд равно(x3 x4)) переменная(x3) не(входит(x3 x4)) или(и(тип(описать) цель(редакция))не(отр))))) не(внешзнак(существует длялюбого и или не)))или(контекст(операнд(x3 теквхожд) символ(x3 не) антецедент(x3)) не(цель(редуцирование)))))", "или(не(тип(исследовать)) и(не(цель(анализфразы)) не(цель(текстоваязадача))))", "или(не(тип(преобразовать)) и(не(цель(нормуравн)) не(цель(нормализатор)) не(цель(упрощзнак)) не(цель(вычпрог))))". Он требует доработки.

Число приемов данного типа - 4.

2. Использование нормализатора для специальной стандартизации.

   (a) Попытка применения нормализатора приведения к заданным заголовкам для декомпозиции элементарного утверждения.

   Пример:

   $\forall_{kmnpq}(m = k \to m^p q|n \leftrightarrow k^p q|n)$

   Левая часть антецедента обрабатывается нормализатором разложения на множители, снабженным комментарием, указывающим необходимость разложения натуральных сомножителей в произведение степеней простых чисел. Проверяется, что результат k имеет своим заголовком символ "умножение" либо "степень"(с точностью до внешнего знака "минус"). Аналогичная стандартизация предпринимается для другого операнда отношения "делит". Таким образом подготавливается срабатывание приемов, заменяющих один целочисленный параметр на другой.

   Спецификация приема имеет вид "тип(номерэлемента)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 3. Он не прорабатывался.

   Число приемов данного типа - 3.

   (b) Применение нормализатора приведения к заданным заголовкам для декомпозиции условия задачи на описание, не содержащего неизвестных.

   Пример:

   $\forall_{ab}(b = a \to 0 < a \leftrightarrow 0 < b)$

   Прием применяется к условию задачи на описание, не содержащему неизвестных и не используемому для сопровождения по о.д.з. Выражение a содержит символ "дробь". Правая часть антецедента обрабатывается нормализатором разложения на множители "видумножение". Результат имеет своим заголовком один из символов "умножение", "степень", "дробь" (с точностью до внешнего знака "минус").

Спецификация приема имеет вид "тип(текприем)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(описать)", "корень", "условие", "не(сопровождение)", "известно(корень)". Он не прорабатывался.

Число приемов данного типа - 3.

(c) Применение нормализатора стандартной формы для контекстной стандартизации.

Пример:

$\forall_{abcdefg}(f = a(b+c)^d + e \to a(b+c)^d + e = g \leftrightarrow f = g)$

Прием применяется к условию задачи на описание. Переменная d идентифицируется с натуральной константой, меньшей 6. Условие содержит целочисленную неизвестную. Правая часть антецедента обрабатывается нормализатором стандартной формы "стандплюс", обеспечивающим раскрывание скобок. Сумма $b+c$ не константная и не содержит неизвестных. Смысл данной контекстной стандартизации заключается в подготовке возможности анализа делимости коэффициентов уравнения.

Спецификация приема имеет вид "тип(контрользамены)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. Он не прорабатывался.

Число приемов данного типа - 6.

3. Стандартизация посылок задачи на исследование.

(a) Специальная стандартизация посылки задачи на исследование.

Примеры:

$\forall_{ab}(a + b = 0 \to \sin a = 0 \leftrightarrow \sin b = 0)$

Прием применяется к посылке задачи на исследование, имеющей цель "известно". Выражение a представляет собой сумму, а b - нет.

$\forall_{abcd}(a = \text{путь}(c)\ \&\ \text{обратныйпуть}(\text{путь}(c)) = \text{путь}(d) \to$
$b = \text{обратныйпуть}(a) \leftrightarrow b = \text{путь}(d))$

Выражение b имеет заголовок "Путь", т.е. задает ориентированную кривую, по которой перемещается материальная точка.

Спецификация приема имеет вид "тип(ответ)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "тип(исследовать)", "корень". Он не прорабатывался.

Число приемов данного типа - 6.

(b) Специальная группировка посылок задачи на исследование.

Пример:

$\forall_{abc}(b - \text{set}\ \&\ c - \text{set} \to \text{непрерывно}(a, b)\ \&\ \text{непрерывно}(a, c) \leftrightarrow$
$\text{непрерывно}(a, b \cup c))$

Прием применяется к паре посылок задачи на исследование, имеющей цель "непрерывно". Такая цель означает, что предпринимается исследование функции на непрерывность. Переменная a - неизвестная.

Спецификация приема имеет вид "тип(массив)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "тип(исследовать)", "корень". Он не прорабатывался.

Число приемов данного типа - 5.

4. Специальная стандартизация посылки задачи на описание.

   Пример:

   $\forall abcdefg($диффлагранжа$(f, a, \lambda_{xy}(bx^2/c + dx^2/e + g(y), h(y))) \leftrightarrow$
   диффлагранжа$(f, a, \lambda_{xy}((b/c + d/e)x^2 + g(y), h(y))))$

   Прием применяется к подутверждению посылки задачи на описание.

   Спецификация приема имеет вид "тип(входит)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "тип(описать)", "посылка". Он не доработан.

   Число приемов данного типа - 4.

5. Специальная стандартизация условия задачи на описание.

   (a) Специальная стандартизация условия задачи на описание.

   Это - наиболее общий тип данного подраздела. Остальные типы представляют собой его частные случаи. Пример:

   $\forall_{abcdf}(\mathrm{set}_x(g(x)\ \&\ x \in a) = d\ \&\ f = \lambda_x(h(x), g(x)) \rightarrow \mathrm{Max}(f, a, b, c) \leftrightarrow \mathrm{Max}(f, d, b, c))$

   Прием применяется к подутверждению условия задачи на описание. Второй антецедент идентифицируется с утверждением из контекста, определяющим функцию f от более чем одной переменной. Утверждение под описателем "класс", задающее область поиска максимума, разрешается вспомогательной задачей на описание относительно неизвестных x. Этим подготавливается возможность срабатывания последующих приемов.

   Спецификация приема имеет вид "тип(номерсимв)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "условие", "тип(описать)", "корень". Он не прорабатывался.

   Число приемов данного типа - 20.

   (b) Преобразование известного подвыражения условия задачи на описание для упрощения разрешения относительно неизвестных.

   Пример:

   $\forall_{abcdx}(d = ab + ac \rightarrow \sin x = ab + ac \leftrightarrow \sin x = d)$

Прием применяется к условию задачи на описание. Выражение x содержит неизвестные. Каждое из выражений b, c - синус либо косинус, с точностью до отбрасывания знака. Правая часть антецедента обрабатывается нормализатором "видумножение", причем результат d - синус либо косинус, с точностью до знака.

Спецификация приема имеет вид "тип(конецотрезка)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(описать)", "условие", "корень". Он не прорабатывался.

Число приемов данного типа - 4.

(c) Стандартизация параметрического описания в условии задачи на описание.

Пример:

$\forall_{abcdef}(a \le b \ \& \ b = ad + e \to \exists_n(f = (an + b)\pi/c \ \& \ n - \text{целое}) \leftrightarrow$
$\exists_n(f = (an + e)\pi/c \ \& \ n - \text{целое}))$

Прием применяется к условию задачи на описание. Выражение f содержит неизвестные; a, b, c - натуральные константы. Второй антецедент выполняет деление с остатком.

Спецификация приема имеет вид "тип(учетпасива)", "направл(N)", "неизвестная(...)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "условие", "тип(описать)", "корень", "не(известно(f))". Он не прорабатывался.

Число приемов данного типа - 13.

(d) Усиление ограничения на известные параметры в условии задачи на описание.

Заметим, что условия на неизвестные естественно ослаблять, а условия на параметры - усиливать, чтобы они расширяли возможности срабатывания приемов. В качестве примера берем теорему предыдущего пункта. Однако, она создает прием, применяемый не к подутверждению кванторной импликации, а непосредственно к не содержащему неизвестных условию задачи на описание. Как и ранее, используется нормализатор "нормцелаячасть", причем проверяется, что результат b не содержит символа "целаячасть". Проверяется также отсутствие в нем символов "суп", "инф".

Спецификация приема имеет вид "тип(нормверхпредел)", "направл(N)". Справочник "заголовокприема" указывает уровень 1 и вводит фильтры "тип(описать)", "условие", "корень", "известно(корень)". Требуется доработка.

Число приемов данного типа - 3.

6. Специальная стандартизация утверждения под описателем "класс".

Пример:

$\forall_n(n - \text{even} \leftrightarrow 2|n)$

Прием применяется к подутверждению выражения вида "card(set(...))". Параметры выражения n пересекаются со связывающей приставкой описателя "класс". Замена условия четности на условие делимости необходима для расширения класса приемов, которые могут сработать при нахождении мощности.

Спецификация приема имеет вид "тип(ответзадачи)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня 2. Он не прорабатывался.

Число приемов данного типа - 10.

7. Специальная стандартизация утверждения под описателем "отображение".

   Пример:

   $\forall_{abc}(b - \text{число } \& \ c - \text{число} \to a \in (b,c) \leftrightarrow a - \text{число } \& \ b < a \ \& \ a < c)$

   Прием применяется для перехода к стандартному обозначению промежутка интегрирования. a - переменная интегрирования, не входящая в выражения b, c.

   Спецификация приема имеет вид "тип(нормфакториал)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. Он не прорабатывался.

   Число приемов данного типа - 3.

Исключение сложных понятий

1. Преобразование утверждения со сложными отношениями, упрощающее проверку истинности этих отношений.

   Пример:

   $\forall_{afg}(g = \lambda_i(f(i), i - \text{число}) \to \text{сходится}(\lambda_n(\sum_{i=a}^{n} f(i), n - \text{натуральное})) \leftrightarrow$
   $\text{сходится}(\lambda_n(\sum_{i=a}^{n} g(i), n - \text{натуральное})))$

   Выражение $f(i)$ в правой части антецедента обрабатывается нормализатором "асимптоценка", которому передаются посылки "натуральное(i)" и "$i \to \infty$". В результате общий член ряда заменяется на свою асимптотическую оценку, что обычно сильно упрощает анализ сходимости. Предварительно проверяется, что $f(i)$ имеет своим обобщенным сомножителем какую-либо тригонометрическую функцию, либо логарифм, либо конечную сумму.

   Спецификация приема имеет вид "тип(условие)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2. Он не прорабатывался.

   Число приемов данного типа - 10.

2. Исключение сложного отношения.

   Пример:

$\forall_{acdf}(0 \le f(i)\ \&\ a = -(\lim_{i\to\infty} lnf(i)/lni)\ \&\ (a - \text{число}\ \&\ \neg(a-1=0)\ \vee\ a = \infty\ \vee\ a = -\infty) \to \text{сходится}(\lambda_n(\sum_{i=c}^{n} f(i), n - \text{натуральное})) \leftrightarrow 1 < a)$

Проверяется наличие среди обобщенных множителей общего члена $f(i)$ логарифма, содержащего i. Первый антецедент обрабатывается проверочным оператором, которому передаются дополнительные посылки "i – натуральное" и "$i \to \infty$". Предел во втором антецеденте вычисляется нормализатором "нормпредел".

Спецификация приема имеет вид "тип(внешоператор)", "направл(N)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Он не прорабатывался.

Число приемов данного типа - 7.

Явное разрешение безотносительно к неизвестным

1. Разрешение относительно неконстантного выражения.

   (a) Разрешение относительно неконстантного выражения.

   Пример:

   $\forall_{ab}(b \le -a \leftrightarrow a \le -b)$

   Замена выполняется справа налево. Выражение b неконстантное, выражение a - константное.

   Спецификация приема имеет вид "тип(точкаотрезка)", "направл(N)", "терм(t)". Здесь t - неконстантное выражение, относительно которого выполняется разрешение. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "константа(a)", "не(константа(b))". Этого достаточно.

   Число приемов данного типа - 8.

   (b) Разрешение сопровождающего утверждения ответа задачи на описание относительно неконстантного выражения.

   Пример:

   $\forall_{abc}(0 < b \to ab \le c \leftrightarrow a \le c/b)$

   Прием применяется к подутверждению не содержащего неизвестных условия задачи на описание, имеющей цель "редакция". Переменная b идентифицируется с непустым произведением всех константных сомножителей. Остаточное произведение a неконстантное. Выражение c константное.

   Спецификация приема имеет вид "тип(удалениепримечания)", "направл(N)", "переменные(x)". Здесь x идентифицируется с неконстантным термом. Справочник "заголовокприема" указывает уровни срабатывания 1 и 5. Он вводит фильтры "условие", "тип(описать)", "цель(редакция)", "известно(теквхожд)", "не(цель(новыенеизвестные))", "или(не(коммент(стандменьше)) уровень(5))", "не(константа(a))", "константа(c)", "не(заголовок(b 1))". Вводятся также указатели "сопровождение", "замечание(стандменьше)", "перечень(b константа(b))". Этого почти достаточно.
   Число приемов данного типа - 36.

2. Разрешение элементарного утверждения относительно переменной, связываемой внешним квантором.

   Пример:

   $\forall_{abx}(\neg(a=0) \to ax+b=0 \leftrightarrow x=-b/a)$

   Переменная x связана внешним квантором и не входит в выражения a, b. В случае квантора существования преобразуемое равенство является конъюнктивным членом подкванторного утверждения, в случае квантора общности - антецедентом.

   Спецификация приема имеет вид "тип(усмнеделит)", "неизвестная(x)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "переменная(x)", "или(входит(длялюбого корень)входит(существует корень))", "или(контекст(операнд(x3 теквхожд) символ(x3 и) операнд(x4 x3) символ(x4 существует) входит(x23 связприставка(x4))) контекст(операнд(x3 теквхожд) символ(x3 длялюбого) входит(x23 связприставка(x3)) не(равно(вхождение(теквхожд) последнийоперанд(x3))))", "не(входит(x a))", "не(входит(x b))", "или(не(тип(преобразовать))посылка не(цель(нормтеорема)))". Этого достаточно.

   Число приемов данного типа - 6.

3. Явное разрешение кванторной посылки относительно функциональной переменной.

   Пример:

   $\forall_{afk}(k - \text{целое} \to \forall_n(n - \text{натуральное} \ \& \ k \le n \to f(n) = af(n-1)) \leftrightarrow$
   $\forall_n(n - \text{натуральное} \ \& \ k-1 \le n \to f(n) = f(k-1)a^{n-k+1}))$

   Прием применяется к посылке задачи.

   Спецификация приема имеет вид "тип(нормнижняяграньN)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "посылка", "корень". Требуется доработка.

   Число приемов данного типа - 4.

Числовые атомы

В этом разделе многие типы приемов перекрываются и различаются лишь степенью мотивированности срабатывания. Это позволяет доводчику приемов выбирать подходящий тип для разделения желательных и нежелательных срабатываний. Изначально генератор приемов вводит самый общий тип, а затем "сужает" его в процессе доводки по задачнику.

1. Выражение невырожденного числового атома через численные параметры в посылке задачи.

   (a) Выражение невырожденного числового атома через численные параметры в задаче на исследование.

i. Выражение невырожденного числового атома через численные параметры в задаче на исследование.

   Пример:

   $\forall_{ABab}(\neg(a=0) \to al(AB)=b \leftrightarrow l(AB)=b/a)$

   Прием применяется к посылке задачи на исследование. Выражения a, b не содержат невырожденных числовых атомов. Уровень срабатывания приема крайне высокий. По существу, это остаточное действие. На меньших уровнях срабатывают более мотивированные приемы разрешения относительно расстояний.

   Спецификация приема имеет вид "тип(запись)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(исследовать)", "корень", "не(контекст(список(x3 a b) числовойатом(x3 x4) не(переменная(x4))))". Этого почти достаточно.

   Число приемов данного типа - 3.

ii. Выражение через численные параметры числового атома, встречающегося невырожденным образом еще в одном уравнении.

   Пример:

   $\forall_{ABabc}(\neg(a=0) \to al(AB)+b=c \leftrightarrow l(AB)=(c-b)/a)$

   Прием применяется к посылке задачи на исследование. Существует другая посылка задачи - уравнение, в котором встречается выражение $l(AB)$, причем его вхождение не является корневым операндом. Выражения a, b, c не имеют невырожденных числовых атомов.

   Спецификация приема имеет вид "тип(выборпозиции)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(исследовать)", "корень", "контекст(посылка(x4) заголовок(x4 равно) не(равно(x4 корень)) вхождениетерма(x4 терм(расстояние(x26 x27)) x5) не(контекст(операнд(x6 x5) символ(x6 равно))))", "не(контекст(список(x4 a b c) числовойатом(x4 x5) не(переменная(x5))))", "или(не(заголовок(b 0)) не(заголовок(a 1)))". Этого почти достаточно.

   Число приемов данного типа - 5.

iii. Выражение через численные параметры невырожденного числового атома, являющегося единственным невырожденным числовым атомом еще одного уравнения.

   Пример:

   $\forall_{ABCabc}(\neg(a=0) \to a\angle(ABC)+b=c \leftrightarrow \angle(ABC)=(c-b)/a)$

   Прием применяется к посылке задачи на исследование. Выражение $\angle(ABC)$ встречается еще в одном уравнении, причем это уравнение не имеет других невырожденных числовых атомов. Выражения a, b, c не содержат невырожденных числовых атомов.

   Спецификация приема имеет вид "тип(Интеграл)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "тип(исследовать)", "корень", "конец(контекст(новаяпосылка(x4) заголовок(x4 равно) вхождениетерма(x4 фикс(0 1 1 1 2))

не(контекст(числовойатом(x4 x5) не(переменная(x5)) не(равно(x5 фикс(0 1 1 1 2))))))", "не(контекст(список(x4 a b c) числовойатом(x4 x5) не(переменная(x5))))", "или(не(заголовок(b 0))не(заголовок(a 1)))". Этого достаточно.
Число приемов данного типа - 3.

(b) Выражение числового атома через численные параметры в посылке задачи на доказательство.

Пример:

$\forall_{ABabcd}(\neg(a=0) \to al(AB)^2/c + d = b \leftrightarrow l(AB) = \sqrt{(b-d)c/a}$ &
$0 \le a(b-d)c)$

Прием применяется к посылке задачи на доказательство. Выражения a, b, c, d не содержат невырожденных числовых атомов.

Спецификация приема имеет вид "тип(вхождениетерма)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит посылки "посылка", "тип(доказать)", "корень", "не(контекст(список(x5 a b c d) числовойатом(x5 x6) не9переменная(x6)))". Этого достаточно.

Число приемов данного типа - 3.

2. Выражение одного невырожденного числового атома через другие.

(a) Выражение одного числового атома через другой.

i. Выражение одного невырожденного числового атома через другой в посылках задачи на доказательство либо на исследование.

Пример:

$\forall_{ABKabci}(\neg(a=0) \to a{\cdot}\text{крд}(A,K,i){+}b{\cdot}\text{крд}(B,K,i) = c \leftrightarrow \text{крд}(A,K,i) = (c - b\cdot\text{крд}(B,K,i))/a)$

Прием применяется к подутверждению посылки задачи на доказательство либо на исследование. Каждое из выражений a, b, c либо не содержит неизвестных, либо имеет тип "внешнеизв", т.е. содержит лишь неизвестные внешней задачи на описание.

Спецификация приема имеет вид "тип(блокнормализации)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "или(тип(доказать)тип(исследовать))", "посылка", "или(известно(a) внешнеизв(a))", "или(известно(b) внешнеизв(b))", "или(известно(c) внешнеизв(c))". Этого достаточно.

Число приемов данного типа - 6.

ii. Выражение одного числового атома через другой в задаче на исследование, чтобы уменьшить количество невырожденных числовых атомов в уравнении с численной неизвестной.

Пример:

$\forall_{ABCDab}(\neg(a=0) \to al(AB) + bl(CD) = 0 \leftrightarrow l(AB) = -bl(CD)/a)$

Прием применяется к посылке задачи на исследование. Выражение a не содержит неизвестных. Выражение b либо не содержит неизвестных, либо имеет тип "внешнеизв". Существует другое уравнение задачи, в которое входят оба числовых атома $l(AB)$ и $l(CD)$, причем оно также содержит неизвестную внешней задачи на описание.

Спецификация приема имеет вид "тип(видпеременной)", "направл(N)", "известно(a)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(исследовать)", "корень", "известно(a)", "или(известно(b) внешнеизв(b))", "конец(контекст(посылка(x3) заголовок(x3 равно) не(равно(x3 корень)) вхождениетерма(x3 $l(AB)$) вхождениетерма(x3 $l(CD)$) неизвестная(x4 внешописать) входит(x4 x3)))". Этого достаточно.

Число приемов данного типа - 5.

iii. Выражение одного числового атома через другой, чтобы получить уравнение с единственным числовым атомом.

Пример:

$\forall_{ABCDabc}(\neg(a=0) \to al(AB)+bl(CD)=c \leftrightarrow l(AB)=(c-bl(CD))/a)$

Прием применяется к посылке задачи на доказательство либо на исследование. Выражения a, b, c не содержат неизвестных. Существует другое уравнение задачи, содержащее числовые атомы $l(AB)$, $l(CD)$ и не содержащее прочих невырожденных числовых атомов.

Спецификация приема имеет вид "тип(усм)", "направл(N)", "известно(a)", "известно(b), "известно(c)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "или(тип(доказать) тип(исследовать))", "посылка", "корень", "известно(a)", "известно(b)", "известно(c)", "конец(контекст(посылка(x4) заголовок(x4 равно) не(равно(x4 корень)) вхождениетерма(x4 $l(AB)$) вхождениетерма(x4 $l(CD)$) не(контекст(числовойатом(x4 x5) не(равно(x5 фикс(0 1 1 1 2))) не(равно(x5 фикс(0 1 1 2 2))) не(переменная(x5))))))". Этого достаточно.

Число приемов данного типа - 6.

(b) Выражение одного числового атома через другие.

i. Выражение числового атома, встречающегося в другом уравнении задачи на исследование, не являющемся равенством двух числовых атомов, через более простые атомы.

Пример:

$\forall_{abcd}(\neg(a=0) \to aS(d)+b=c \leftrightarrow S(d)=(c-b)/a)$

Прием применяется к посылке задачи на исследование, имеющей цель "известно". Существует вхождение выражения $S(d)$ в другое уравнение задачи, не являющееся операндом равенства двух числовых атомов. Ни одно из выражений a, b, c не имеет невырожденного числового атома, оценка сложности которого больше 4.

Спецификация приема имеет вид "тип(разделысимволов)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(исследовать)", "цель(известно)", "корень",

"не(контекст(список(x5 a b c) числовойатом(x5 x6) не(переменная(x6)) не(меньше(Оценка(x6)13))))", "контекст(новаяпосылка(x5) заголовок(x5 равно) вхождениетерма(x5 фикс(0 1 1 1 2)x6) не(контекст(подтерм(равно(теквхожд(x6)x7)) контекст(числовойатом(x7 x8) равно(x7 x8))))", "или(не(заголовок(b 0)) не(заголовок(a 1)))". Этого достаточно. Заметим, что оценка сложности символа "площадь" равна 13.

Число приемов данного типа - 3.

ii. Выражение невырожденного числового атома через атомы других типов в задаче на исследование.

Пример:

$\forall_{Kabci}(\neg(a = 0) \to a \cdot \text{крд}(c, K, i) + b = d \leftrightarrow \text{крд}(c, K, i) = (d - b)/a)$

Прием применяется в задаче на исследование. Выражение d не содержит неизвестных, выражение a не содержит невырожденных числовых атомов, выражение b не содержит символа "крд".

Спецификация приема имеет вид "тип(базавхождения)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(исследовать)", "цель(известно)", "корень", "не(контекст(список(x5 a b d) вхождениетерма(x5 крд(c K i))))", "или(не(заголовок(b 0)) не(заголовок(a 1)))". Требуется доработка.

Число приемов данного типа - 7.

3. Стандартизация равенства с числовыми атомами в посылке задачи на исследование либо на доказательство.

   (a) Упрощение посылки задачи на исследование либо на доказательство относительно невырожденных числовых атомов.

   Пример:

   $\forall_{ABabcd}(0 \le a \;\&\; 0 \le b \;\&\; 0 \le d \;\&\; c = d^2 \to al(AB)^2 = bc \leftrightarrow \sqrt{a}l(AB) = \sqrt{b}d)$

   Прием применяется к посылке задачи на доказательство либо на исследование. Выражения a, b не содержат неизвестных. Переменная c идентифицируется с произведением всех сомножителей, содержащих неизвестные. Последний антецедент, выделенный указателем "идентификатор", усматривает в c полный квадрат.

   Спецификация приема имеет вид "тип(исключениеоперанда)", "направл(N)", "см(известно(a) известно(b))". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "корень", "известно(a)", "известно(b)". Он почти не проработан.

   Число приемов данного типа - 16.

   (b) Преобразование посылки для последующего разрешения относительно невырожденных числовых атомов.

   Пример:

   $\forall_{ABabcde}((al(AB) + b)(cl(AB) + d) = e \leftrightarrow acl(AB)^2 + (bc + ad)l(AB) + bd = e)$

Прием применяется к посылке задачи на доказательство либо на исследование. Выражения a, b, c, d не содержат неизвестных. Выражение e не содержит невырожденных числовых атомов и отлично от нуля.

Спецификация приема имеет вид "тип(4)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "посылка", "корень", "или(тип(доказать) тип(исследовать))". Требуется доработка.

Число приемов данного типа - 4.

(c) Переход к соотношению пропорциональности для числовых атомов.

Пример:

$\forall_{ABCDEFab}(a\angle(ABC) - b\angle(DEF) = 0 \leftrightarrow a\angle(ABC) = b\angle(DEF))$

Прием применяется к посылке задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(результподст)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "или(тип(доказать) тип(исследовать))", "посылка", "корень". Этого почти достаточно.

Число приемов данного типа - 3.

4. Комбинация уравнений с числовыми атомами.

(a) Комбинация уравнений с невырожденными числовыми атомами для получения уравнения с численными параметрами.

Пример:

$\forall_{ABCDEFabcpqr}(al(AB) + bl(CD) = cl(EF)\ \&\ aq - bp = 0\ \&\ \neg(l(EF) = 0)\ \&\ \neg(a = 0)\ \&\ \neg(p = 0) \rightarrow pl(AB) + ql(CD) = rl(EF) \leftrightarrow pc - ar = 0)$

Прием применяется к посылке задачи на доказательство либо на исследование. Выражения a, c, p, r не содержат невырожденных числовых атомов. Первый антецедент идентифицируется с другой посылкой. Второй антецедент выделен указателем "идентификатор". Его левая часть обрабатывается нормализатором раскрывания скобок.

Спецификация приема имеет вид "тип(пересечениесписков)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "посылка", "корень", "или(тип(доказать) и(тип(исследовать) цель(известно)))", "не(контекст(числовойатом(a x4) не(переменная(x4))))", "не(контекст(числовойатом(c x4) не(переменная(x4))))", "не(контекст(числовойатом(p x4) не(переменная(x4))))", "не(контекст(числовойатом(r x4) не(переменная(x4))))". Требуется доработка.

Число приемов данного типа - 9.

(b) Комбинация уравнений, позволяющая исключить часть вхождений числовых атомов.

Пример:

$\forall_{ABCDabcpqr}(\neg(a = 0)\ \&\ al(AB)l(CD)/b = c \rightarrow pl(AB)l(CD) + q = r \leftrightarrow bcp/a + q = r)$

Прием применяется к посылке задачи на доказательство либо на исследование. Второй антецедент идентифицируется с другой посылкой. Выражения a, b, c не содержат невырожденных числовых атомов.

Спецификация приема имеет вид "тип(альтоперанд)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "корень", "или(тип(доказать) тип(исследовать))", "не(контекст(список(x4 *a b c*) числовойатом(x4 x5) не(переменная(x5))))". Этого почти достаточно.

Число приемов данного типа - 4.

Координаты

1. Переформулировка утверждения через координаты.

   (a) Переформулировка подутверждения условия задачи через координаты.

   Пример:

   $\forall_{ABCDKabcdpqr}(\text{коорд}(A, K) = (a, b)\ \&\ \text{коорд}(B, K) = (c, d)\ \&\ \text{разныеточки}(A, B)\ \&\ \text{коорд}(\text{прямая}(CD), K) = \text{set}_{xy}(px + qy + r = 0\ \&\ x - \text{число}\ \&\ y - \text{число}) \rightarrow \text{прямая}(AB) \parallel \text{прямая}(CD) \leftrightarrow q(d - b) - p(a - c) = 0)$

   Прием применяется к подутверждению условия.

   Спецификация приема имеет вид "тип(сравн)", "направл(N)", "антецедент(4)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "условие", "или(не(тип(описать))не(цель(исследовать)))", "не(равно(*A B*))". Этого достаточно.

   Число приемов данного типа - 54.

   (b) Переформулировка подутверждения посылки задачи на исследование через координаты.

   Пример:

   $\forall_{Kpqr}(\text{одномерный}(p, K)\ \&\ \text{одномерный}(q, K)\ \&\ \text{одномерный}(r, K) \rightarrow p + q = r \leftrightarrow \text{крд}(p, K, 1) + \text{крд}(q, K, 1) = \text{крд}(r, K, 1))$

   Прием применяется в задачах на исследование.

   Спецификация приема имеет вид "тип(базаприемов)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтр "тип(исследовать)". Этого почти достаточно.

   Число приемов данного типа - 4.

2. Задача на преобразование, имеющая цель "класс".

   Напомним, что в таких задачах требуется исключить описатели "класс", "отображение". Они возникали в аналитической геометрии для перехода к элементарному описанию множества точек. Сначала вводятся координаты и составляется уравнение этого множества, затем - уравнение исключается и множество выражается некоторым стандартным образом без привлечения описателей.

(a) Исключение вспомогательных параметров в условии задачи на преобразование, имеющей цель "класс".

Пример:

$\forall_{ABCDEKabc}$(прямкоорд(K) & $K = (A, B, C)$ & коорд(D, K) = (a, b) & коорд(E, K) = $(c, 0)$ & $0 < c$ & разныеточки($A < D$) $\rightarrow a < 0 \leftrightarrow$ $\pi/2 < \angle(DAE)$)

Прием применяется к подутверждению условия задачи на преобразование, имеющей цель "класс". Никакая свободная переменная x выражений A, D, E не выделена комментарием (вспомпараметр x). Выражение a имеет свободную переменную, выделенную таким комментарием.

Спецификация приема имеет вид "тип(параллелпрямые)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "условие", "тип(преобразовать)", "цель(класс)", "не(контекст(список(x4 A D E) входит(x5 параметры(x4)) не(коммент(вспомпараметр переменная(x5)))))", "контекст(входит(x5 параметры(a)) не(коммент(вспомпараметр переменная(x5))))", "не(равно(A D))". Этого достаточно.

Число приемов данного типа - 3.

Ввод вспомогательного обозначения

1. Ввод обозначения для функции, рассматриваемой в условии задачи на описание.

   Пример:

   $\forall_{abcfuv}(\mathrm{Min}(\lambda_x(u(x), v(x)), a, b, c) \leftrightarrow \mathrm{Min}(f, a, b, c))$

   Прием применяется к условию задачи на описание. Описатель "отображение" не содержит неизвестных. Для определяемой им функции вводится новая переменная f, причем добавляется посылка $f = \lambda_x(u(x), v(x))$. Дальнейшие ссылки на функцию при исследовании ее свойств будут происходить через данную переменную.

   Спецификация приема имеет вид "тип(тринадцать)", "направл(N)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "корень", "известно(фикс(0 1 1))". Вводится также указатели "посылка($\lambda_x(u(x), v(x)) = f$, примечание(упростить) примечание(ориентацияравенства) примечание(определениепараметра))", "обозначения(f)", "новаяпеременная(f)". Этого почти достаточно.

   Число приемов данного типа - 4.

14.2.4 Усмотрение истинности либо ложности

Все приемы этого раздела имеют заголовок "второйтерм". За исключением особо оговариваемых случаев, они выполняют замену некоторого утверждения на константу "истина" либо "ложь".

Непосредственное усмотрение истинности либо ложности

Пример:

$\forall_n(n - \text{целое} \rightarrow n|n)$

Прием применяется без ограничений.

Спецификация приема имеет вид "тип(элементызадачи)". Справочник "заголовокприема" указывает уровень срабатывания 0. Этого достаточно.

Число приемов данного типа - 106.

Усмотрение истинности либо ложности с помощью проверочного оператора

1. Усмотрение истинности либо ложности с помощью проверочного оператора.

   Примеры:

   $\forall_{ab}(0 \leq a \rightarrow 0 \leq a^b)$

   $\forall_a(a - \text{set} \rightarrow a - \text{set})$

   Антецеденты обрабатываются проверочными операторами.

   Спецификация приема имеет вид "тип(блокпрограммы)". Справочник "заголовокприема" указывает уровень срабатывания 1. Во втором приеме он также вводит фильтр "или(не(корень) не(переменная(a)) и(тип(описать) неизвестная(a)))". Этого достаточно.

   Число приемов данного типа - 82.

2. Усмотрение истинности либо ложности подутверждения условия с помощью проверочного оператора.

   Этот тип приема введен, чтобы предотвратить попытки устранения полезных посылок задачи, являющихся очевидными следствиями других посылок.

   (a) Усмотрение истинности либо ложности подутверждения условия с помощью проверочного оператора.

   Пример:

   $\forall_a(\neg(a = 0) \rightarrow \neg(a = 0))$

   Прием применяется к подутверждению условия, не используемого для сопровождения по о.д.з. Наличие отрицания перед равенством обязательно. Антецедент обрабатывается проверочным оператором.

   Спецификация приема имеет вид "тип(простаяимпликация)". Справочник "заголовокприема" указывает уровни срабаывания 2 и 4. Он вводит фильтры "условие", "альтернатива(и(тип(описать) не(известно(корень))) уровень(4) уровень(2))", "не(сопровождение)". Требуется доработка.

   Число приемов данного типа - 6.

(b) Усмотрение истинности либо ложности подутверждения условия задачи на описание с помощью проверочного оператора.

Обычно приемы данного типа применяются при наличии дополнительных ограничений - при специальных целях задачи либо специальном виде условия. Пример:

$\forall_{ab}(a < b \rightarrow a < b)$

Прием применяется к подутверждению дизъюнктивного условия задачи на описание, не связанному внешними кванторами и описателями. Антецедент обрабатывается проверочным оператором.

Спецификация приема имеет вид "тип(то)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "тип(описать)", "условие". Он не прорабатывался.

Число приемов данного типа - 18.

(c) Усмотрение истинности либо ложности подутверждения условия задачи на доказательство с помощью проверочного оператора.

В данном контексте активность приемов усмотрения истинности подутверждений усиливается. Примером может служить теорема приема из предыдущего пункта. В этой версии прием применяется к подутверждению условия задачи на доказательство, не связанному внешними кванторами и описателями. Других ограничений нет.

Спецификация приема имеет вид "тип(префиксныйфильтр)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(доказать)", "условие", "свобоперанд(теквхожд)". Этого достаточно.

Число приемов данного типа - 15.

(d) Усмотрение истинности либо ложности условия задачи на доказательство с помощью усиленного проверочного оператора.

Примером может служить теорема приема из предыдущего пункта. Однако, прием применяется не к подутверждению условия, а к самому условию. При этом антецедент выделен не указателем "блокпроверок", а указателем "проверка" и обрабатывается усиленным проверочным оператором "провменьшеилиравно", содержащим целый ряд специальных группировок для усмотрения неравенств многочленного типа.

Спецификация приема имеет вид "тип(чисткапрограммы)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(доказать)", "условие", "корень", "менее(4 максимальныйуровень)". Вводится также указатель "проверка(1)". Этого почти достаточно.

Число приемов данного типа - 3.

3. Усмотрение истинности с помощью проверочного оператора, обрабатывающего вспомогательное константное утверждение.

Пример:

$\forall_{abc}(0 \le 2c - \pi|b| \to 0 < b\,\mathrm{arctg}\,a + c)$

Выражения b, c не содержат переменных. Антецедент обрабатывается проверочным оператором.

Спецификация приема имеет вид "тип(вычисление)", "блокпроверок(1)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "константа(b)", "константа(c)". Этого почти достаточно.

Число приемов данного типа - 5.

Усмотрение истинности либо ложности из утверждений, содержащихся в контексте

1. Усмотрение истинности либо ложности из утверждений, содержащихся в контексте.

   Пример:

   $\forall_f(\text{последовательность}(f, \mathbb{R})\ \&\ \text{сходится}(f) \to \text{огрсверху}(\text{Val}(f)))$

   Второй антецедент идентифицируется с утверждением из контекста, первый - обрабатывается проверочным оператором.

   Спецификация приема имеет вид "тип(обл)". Справочник "заголовокприема" указывает уровни срабатывания 1 и 3; вводится фильтр "или(не(корень) не(посылка))". Требуется доработка.

   Число приемов данного типа - 75.

2. Усмотрение истинности либо ложности подутверждения условия задачи на описание с помощью утверждения из контекста.

   (a) Усмотрение истинности либо ложности подутверждения условия задачи на описание с помощью утверждения из контекста.

   Пример:

   $\forall_{abc}(c \le a\ \&\ a - b < 0 \to \neg(b \le c))$

   Прием применяется к подутверждению условия задачи на описание. Первый антецедент идентифицируется с утверждением из контекста, второй - обрабатывается проверочным оператором. Либо анализируемое неравенство само является условием, причем c содержит неизвестные, либо задача имеет цель "редакция", неравенство не содержит неизвестных, выражение c неконстантное, а выражения a, b константные.

   Спецификация приема имеет вид "тип(двоичное)". Справочник "заголовокприема" ограничивается указанием уровней срабатывания 1,3,6 и вводом фильтров "тип(описать)", "условие". Он не прорабатывался.

   Число приемов данного типа - 13.

(b) Усмотрение истинности условия задачи на описание с помощью утверждения из контекста.

Пример: $\forall_{ABCD}$(биссектриса$(ABCD)$ & разныепрямые(прямая(AB), прямая$(BC)) \rightarrow \neg(C = D)$)

Прием применяется к условию задачи на описание, имеющему вид отрицания равенства. Первый антецедент идентифицируется с утверждением из контекста, второй - обрабатывается проверочным оператором.

Спецификация приема имеет вид "тип(комплексныечисла)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "тип(описать)", "условие", "отр", "не(равно(терм(прямая(AB)) терм(прямая(BC))))". Требуется доработка.

Число приемов данного типа - 12.

3. Усмотрение истинности подутверждения условия либо собственного подутверждения неэлементарной посылки с помощью утверждения из контекста.

Пример:

$\forall_{abc}(b \in a \ \& \ \neg(c \in a) \rightarrow \neg(b - c = 0))$

Прием применяется к подутверждению условия либо к подутверждению посылки, не являющейся элементарным утверждением. Первый антецедент идентифицируется с утверждением из контекста, второй - обрабатывается проверочным оператором.

Спецификация приема имеет вид "тип(подстзамена)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтр "или(условие не(отрицание))". Этого почти достаточно.

Число приемов данного типа - 5.

Усмотрение истинности либо ложности подутверждения условия с помощью идентифицирующих операторов

Пример:

$\forall_{ABCDE}(D \in$ отрезок(AE) & $E \in$ отрезок$(BC) \rightarrow D \in$ фигура$(ABC))$

Прием применяется к подутверждению условия. Антецеденты обрабатываются идентифицирующими операторами.

Спецификация приема имеет вид "тип(Норм)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтр "условие". Этого достаточно.

Число приемов данного типа - 3.

Усмотрение истинности либо ложности с помощью непосредственных вычислений

1. Усмотрение истинности либо ложности подутверждения условия с помощью непосредственных вычислений.

   Пример:

   $\forall_{abcdmn}(a = mc \ \& \ d = \text{нод}(m, n) \ \& \ \neg(d|b) \rightarrow \neg(a(\text{mod}n) = b))$

   Прием применяется к подутверждению условия. Выражение a представляет собой сумму. Первый антецедент выделен указателем "идентификатор". Его левая часть обрабатывается нормализатором "факторизация". Переменные b, m идентифицируются с целочисленными константами, переменная n - с натуральной константой. Второй и третий антецеденты выделены указателем "программа". Они реализуют непосредственные вычисления. Константа d отлична от единицы.

   Спецификация приема имеет вид "тип(пересечение)". Кроме того, для данного примера в ней присутствуют элементы "указатель(программа(2 3))", "типданных(целое b m)", "типданных(натуральное n)".

   Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "условие", "целое(b)", "целое(m)", "натуральное(n)". Создается также указатель "программа(2 3)". Требуется доработка.

   Число приемов данного типа - 4.

Усмотрение истинности или ложности подутверждения условия с помощью нормализатора вычисления

Пример:

$\forall_{abdf}(\lim_{n\rightarrow\infty}|f(n+1)/f(n)| = a \ \& \ a - 1 < 0 \rightarrow \text{сходится}(\lambda_m(\sum_{n=b}^{m} f(n), m - \text{натуральное})))$

Прием применяется к подутверждению условия. Первый антецедент, выделенный указателем "идентификатор", обращается к нормализатору "нормпредел" для вычисления предела. Второй антецедент обрабатывается проверочным оператором. Прием имеет ряд дополнительных фильтров, направленных на отсечение тех случаев, когда вычисление предела маловероятно.

Спецификация приема имеет вид "тип(унитерм)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтр "условие". Он не прорабатывался.

Число приемов данного типа - 10.

Усмотрение истинности либо ложности с помощью пакетных синтезаторов

1. Усмотрение истинности условия задачи на доказательство с помощью пакетного синтезатора.

   Пример:

$\forall_{abc}(a - b \le c \ \& \ c \le 0 \to a \le b)$

Прием применяется к условию задачи на доказательство. Первый антецедент обрабатывается синтезатором "верхняяоценка". Проверяется, что результат c константный. Затем второй антецедент обрабатывается проверочным оператором.

Спецификация приема имеет вид "тип(сжатиефильтра)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "условие", "тип(доказать)", "корень". Требуется доработка.

Число приемов данного типа - 3.

2. Усмотрение истинности либо ложности условия задачи на описание с помощью пакетного синтезатора.

Пример:

$\forall_{abcdepx}(\sqrt{a^2 + b^2} \le p \ \& \ 0 < d - p \to \neg(a\cos x + b\sin x = d))$

Прием применяется к уравнению задачи на описание либо к отрицанию уравнения, не используемому для сопровождаения по о.д.з. Хотя бы одно из выражений a, b неконстантное. Выражение d константное. Первый антецедент обрабатывается синтезатором "верхняяоценка", второй - проверочным оператором.

Спецификация приема имеет вид "тип(разность)", "указатель(значения(1))". Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "условие", "тип(описать)", "отрицание", "не(цель(редакция))", "не(сопровождение)", "или(не(константа(a)) не(константа(b)))", "не(равно($d\,p$))". Этого почти достаточно.

Число приемов данного типа - 7.

Усмотрение истинности либо ложности условия с помощью вспомогательных задач на доказательство

1. Усмотрение истинности условия с помощью вспомогательных задач на доказательство.

Пример:

$\forall_{fghp}(f(i) = g(i)h(i)/p(i) \ \&$ сходится$(\lambda_n(\sum_{i=1}^{n} |g(i)/p(i)|, n -$ натуральное$)) \to$ сходится$(\lambda_n(\sum_{i=1}^{n} f(i), n -$ натуральное$)))$

Прием применяется к условию задачи на доказательство. К непустому произведению $h(i)$ относятся все сомножители, имеющие либо вид неотрицательных степеней синуса или косинуса, либо вид степени минус единицы. Истинность второго антецедента усматривается с помощью вспомогательной задачи на доказательство. Уровень обращения к ней равен 5; прием имеет ограничитель трудоемкости.

Спецификация приема имеет вид "тип(объединениефрагментов)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "корень". Он не прорабатывался.

Число приемов данного типа - 8.

2. Усмотрение истинности условия задачи на доказательство с помощью вспомогательных задач на доказательство.

   (a) Усмотрение истинности условия задачи на доказательство с помощью вспомогательных задач на доказательство.

   Пример:

   $\forall_{FGabcdfg}(\forall_y(y \in [a,b] \to 0 \le F(y))$ & Производная(f,g) & $g = \lambda_y(F(y),G(y))$ $\to$ неубывает$(f,[a,b]))$

   Прием применяется к условию задачи на доказательство. Последние два антецедента идентифицируются с посылками, заблаговременно созданными для исследования функции с помощью производной. Первый антецедент обрабатывается вспомогательной задачей на доказательство.

   Спецификация приема имеет вид "тип(смоператор)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1 и вводом фильтров "тип(доказать)", "условие", "корень". Требуется доработка. Число приемов данного типа - 22.

   (b) Решение вспомогательной задачи для доказательства шага индукции.

   Пример:

   $\forall_{abcf}(0 \le \sum_{n=1}^{c} f(n) + a$ & $0 \le f(c+1) + b - a \to 0 \le \sum_{n=1}^{c+1} f(n) + b)$

   Прием применяется к условию задачи на доказательство, имеющей комментарий (натуральное k). Такой комментарий указывает, что происходит доказательство индукцией по параметру k, запущенное некоторым другим приемом. Первый антецедент идентифицируется с посылкой - индуктивным предположением. Истинность второго антецедента устанавливается при помощи задачи на доказательство.

   Спецификация приема имеет вид "тип(плюсбеск)", "указатель(доказать(N))". Второй элемент выделяет антецедент, истинность которого устанавливается задачей на доказательство. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(доказать)", "условие", "корень". Он не прорабатывался.

   Число приемов данного типа - 4.

3. Усмотрение истинности либо ложности неизвестного условия задачи на описание с помощью вспомогательной задачи на доказательство.

   Пример:

   $\forall_{ab}(0 < a - b \to \neg(a = b))$

   Прием применяется к содержащему неизвестные равенству - условию задачи на описание, не имеющей цели "редакция". Это условие не является равенством неизвестной известному выражению. Усматривается, что равенство - числовое. Антецедент обрабатывается задачей на доказательство. Уровень срабатывания

достаточно высок - попытка предпринимается лишь при отсутствии более естественных средств.

Спецификация приема имеет вид "тип(стрелкапирса)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 8 и вводом фильтров "условие", "тип(описать)", "корень", "не(известно(корень))", "не(цель(редакция))", "не(равно(a b))". Он не прорабатывался.

Число приемов данного типа - 12.

Усмотрение истинности либо ложности с помощью вспомогательной задачи на преобразование

1. Усмотрение истинности либо ложности подутверждения условия с помощью вспомогательной задачи на преобразование.

   Пример:

   $\forall_{abcdefghmnpqruv}(f = \lambda_{xy}(g(x,y), h(x,y))\ \&\ g(p+x, q+ax) = b(a)x^n/c(a) + d(a)x^m/e(a) + r(x)\ \&\ \exists_i(b(i) = 0\ \&\ \neg(d(i) = 0)) \to \neg(\mathrm{Extr}(f, (p,q), u, v)))$

   Прием применяется к подутверждению условия. Второй антецедент выделен указателем "идентификатор". Его левая часть обрабатывается задачей на преобразование, имеющей цель "формулатейлора x 0", а затем - нормализатором "Норммногочлен". В формуле Тейлора берутся первые шесть членов. Переменная m идентифицируется с нечетной натуральной константой, а переменная n - с четной, причем меньшей m. Остаточная сумма $r(x)$ не имеет одночленов от x, степень которых не превосходит m. В качестве a берется вспомогательная новая переменная. Последний антецедент обрабатывается задачей на доказательство.

   Спецификация приема имеет вид "тип(выводусловия)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 2 и вводом фильтра "условие". Он не прорабатывался.

   Число приемов данного типа - 7.

Использование нормализатора утверждений для усмотрения избыточного условия на неизвестную в ответе задачи на описание

Пример:

$\forall_{abx}((0 < a - b) = \text{истина}\ \&\ a \le x \to \neg(x = b))$

Прием применяется к условию задачи на описание, имеющему вид отрицания равенства. Задача имеет цель "редакция". Переменная x - неизвестная; выражения a, b не содержат неизвестных. Второй антецедент идентифицируется с другим условием. Первый антецедент выделен указателем "идентификатор". Его левая часть обрабатывается нормализатором "стандменьше", предпринимающим попытку упрощения строгих неравенств для известных параметров.

Спецификация приема имеет вид "тип(круг)", "неизвестная(x)", "оператор(стандменьше)", "антецедент(2)". Справочник "заголовокприема"указывает уровень срабатывания 2 и вводит фильтры "условие", "тип(описать)", "цель(редакция)", "отр", "неизвестная(x)", "известно(a)", "известно(b)". Требуется доработка.

Число приемов данного типа - 4.

14.2.5 Вывод в посылках

Приемы данного раздела имеют заголовок "вывод".

Вывод одного отношения

1. Вывод одноместного отношения.

   (a) Вывод одноместного отношения.

   Пример:

   $\forall_{ABf}$(изоморфизм$(f, A, B) \rightarrow$ взаимнооднозначно(f))

   Антецедент идентифицируется с посылкой.

   Спецификация приема имеет вид "тип(антецедент)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтр "посылка". Этого почти достаточно.

   Число приемов данного типа - 23.

2. Частные случаи вывода равенств для числовых атомов.

   Этот и два следующих раздела занимают особое место в логическом ассемблере. При обучении решателя планиметрическим задачам оказалось, что необходима весьма тонкая калибровка степени мотивированности срабатывания приема, выводящего некоторое соотношение. При ослаблении фильтров приема происходило быстрое заполнение списка посылок задачи бесполезными фактами, при усилении - отсекались необходимые для решения задачи срабатывания. Для преодоления этого явления возможны два пути. Первый - использование древовидной системы типов и подтипов приемов вывода, характеризующихся различными степенями мотивированности срабатывания. Для одной и той же теоремы создавались несколько различных приемов вывода, отличающихся уровнем срабатывания и силой фильтров. На малых уровнях размещались приемы с сильной мотивацией срабатывания, на больших - с ослабленной. При таком подходе пришлось ввести в логический ассемблер очень большое количество типов приемов вывода равенств с числовыми атомами. Выбор конкретной версии приема должен осуществляться генератором приемов в процессе примерки на задачнике. Второй путь - использование пакетных анализаторов в качестве усилителя. В рамках пакетного анализатора количество приемов, выводящих следствия, существенно меньше, чем в основной базе приемов. Это позволяет ослабить ограничения на их срабатывания и выполнять вывод на значительно большую глубину. В решаемую задачу будут передаваться лишь те следствия, которые окажутся наиболее интересными. Необходима разумная кластеризация приемов вывода и создание для этих кластеров множества различных пакетных анализаторов. Данный подход неплохо зарекомендовал себя в планиметрии. Однако, он замедляет решение задачи в стандартных ситуациях, и применяется лишь на повторном цикле решения - если обычных методов оказалось недостаточно. По-видимому, целесообразно использовать оба подхода.

(a) Равенство двух невырожденных числовых атомов.

i. Равенство двух невырожденных числовых атомов.

Пример:

$\forall_{ABCDEF}$(биссектриса$(ABCD)$ & $F \in$ прямая(BD) & прямая$(EF) \parallel$ прямая(BC) & $E \in$ прямая$(AB) \to l(BE) = l(EF)$)

Прием применяется в посылках задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(редакторответа)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(прямая(BD)))", "не(равно$(B\ D)$)", "усм(актив(прямая(EF)))", "не(равно$(E\ F)$)", "усм(актив(прямая(BC)))", "не(равно$(B\ C)$)", "усм(актив(прямая(AB)))", "не(равно$(A\ B)$)", "не(равно(терм(прямая(EF))) терм(прямая(BC))))", "не(равно$(B\ E)$)". Этого почти достаточно.

Число приемов данного типа - 48.

ii. Равенство двух невырожденных числовых атомов, один из которых - старый.

A. Равенство двух невырожденных числовых атомов, хотя бы один из которых - старый.

Пример:

$\forall_{ABCD}$(параллелограмм$(ABCD) \to l(AB) = l(CD)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Хотя бы один из атомов $l(AB)$, $l(CD)$ уже встречается в задаче.

Спецификация приема имеет вид "тип(факторизация)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "или(усм(актив(расстояние(AB))) усм(актив(расстояние(CD))))", "не(равно$(A\ B)$)", "не(равно$(C\ D)$)". Этого почти достаточно.

Число приемов данного типа - 6.

B. Равенство невырожденного числового атома заданному старому невырожденному атому.

Пример:

$\forall_{ABC}$(актив$(l(AB))$ & прямая$(AB) \perp$ прямая(BC) & $\angle(BAC) = \pi/4 \to l(AB) = l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Первый антецедент указывает на то, что атом $l(AB)$ уже встречается в задаче.

Спецификация приема имеет вид "тип(нормализация)", "терм(t)". Здесь t - числовой атом, который должен уже иметься в задаче. В нашем примере такое указание избыточно, однако при отсутствии соответствующего антецедента был бы создан фильтр "актив$(l(AB))$". Справочник "заголовокприема" указывает уровень

срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(прямая(AB)))", "не(равно($A\ B$))", "усм(актив(прямая($B\ C$)))", "не(равно($B\ C$))". Требуется доработка.

Число приемов данного типа - 24.

C. Равенство числового атома заданному атому "неизв".

Числовой атом относится к типу "неизв" для задачи на исследование, если он либо является неизвестной внешней задачи на описание, либо входит в уравнение с такой неизвестной. В случае задачи на доказательство вместо неизвестной внешней задачи на описание рассматривается неизвестная самой задачи на доказательство. Пример:

$\forall_{ABCD}$(актив($\angle(CAD)$) & актив($\angle(BAC)$) & $\angle(CAD) = 2\angle(BAC)$ & разныестороны($C, D,$ прямая(AB)) $\rightarrow \angle(BAD) = \angle(BAC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Угол BAC имеет тип "неизв".

Спецификация приема имеет вид "тип(арность)", "терм(t)", где t - числовой атом типа "неизв". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "неизв(терм(угол(BAC)))", "не(равно(терм(угол(BAD) терм(угол(BAC)))))", "усм(актив(прямая(AB)))", "усм(актив(прямая(AC)))", "не(усм(принадлежит(D прямая(AC))))", "не(усм(принадлежит(C прямая(AB))))". Этого почти достаточно.

Число приемов данного типа - 5.

D. Равенство числового атома заданному известному числовому атому.

Пример:

$\forall_{BDEF}$(актив($\angle(EFD)$) & прямая(BE) $\parallel$ прямая(FD) & актив($l(BE)$) & разныестороны($B, D,$ прямая(EF)) & разныепрямые(прямая(BE), прямая(FD)) $\rightarrow \angle(EFD) = \angle(BEF)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Угол EFD известен, угол BEF - не известен.

Спецификация приема имеет вид "тип(Делитель)", "терм(t)", где t - известный числовой атом. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(известно(терм(угол(EFD))))", "конец(не(известно(терм(угол(BEF)))))", "усм(актив(прямая(EF)))", "усм(актив(прямая(DF)))", "не(усм(принадлежит(D прямая(EF))))", "не(равно(терм(прямая(BE)) терм(прямая(DF))))". Этого почти достаточно.

Число приемов данного типа - 7.

E. Равенство двух старых числовых атомов.

Пример:

$\forall_{ABCDE}$(актив($\angle(DAE)$) & актив($\angle(BAC)$) & $A \in$ отрезок(BE) & $A \in$ отрезок(CD) $\rightarrow \angle(BAC) = \angle(DAE)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Как видно из теоремы приема, углы BAC, DAE уже рассматриваются в задаче.

Спецификация приема имеет вид "тип(подборнеизвестных)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(прямая(AB)))", "усм(актив(прямая(AC)))", "не(равно(B E))", "не(равно(C D))", "не(усм(принадлежит(E прямая(AD))))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(A B))", "не(равно(A C))". Этого почти достаточно.

Число приемов данного типа - 21.

F. Равенство заданного старого числового атома атому, косвенно связанному с выделенными в задаче объектами.

Пример:

$\forall_{ABCD}$(прямая(AB) $\perp$ прямая(BC) & $l(AD) = l(DC)$ & $D \in$ прямая(AC) $\rightarrow l(BD) = l(DC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Так как второй антецедент выделен указателем "усм", атом $l(DC)$ уже имеется в посылках задачи. Прямая BD тоже рассматривается в задаче. Таким образом, имеется косвенное указание на рассмотрение атома $l(BD)$.

Спецификация приема имеет вид "тип(точкапривязки)", "терм(t)", где t - старый числовой атом. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(прямая(B D)))", "усм(актив(прямая(A B)))", "не(равно(A B))", "усм(актив(прямая(B C)))", "не(равно(B C))", "усм(актив(прямая(A C)))", "не(равно(A C))", "не(равно(B D))", "не(равно(D C))". Этого почти достаточно.

Число приемов данного типа - 5.

iii. Равенство числовых атомов, косвенно связанных с выделенными в задаче объектами.

A. Равенство числовых атомов, косвенно связанных с выделенными в задаче объектами.

Пример:

$\forall_{ABCD}$(квадрат($ABCD$) $\rightarrow l(AB) = l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Прямые AB и BC уже введены в рассмотрение.

Спецификация приема имеет вид "тип(подстановка)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))",

"усм(актив(прямая(AB)))", "усм(актив(прямая(BC)))", "не(равно($A\ B$))", "не(равно($B\ C$))". Этого достаточно.

Число приемов данного типа - 7.

iv. Вывод существенного равенства числовых атомов.

Пример:

$\forall_{ABCDEF}(l(AB) = l(DE)\ \&\ C \in \text{отрезок}(AB)\ \&\ F \in \text{отрезок}(DE)\ \&\ l(AC) = l(DF) \to l(BC) = l(EF))$

Прием применяется в посылках задачи на доказательство либо на исследование. Пакетный индикатор "существравно" усматривает, что выводимое равенство представляет интерес.

Спецификация приема имеет вид "тип(контрольвывода)". Справочник "заголовокприема" указывает уровень срабатывания 9 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(легковидеть(существравно($l(BC) = l(EF)$)))", "не(равно($A\ B$))", "не(равно($D\ E$))", "не(равно($A\ C$))", "не(равно($B\ C$))", "не(равно($D\ F$))", "не(равно($E\ F$))". Этого достаточно.

Число приемов данного типа - 5.

(b) Соотношение пропорциональности для двух невырожденных числовых атомов.

i. Соотношение пропорциональности для двух числовых атомов.

Пример:

$\forall_{ABCDEFabpq}(D \in \text{отрезок}(AB)\ \&\ bl(AD) = al(BD)\ \&\ E \in \text{отрезок}(AC)\ \&\ F \in \text{прямая}(CD)\ \&\ F \in \text{прямая}(BE)\ \&\ pl(BF) = ql(EF)\ \&\ \text{разныепрямые}(\text{прямая}(AB), \text{прямая}(AC))\ \&\ \text{разныеточки}(A, C) \to (aq - bp)l(CE) = bpl(AE))$

Прием применяется в посылках задачи на доказательство либо на исследование. Соотношения пропорциональности в антецедентах усматриваются пакетным синтезатором "пропорциональны". Расстояния $l(AD)$ и $l(BD)$ уже рассматриваются в задаче.

Спецификация приема имеет вид "тип(оценка)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(известно(результат))", "конец(или(не(известно(a)) не(известно(q)) не(известно(p))не(известно(b)) не(контекст(значения(равно(умножение(c расстояние(CE)) умножение(d расстояние(AE))))известно(c) известно(d)))))", "не(равно($A\ C$))", "усм(актив(прямая(CD)))", "не(равно($C\ D$))", "усм(актив(прямая(BE)))", "не(равно($B\ E$))", "усм(актив(расстояние(AD)))", "усм(актив(расстояние(BD)))", "усм(актив(расстояние(BF)))", "усм(актив(расстояние(EF)))", "не(равно($A\ D$))", "не(равно($B\ D$))", "не(равно($A\ E$))", "не(равно($C\ E$))", "не(равно(терм(прямая(AB)) терм(прямая(AC))))". Требуется доработка.

Число приемов данного типа - 8.

ii. Соотношение пропорциональности с известными коэффициентами связывает два числовых атома.

A. Соотношение пропорциональности с известными коэффициентами связывает два числовых атома.

Пример:

$\forall_{ABCDEF}$(окружность(AB)вписана в фигура($FCDE$) & квадрат($FCDE$) $\rightarrow 2l(AE) = \sqrt{2}l(CD)$)

Прием применяется в посылках задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(кортеж)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(известно(результат))", "не(равно(A E))", "не(равно(C D))". Этого почти достаточно.

Число приемов данного типа - 4.

B. Вывод из соотношения пропорциональности для двух неизвестных числовых атомов, имеющего известные коэффициенты, другого соотношения пропорциональности с известными коэффициентами.

Пример:

$\forall_{ABCDab}(\angle(ABD) = \angle(DBC)$ & $D \in$ прямая(AC) & $al(AB) = bl(BC)$ & разныеточки(A, C) & разныепрямые(прямая(AB), прямая(BC)) $\rightarrow bl(CD) = al(AD)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Выражения a, b не содержат неизвестных, а результаты обработки нормализаторами общей стандартизации числовых атомов $l(AB)$, $l(BC)$ - содержат.

Спецификация приема имеет вид "тип(оглавление)", "пропорция(n)", где n - номер антецедента, представляющего собой соотношение пропорциональности, обрабатываемое пакетным синтезатором "пропорциональны". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(известно(результат))", "известно(b)", "известно(a)", "проверка(не(известно(терм(расстояние(AB))))3)", "проверка(не(известно(терм(расстояние(BC))))3)", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(расстояние(AB)))", "усм(актив(расстояние(BC)))", "не(усм(принадлежит(D прямая(AB))))", "не(усм(принадлежит(C прямая(BA))))", "не(усм(принадлежит(C прямая(BD))))", "не(равно(C D))", "не(равно(A D))", "не(равно(терм(прямая(AB)) терм(прямая(BC))))". Этого почти достаточно.

Число приемов данного типа - 3.

iii. Соотношение пропорциональности связывает заданный числовой атом со старым.

A. Соотношение пропорциональности для двух старых числовых атомов.

Пример:

$\forall_{ABCDEab}$(актив($l(BD)$) & актив($l(AB)$) & $B \in$ отрезок(AD) & прямая(DE) $\parallel$ прямая(BC) & $C \in$ прямая(AE) & $al(AB) = bl(AC)$ & разныепрямые(прямая(AD), прямая(AE)) $\rightarrow al(BD) = bl(CE)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Числовые атомы $l(BD)$, $l(CE)$ уже рассматриваются в задаче.

Спецификация приема имеет вид "тип(преобразователь)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "альтернатива(тип(доказать) не(константа(результат)) не(известно(результат)))", "усм(актив(расстояние(CE)))", "не(равно(A D))", "усм(актив(прямая(DE)))", "не(равно(D E))", "усм(актив(прямая(BC)))", "не(равно(B C))", "усм(актив(прямая(AE)))", "не(равно(A E))", "не(равно(A B))", "не(равно(B D))", "не(равно(терм(прямая(DE)) терм(прямая(BC))))", "не(равно(C E))", "не(равно(терм(прямая(AD)) терм(прямая(AE))))". Этого почти достаточно.

Число приемов данного типа - 16.

(c) Вывод явного выражения для значения невырожденного числового атома.

i. Вывод явного выражения для значения числового атома.

Пример:

$\forall_{ABCDEab}(D \in$ окружность(AB) & $E \in$ окружность(AB) & $D \in$ отрезок(CE) & $bl(DE) = cl(CD)$ & актив($\angle(ECA)$) & $\angle(ECA) = a$ & актив(окружность(AB)) $\rightarrow \angle(DAE) = \pi - 2\,\mathrm{arctg}((2b/c + 1)\,\mathrm{tg}\,a)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Выражения a, b, c не содержат неизвестных, числовой атом $\angle(DAE)$ - содержит.

Спецификация приема имеет вид "тип(частичныйответ)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "известно(b)", "известно(c)", "известно(a)", "не(известно(терм(угол(DAE))))", "не(равно(D E))", "не(равно(C E))", "не(равно(C D))", "не(усм(принадлежит(A прямая(CE))))". Этого почти достаточно.

Число приемов данного типа - 8.

ii. Вывод явного выражения для значения текущего числового атома.

Пример:

$\forall_{abcdmn}(d = a + b$ & уголмежду$(a, b) = c$ & длина$(a) = m$ & длина$(b) = n \rightarrow$ длина$(d) = \sqrt{m^2 + n^2 + 2mn\cos c}$)

В первом антецеденте рассматривается векторная сумма. Прием применяется в посылках задачи на доказательство либо на исследование. Попытка его применения инициируется усмотрением подвыражения "длина(d)". Выражения c, m, n известны, длина вектора d - не известна.

Спецификация приема имеет вид "тип(номероперанда)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "известно(m)",

"известно(n)", "известно(c)", "не(известно(терм(длина(d))))". Создается также указатель "контрольвывода(длина(d))". Этого достаточно.

Число приемов данного типа - 11.

iii. Вывод явного выражения для значения невырожденного числового атома "неизв".

Пример:

$\forall_{ABCpqr}$(прямая(AC) $\perp$ прямая(BC) & актив($\angle(BAC)$) & $l(AC) = m$ & $m = pr/su$ & $l(AB) = qr/tu \rightarrow \angle(BAC) = \arccos(pt/qs)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Третий антецедент идентифицируется с посылкой, четвертый и пятый - выделены указателем "идентификатор". Выражения p, q, s, t не содержат неизвестных. Числовой атом "$\angle(BAC)$" имеет тип "неизв". Выражение $l(AB)$ уже рассматривается в задаче.

Спецификация приема имеет вид "тип(выпуклавниз)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "известно(q)", "известно(t)", "известно(p)", "известно(s)", "неизв(терм(угол(BAC)))", "усм(актив(прямая(AC)))", "не(равно($A\ C$))", "усм(актив(прямая(BC)))", "не(равно($B\ C$))", "усм(актив(прямая(AB)))", "не(усм(принадлежит(C прямая(AB))))". Требуется доработка.

Число приемов данного типа - 5.

iv. Усмотрение константного значения числового атома.

A. Усмотрение константного значения числового атома.

Пример:

$\forall_{ABC}$(прямая(AB) $\perp$ прямая(BC) & $l(AC) = 2l(AB) \rightarrow \angle(ACB) = \pi/6$)

Прием применяется в посылках задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(возведениевстепень)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(равно(терм(угол(ACB))терм(дробь(пи 6))))", "усм(актив(прямая(AB)))", "не(равно($A\ B$))", "усм(актив(прямая(BC)))", "не(равно($B\ C$))", "усм(актив(расстояние(AC)))", "усм(актив(расстояние(AB)))". Требуется доработка.

Число приемов данного типа - 5.

B. Усмотрение константного значения текущего числового атома.

Пример:

$\forall_{abc}(a =$ вектумнож$(b, c) \rightarrow$ уголмежду$(a, b) = \pi/2)$

Прием применяется в посылках задачи на доказательство либо на исследование. Попытка его применения инициируется усмотрением подвыражения "уголмежду(a, b)". Антецедент идентифицируется с посылкой.

Спецификация приема имеет вид "тип(квазипеременная)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(равно(терм(уголмежду(вектумнож(b, c),c)) терм(дробь(пи 2))))". Создается также указатель "контрольвывода(уголмежду(a, b))". Этого достаточно.

Число приемов данного типа - 4.

C. Усмотрение константного значения старого числового атома.

Пример:

$\forall_{ABCD}$(квадрат($ABCD$) & актив($\angle(ABD)$) $\rightarrow \angle(ABD) = \pi/4$)

Прием применяется в посылках задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(тангенс)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(прямая(AB)))", "усм(актив(прямая(BD)))", "не(усм(принадлежит(D прямая(AB))))". Последние три фильтра избыточны. Тем не менее, прием будет вполне работоспособен.

Число приемов данного типа - 6.

(d) Выражение невырожденного числового атома через более простые числовые атомы.

i. Выражение текущего числового атома через более простые числовые атомы.

Пример:

$\forall_{ABCD}$(прямоугольник($ABCD$) $\rightarrow S$(фигура($ABCD$)) $= l(AB)l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Попытка его применения инициируется при усмотрении подвыражения "S(фигура($ABCD$))". Специальный комментарий блокирует повторные попытки выражения площади.

Спецификация приема имеет вид "тип(усмцелое)", "терм(t)", где t - текущий числовой атом. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "или(тип(доказать)тип(исследовать))", "не(равно(A B))", "не(равно(B C))". Требуется доработка.

Число приемов данного типа - 27.

ii. Выбор наилучшего выражения текущего числового атома через более простые атомы.

A. Выбор наилучшего выражения текущего числового атома через более простые атомы.

Иногда для "сложного" числового атома существует несколько различных выражений через более простые атомы. Например, для площади треугольника имются различные формулы - через длину

высоты и сторону, через длины двух сторон и угол между ними, через длины трех сторон и т.п. Чтобы выбрать наилучший для текущей ситуации способ, используется специальный механизм. Приемы снабжаются двумя уровнями срабатывания - предварительным и основным. На предварительном уровне вводится оценка целесообразности использования данного приема, сохраняемая в комментариях. На основном уровне прием убеждается, что его оценка не хуже оценок других версий, и лишь тогда срабатывает. Более приоритетной считается версия с меньшим значением оценки. В данном разделе собраны приемы именно таких типов. Начинаем с общего типа; выбор того или иного его подтипа осуществляется доводчиком генератора приемов в процессе примерки на задачах. Пример:

$\forall_{ABCD}$(прямая$(AB) \perp$ прямая(CD) & $D \in$ прямая$(AB) \rightarrow$
$2S$(фигура$(ABC)) = l(AB)l(CD)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Попытка его применения инициируется при усмотрении подвыражения "S(фигура(ABC))". Имеется два уровня срабатывания - 1 и 2. Указатель "оценка(1 треугольник фигура(ABC) 2)" определяет ввод оценки 2 на уровне 1. На уровне 2 происходит принятие решения о срабатывании, если другие приемы для площади треугольника не предложили оценку 1.

Спецификация приема имеет вид "тип(отборприемов)", "терм(t)", "оценка(...)". Здесь t - текущий числовой атом. Элемент "оценка(...)" непосредственно переносится в указатели. Справочник "заголовокприема" указывает уровни срабатывания 1,2 и вводит фильтры "или(тип(доказать)тип(исследовать))", "усм(актив(прямая(AB)))", "усм(актив(прямая(CD)))", "не(равно(C D))", "не(равно(A B))". Создаются также указатели "контрольвывода(S(фигура(ABC)))" и указатель "оценка(1 треугольник фигура(ABC) 2)". Требуется доработка.

Число приемов данного типа - 6.

(e) Вывод выражения невырожденного числового атома через численные параметры.

i. Вывод выражения текущего невырожденного числового атома через численные параметры.

Пример:

$\forall_{PXm}$(дисперсия$(X, P) = m \rightarrow$ среднквадроткл$(X, P) = \sqrt{m}$)

Прием применяется в задачах на исследование. Попытка его применения инициируется усмотрением подвыражения "среднквадроткл(X, P)". Антецедент идентифицируется с посылкой. Выражение m не содержит невырожденных числовых атомов.

Спецификация приема имеет вид "тип(обозначпеременных)", "терм(t)", где t - текущий атом. Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(исследовать)", "не(контекст(числовойатом(m x2)не(переменная(x2))))". Создается также ука-

затель "контрольвывода(среднквадроткл(X, P))". Этого почти достаточно.

Число приемов данного типа - 3.

(f) Вывод уравнения для численных параметров нечислового объекта.

Пример:

$\forall_{ABfg}$(плотнраспред$(A, B) = \lambda_t(f(t), g(t))$ & матожидание$(A, B) = a \rightarrow a = \int_{-\infty}^{\infty} tf(t)dt$)

Прием применяется в посылках задачи на доказательство либо на исследование. Оба антецедента идентифицируются с посылками. Хотя бы одно из выражений a, $f(t)$ содержит неизвестные. Ни одно из них не содержит невырожденных числовых атомов.

Спецификация приема имеет вид "тип(прогртерм)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "или(не(известно(a)) не(известно($f(t)$)))", "не(контекст(список(x2 a $f(t)$)) числовойатом(x2 x3)не(переменная(x3)))". Требуется доработка.

Число приемов данного типа - 5.

(g) Вывод стандартизирующего равенства, сводящего текущий невырожденный числовой атом к однотипному атому.

Пример:

$\forall_{abcdpqxy}([x + a, b] = p$ & $[b, x + c] = q \rightarrow$ длина$(q) = c - a -$ длина$(p))$

Прием применяется в задачах на доказательство либо на исследование. Попытка его применения инициируется усмотрением подвыражения "длина(q)". Выражение "длина(p)" уже встречается в посылках. Выражения a, c не имеют невырожденных числовых атомов. Заметим, что прием был создан для задач по физике, причем p, q - два последовательных временных промежутка.

Спецификация приема имеет вид "тип(доопределение)", "терм(t)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "или(тип(доказать) тип(исследовать))", "контекст(посылка(x5) вхождениетерма(x5 длина(p)))", "не(контекст(список(x5 a c) числовойатом(x5 x6)не(переменная(x6))))". Создается также указатель "контрольвывода(длина(q))". Этого почти достаточно.

Число приемов данного типа - 3.

(h) Вывод следствий из численных уравнений.

В данном подразделе рассматриваются приемы, выводящие следствия из уравнений безотносительно к наличию или отсутствию в них невырожденных числовых атомов.

i. Вывод следствия из численного уравнения, ориентированного на сближение с другим уравнением.

Пример:

$\forall_{abcde}(a\log_b c + d = e \to c^a b^d = b^e)$

Прием применяется в задачах на исследование. Выражение d содержит неизвестные. В посылках уже встречается выражение вида b^X, где X содержит неизвестные. Внутри выражения d отсутствует неизвестный логарифм, рсположенный внутри степенного выражения.

Спецификация приема имеет вид "тип(стмногочлена)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "тип(исследовать)", "корень", "конец(не(контекст(числовойатом(корень х6) не(переменная(х6)))))". Он не прорабатывался.

Число приемов данного типа - 5.

ii. Вывод следствия из численных уравнений, в котором неизвестные группируются под одной сложной операцией.

Пример:

$\forall_{abcdefghijk}(h = ba \;\&\; i = ea \;\&\; h\sin j\cos k + c = d \;\&\; i\cos j\sin k + f = g \to abe\sin(j-k) + ce - bf = de - bg)$

Прием применяется в задачах на исследование. Два последних антецедента идентифицируются с посылками, не содержащими невырожденных числовых атомов. Два первых антецедента выделены указателем "идентификатор". Выражения j, k содержат неизвестные.

Спецификация приема имеет вид "тип(упрощвариант)", "антецедент(3)", "антецедент(4)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(исследовать)", "не(контекст(числовойатом(фикс(3) х12) не(переменная(х12))))", "не(контекст(числовойатом(фикс(4) х12) не(переменная(х12))))". Требуется доработка.

Число приемов данного типа - 4

3. Общий случай вывода равенств для невырожденных числовых атомов (симметричные типы).

В этом и следующем подразделах глубина вложенности дерева подтипов настолько велика, что приходится использовать альтернативную запись иерархии подпунктов. Каждый подпункт будет обозначаться последовательностью номеров, отделенных точками. Отступы с левой стороны страницы везде будут одинаковыми.

Приводимый ниже список типов может показаться чрезвычайно избыточным. Возникает естественное желание сократить его хотя бы до одного-двух десятков. Какие-то типы вообще связаны с единственным приемом. Однако, надо понимать, каким образом данный список возникал. Новый тип приема создавался лишь в тех ситуациях, когда других способов объяснить, как полезное срабатывание выделяется на фоне множества бесполезных или даже ломающих ход решения, попросту не было. Во всяком случае, это сделало данный список хотя бы полезной информацией о контекстах принятия решений в потоке реальных задач. Можно представить себе программу автоматической доводки на примерах, которая использовала бы его как источник стандартных шаблонов, позволяющий возможно более точным образом выработать эвристические

решающие правила. Кроме того, речь здесь идет только о настройке режима решения задач "в лоб". Если развивать альтернативный режим, использующий небольшие локальные переборы для отбора наиболее ценного срабатывания, то число типов приемов можно было бы резко сократить. Впрочем, и решение задач при этом замедлилось бы. Целесообразно использовать оба режима, прибегая ко второму при неудаче первого. Преимущество первого подхода состоит в том, что "индивидуальные" для данного типа фильтры приема обычно располагаются в начале ЛОС-программы и резко усиливают отсечение ненужных попыток, существенно ускоряя работу решателя. При втором подходе эта возможность утрачивается.

В данном разделе собраны типы, у которых все входящие в теорему приема невырожденные числовые атомы рассматриваются в фильтрах симметричным образом - ни один из них специально не выделяется. В следующем разделе будут собраны типы, где один или несколько атомов играют в фильтрах особую роль.

3.1. Общий случай соотношения для числовых атомов.

Это - случай неограниченного использования теоремы приема. Остальные типы данного раздела - его подтипы. Нужный подтип определяется в процессе доводки приема на задачах. Как правило, возникают несколько разных версий приема, срабатывающих на разных уровнях. Пример приема данного типа:

$\forall_{ABCDE}(l(AB) = l(BC)$ & $l(AB) = l(AC)$ & окружность(DE) вписана в фигура $(ABC) \to 2\sqrt{3}l(DE) = l(AC))$

Прием применяется в посылках задачи на доказательство либо на исследование. Последний антецедент идентифицируется с посылкой, первые два - выделены указателем "идентификатор".

Спецификация приема имеет вид "тип(родобъекта)". Справочник "заголовок-приема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(равно(D E))", "не(равно(A C))". Этого почти достаточно.

Число приемов данного типа - 18.

3.2. Соотношение для старых числовых атомов.

3.2.1. Соотношение для старых числовых атомов.

Пример:

$\forall_{ABC}($актив$(l(AB))$ & $B \in$ отрезок$(AC) \to l(AC) = l(AB) + l(BC))$

Прием применяется в посылках задачи на доказательство либо на исследование. Числовые атомы $l(AB)$, $l(AC)$, $l(BC)$ уже рассматриваются в задаче. Выводимое равенство содержит неизвестные.

Спецификация приема имеет вид "тип(род)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(расстояние(AC)))", "усм(актив(расстояние(BC)))", "не(известно(результат))", "не(равно(A C))", "не(равно(A B))", "не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 71.

3.2.2. Соотношение связывает два старых атома через известные параметры.

Пример:

$\forall_{ABCDE}(B \in$ прямая(AD) & $E \in$ прямая(AC) & прямая$(AC) \perp$ прямая(DE) & прямая$(AD) \perp$ прямая$(BC) \rightarrow l(AE)l(BC) = l(AB)l(DE))$

Прием применяется в посылках задачи на доказательство либо на исследование. Числовые атомы $l(AE)$, $l(BC)$, $l(AB)$, $l(DE)$ уже рассматриваются в задаче. Какие-то два из них известны. Выводимое равенство содержит неизвестные.

Спецификация приема имеет вид "тип(планиметрия)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(известно(результат))", "усм(актив(расстояние(AE)))", "усм(актив(расстояние(BC)))", "усм(актив(расстояние(AB)))", "усм(актив(расстояние(DE)))", "конец(числатомы(набор(терм(расстояние(AE)) терм(расстояние(BC)) терм(расстояние(AB)) терм(расстояние(DE))) и(актив(2) известно(2))))", "усм(актив(прямая(AD)))", "не(равно(A D))", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(прямая(DE)))", "не(равно(D E))", "усм(актив(прямая(BC)))", "не(равно(B C))", "не(равно(A E))", "не(равно(A B))". Этого достаточно.

3.2.3. Связь старого атома с численными параметрами.

3.2.3.2. Соотношение связывает числовой атом "неизв" с численными параметрами.

Пример:

$\forall_{ABC}($прямая$(AB) \perp$ прямая(AC) & актив$(\angle(ABC)) \rightarrow l(AC) = \sin(\angle(ABC))l(BC))$

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояния $l(AC)$, $l(BC)$ уже рассматриваются в задаче. Хотя бы два из атомов $l(AC)$, $l(BC)$, $\angle(ABC)$ выразимы через численные параметры задачи (известные либо неизвестные). Оставшийся атом имеет тип "неизв".

Спецификация приема имеет вид "тип(обобщантецедент)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(расстояние(AC)))", "усм(актив(расстояние(BC)))", "конец(числатомы(набор(терм(расстояние(AC)) терм(угол(ABC)) терм(расстояние(BC))) Числопред(неизв)))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))",

"усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 7.

3.2.3.3. Определение старого числового атома.

3.2.3.3.1. Определение старого числового атома.

В качестве примера рассмотрим прием, теорема которого - та же, что в предыдущем пункте. Расстояния $l(AC)$, $l(BC)$ уже рассматриваются в задаче. Хотя бы два из атомов $l(AC)$, $l(BC)$, $\angle(ABC)$ уже известны.

Спецификация приема имеет вид "тип(стандменьшеилиравно)". Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(расстояние(AC)))", "усм(актив(расстояние(BC)))", "конец(числатомы(набор(терм(расстояние(AC)) терм(угол(ABC)) терм(расстояние(BC))) опред(актив)))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 18.

3.2.3.3.2. Определение числового атома "неизв".

Пример:

$\forall_{ABC}(l(AB) = l(BC)$ & актив($\angle(ABC)$) & разныеточки(A, C) &
актив($\angle(BAC)$) $\rightarrow 2\angle(BAC) + \angle(ABC) = \pi$)

Прием применяется в посылках задачи на доказательство либо на исследование. Один из атомов $\angle(BAC)$, $\angle(ABC)$ известен, другой - имеет тип "неизв".

Спецификация приема имеет вид "тип(нормусп)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм(угол(BAC)) терм(угол(ABC))) опред(неизв)))", "усм(актив(прямая(AB)))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(A C))". Этого достаточно.

Число приемов данного типа - 22.

3.2.4. Соотношение для старых числовых атомов, среди которых имеется атом "Неизв".

Напомним, что выражение t в задаче на исследование имеет тип "Неизв", если либо оно содержит неизвестную внешней задачи на описание, либо имеет общий невырожденный числовой атом с уравнением, содержащим неизвестную внешней задачи на описание, либо существует уравнение, включающее такие невырожденные числовые атомы, один из которых входит в t, а другой - в

уравнение, содержащее неизвестную внешней задачи на описание. Это - расширение типа "неизв".

3.2.4.1. Соотношение для старых числовых атомов, среди которых имеется атом "Неизв".

Пример:

$\forall_{ABCD}(\angle(ACB) = \angle(ADB)$ & актив$(\angle(ABC))$ & актив$(\angle(ABD))$ & актив$(l(AC))$ & актив$(l(AD)) \to l(AC)\sin(\angle(ABD)) = l(AD)\sin(\angle(ABC)))$

Прием применяется в посылках задачи на доказательство либо на исследование. Хотя бы один из числовых атомов $\angle(ABC)$, $\angle(ABD)$, $l(AC)$, $l(AD)$ имеет тип "Неизв".

Спецификация приема имеет вид "тип(вспомпараметр)". Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм($l(AC)$) терм($\angle(ABD)$) терм($l(AD)$) терм($\angle(ABC)$))) Неизв(1)))", "усм(актив(прямая(AB)))", "усм(актив(прямая(BC)))", "усм(актив(прямая(BD)))", "не(усм(принадлежит(C прямая(AB))))", "не(усм(принадлежит(D прямая(AB))))","не(равно(A C))", "не(равно(A D))". Этого почти достаточно.

Число приемов данного типа - 4.

3.2.4.2. Соотношение для старых числовых атомов, среди которых имеется атом "неизв".

3.2.4.2.1. Соотношение для старых числовых атомов, среди которых имеется атом "неизв".

Пример:

$\forall_{ABCD}$(трапеция($ABCD$) & актив$(\angle(BCD))$ & актив$(\angle(ADC)) \to \angle(BCD) + \angle(ADC) = \pi)$

Прием применяется в посылках задачи на доказательство либо на исследование. Хотя бы один из числовых атомов $\angle(BCD)$, $\angle(ADC)$ имеет тип "неизв".

Спецификация приема имеет вид "тип(узелраздела)". Справочник "заголовокприема" указывае уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм($\angle(BCD)$) терм($\angle(ADC)$))) неизв(1)))", "усм(актив(прямая(BC)))", "усм(актив(прямая(CD)))", "усм(актив(прямая(AD)))", "не(усм(принадлежит(D прямая(BC))))", "не(усм(принадлежит(C прямая(AD))))". Этого почти достаточно.

Число приемов данного типа - 9.

3.2.4.2.2. Соотношение связывает числовые атомы "неизв" через известные параметры.

3.2.4.2.2.1. Соотношение связывает числовые атомы "неизв" через известные параметры.

Пример:

$\forall_{ABC}$(прямая(AB) $\perp$ прямая(BC) $\rightarrow l(AC)^2 = l(AB)^2 + l(BC)^2$)

Прием применяется в посылках задачи на доказательство либо на исследование. Хотя бы один из числовых атомов $\angle(BCD)$, $\angle(ADC)$ имеет тип "неизв". В наборе $l(AC)$, $l(AB)$, $l(BC)$ каждый числовой атом либо известен, либо имеет тип "неизв", причем хотя бы один имеет тип "неизв".

Спецификация приема имеет вид "тип(приоритет)". Спавочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AC)$))", "усм(актив($l(AB)$))", "усм(актив($l(BC)$))", "конец(числатомы(набор(терм($l(AC)$) терм($l(AB)$) терм($l(BC)$)) смнеизв))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(BC)))", "не(равно(B C))", "не(равно(A C))". Этого почти достаточно.

Число приемов данного типа - 10.

3.2.4.2.2.2. Соотношение связывает числовые атомы "неизв".

Пример:

$\forall_{ABC}$(прямая(AB) $\perp$ прямая(AC) & актив($\angle(ABC)$) $\rightarrow l(AB) =$ $\cos(\angle(ABC))l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Числовые атомы $l(AB)$, $\angle(ABC)$, $l(BC)$ имеют тип "неизв".

Спецификация приема имеет вид "тип(утверждение)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AB)$))", "усм(актив($l(BC)$))", "конец(числатомы(набор(терм($l(AB)$) терм($\angle(ABC)$) терм($l(BC)$)) неизв(3)))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 3.

3.2.4.2.2.3. Соотношение связывает между собой два числовых атома "неизв" через известные параметры.

Пример:

$\forall_{ABCD}$(прямая(AB) $\perp$ прямая(BC) & актив($\angle(BAD)$) & прямая(DC) $\parallel$ прямая(AB) $\rightarrow l(BC) = \sin(\angle(BAD))l(AD)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Один из атомов $l(BC)$, $\angle(BAD)$, $l(AD)$ известен, два других - имеют тип "неизв".

Спецификация приема имеет вид "тип(случаи)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(BC)$))", "усм(актив($l(AD)$))", "конец(числатомы(набор(терм($l(BC)$) терм($\angle(BAD)$) терм($l(AD)$))) и(неизв(2) известно(1))))", "усм(актив(прямая(AB)))", "не(равно($A\ B$))", "усм(актив(прямая(BC)))", "не(равно($B\ C$))", "усм(актив(прямая(AD)))", "не(усм(принадлежит(D прямая(AB))))", "не(равно($A\ D$))". Этого почти достаточно.

Число приемов данного типа - 6.

3.2.4.2.3. Соотношение связывает старый числовой атом с атомами "неизв" и известными числовыми атомами.

3.2.4.2.3.2. Соотношение связывает атом "неизв" со старым атомом через известные параметры.

3.2.4.2.3.2.1. Соотношение связывает атом "неизв" со старым атомом через известные параметры.

Пример:

$\forall_{ABCD}$(прямая(AB) $\perp$ прямая(BC) & актив($\angle(BAD)$) & прямая(DC) $\|$ прямая(AB) $\rightarrow l(BC) = \sin(\angle(BAD))l(AD)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Один из атомов $l(BC)$, $\angle(BAD)$, $l(AD)$ известен, другой - уже встречается в задаче, третий - имеет тип "неизв".

Спецификация приема имеет вид "тип(пара)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(BC)$))", "усм(актив($l(AD)$))", "конец(числатомы(набор(терм($l(BC)$) терм($\angle(BAD)$)терм($l(AD)$)) и(неизв(1) актив(1) известно(1))))", "усм(актив(прямая(AB)))", "не(равно($A\ B$))", "усм(актив(прямая(BC)))", "не(равно($B\ C$))", "усм(актив(прямая(AD)))", "не(усм(принадлежит(D прямая(AB))))", "не(равно($A\ D$))". Этого почти достаточно.

Число приемов данного типа - 4.

3.2.4.2.3.3. Соотношение связывает два атома "неизв" со старым атомом через известные параметры.

Пример:

$\forall_{ABC}$(актив($\angle(BAC)$) & актив($l(AC)$) & актив($\angle(ABC)$) $\rightarrow \sin(\angle(BAC))l(AC)$ $= \sin(\angle(ABC))l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояние $l(BC)$ уже рассматривается в задаче. Среди атомов $\angle(BAC)$, $l(AC)$, $\angle(ABC)$, $l(BC)$ имеется хотя бы один известный и хотя бы два, имеющих тип "неизв".

Спецификация приема имеет вид "тип(знаксуммы)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(BC)$))", "конец(числатомы(набор(терм($\angle(BAC)$) терм($l(AC)$) терм(угол(ABC)) терм($l(BC)$)) и(неизв(2) известно(1))))", "усм(актив(прямая(AB)))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))", не(равно(A C))", "не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 3.

3.2.4.3. Соотношение связывает числовые атомы "Неизв" через известные параметры.

Пример:

$\forall_{ABCD}$(прямая(AB) $\perp$ прямая(AC) & $D \in$ прямая(BC) & прямая(AD) $\perp$ прямая(BC) $\rightarrow l(BD)l(BC) = l(AB)^2$)

Прием применяется в посылках задачи на доказательство либо на исследование. Числовые атомы $l(BD)$, $l(BC)$, $l(AB)$ уже встречаются в задаче, причем каждый из них либо известен, либо имеет тип "Неизв". Хотя бы один атом не известен.

Спецификация приема имеет вид "тип(Степень)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(BD)$))", "усм(актив($l(BC)$))", "усм(актив($l(AB)$))", "конец(числатомы(набор(терм($l(BD)$) терм($l(BC)$) терм($l(AB)$)) смНеизв))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(прямая(BC)))", "не(равно(B C))", "усм(актив(прямая(AD)))", "не(равно(A D))", "не(равно(B D))". Этого почти достаточно.

Число приемов данного типа - 5.

3.2.6. Соотношение пропорциональности, позволяющее получить численное уравнение.

Пример:

$\forall_{ABC}$(актив($\angle(BAC)$) & актив($l(AC)$) & актив($\angle(ABC)$) $\rightarrow \sin(\angle(BAC))l(AC) = \sin(\angle(ABC))l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояние $l(BC)$ уже встречается в задаче. Выводимое соотношение пропорциональности позволяет, в сочетании с некоторым уравнением текущей задачи, получить невырожденное уравнение для численных параметров. Последнее условие проверяется оператором "пропорцуравн".

Спецификация приема имеет вид "тип(блокзамен)". Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(BC)$))", "конец(пропорцуравн(

фикс(0 1) фикс(0 2)))", "не(известно(результат))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(A C))", "не(равно(B C))". Требуется доработка.

Число приемов данного типа - 5.

3.2.7. Соотношение для численных параметров, содержащее неизвестную.

Пример:

$\forall_{ABC}(l(AB) = l(BC)$ & актив$(\angle(ABC)) \to 2l(AB)\sin(\angle(ABC)/2) = l(AC))$

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояние $l(AC)$ уже встречается в задаче. Все параметры выражений для числовых атомов $l(AB)$, $\angle(ABC)$, $lAC)$ численные.

Спецификация приема имеет вид "тип(фигура)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AC)$))", "конец(числатомы(набор(терм($l(AB)$) терм($\angle(ABC)$) терм($l(AC)$))Числпарам))", "усм(актив(прямая(AB)))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(A B))", "не(равно(A C))". Этого почти достаточно.

Число приемов данного типа - 3.

3.3. Соотношение связывает числовой атом с численными параметрами.

3.3.1. Соотношение связывает числовой атом с численными параметрами.

Пример:

$\forall_{ABC}($прямая$(AB) \perp$ прямая$(BC) \to l(AC)^2 = l(AB)^2 + l(BC)^2)$

Прием применяется в посылках задачи на доказательство либо на исследование. Все атомы набора $l(AC)$, $l(AB)$, $l(BC)$, кроме, быть может, одного, выразимы через численные параметры - известные либо неизвестные. Хотя бы один из этих атомов не известен.

Спецификация приема имеет вид "тип(множители)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм($l(AC)$) терм($l(AB)$) терм($l(BC)$))) Числопред(фикс)))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(BC)))", "не(равно(B C))", "не(равно(A C))". Этого почти достаточно.

Число приемов данного типа - 5.

3.3.2. Определение числового атома.

3.3.2.1. Определение числового атома.

Пример:

$\forall_{ABC}$(прямая$(AB) \perp$ прямая(AC) & актив$(\angle(ABC)) \rightarrow l(AC) = \sin(\angle(ABC))l(BC))$

Прием применяется в посылках задачи на доказательство либо на исследование. Все атомы набора $l(AC)$, $l(AB)$, $l(BC)$, кроме одного, известны.

Спецификация приема имеет вид "тип(днф)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм($l(AC)$) терм($\angle(ABC)$) терм($l(BC)$))) опред(фикс)))", "усм(актив(прямая(AB))))", "не(равно$(A\ B)$)", "усм(актив(прямая(AC))))", "не(равно$(A\ C)$)", "усм(актив(прямая(BC))))", "не(усм(принадлежит(C прямая(AB))))", "не(равно$(B\ C)$)". Этого почти достаточно.

Число приемов данного типа - 24.

3.3.2.2. Определение применимого числового атома.

Пример:

$\forall_{ABCDE}$(окружность(DE)описана около фигура(ABC) & актив$(l(AC)) \rightarrow 2\sin(\angle(ABC))l(DE) = l(AC))$

Прием применяется в посылках задачи на доказательство либо на исследование. Два атома набора $\angle(ABC)$, $l(DE)$, $l(AC)$ известны, третий - имеет тип "применимо". Напомним, что такой тип усматривается пакетным индикатором "применимо". Он означает возможность использования атома для получения численного уравнения.

Спецификация приема имеет вид "тип(задачи)". Справочник "заголовокприема" указывает уровень срабатывания 10 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм($\angle(ABC)$) терм($l(DE)$) терм($l(AC)$))) и(применимо(1) известно(2))))", "не(равно$(D\ E)$)", "усм(актив(прямая(AC))))". Этого почти достаточно.

Число приемов данного типа - 4.

3.4. Уравнение, связывающее числовой атом с численными параметрами.

3.4.3. Уравнение для определения значения числового атома.

3.4.3.3. Уравнение для определения значения старого числового атома.

3.4.3.3.1. Уравнение для определения значения старого числового атома.

Пример:

$\forall_{ABCD}$(прямая$(AB) \perp$ прямая(BC) & $D \in$ отрезок(AC) & $l(AB) = l(BD)$ & разныеточки(A, D) & актив$(l(AD))$ & актив$(l(AC)) \rightarrow l(AD)l(AC) = 2l(AB)^2)$

Прием применяется в посылках задачи на доказательство либо на исследование. Выражения $l(AD)$, $l(AC)$, $l(AB)$ содержат ровно один неизвестный числовой атом. Он уже встречается в задаче.

Спецификация приема имеет вид "тип(сборканабора)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(Числатомы(набор(терм((AD)), терм($l(AC)$), терм($l(AB)$))) смактив))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(BC)))", "не(равно(B C))", "не(равно(A C))", "не(равно(A D))", "не(равно(C D))". Этого почти достаточно.

Число приемов данного типа - 5.

3.4.3.3.2. Уравнение для определения значения числового атома "неизв".

Пример:

$\forall_{ABCDEF}$(окружность(EF)вписана в фигура($ABCD$) & $l(AB) = l(CD)$ & трапеция($ABCD$) $\rightarrow 4l(EF)^2 = l(BC)l(AD)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Выражения $l(EF)$, $l(BC)$, $l(AD)$ содержат ровно один неизвестный числовой атом. Он имеет тип "неизв".

Спецификация приема имеет вид "тип(выделениестепени)". Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(EF)$))", "усм(актив($l(BC$))", "усм(актив($l(AD)$))", "конец(Числатомы(набор(терм($l(EF)$), терм($l(BC)$), терм($l(AD)$))) неизв))", "не(равно(E F))", "не(равно(B C))", "не(равно(A D))". Этого почти достаточно.

Число приемов данного типа - 4.

3.6. Уравнение, связывающее два числовых атома через посредство численных параметров.

Пример:

$\forall_{ABCD}$(актив($l(AD)$) & актив($l(AB)$) & актив($l(BC)$) & актив($l(CD)$) &
$B \in$ отрезок(AD) & $C \in$ отрезок(BD) $\rightarrow l(AD) = l(AB) + l(BC) + l(CD)$)

Прием применяется в посылках задачи на доказательство либо на исследование. После обработки нормализаторами общей стандартизации все атомы списка $l(AD)$, $l(AB)$, $l(BC)$, $l(CD)$, кроме двух, выражены через численные параметры.

Спецификация приема имеет вид "тип(вписанныйугол)". Справочник "заголовокприема" указывает уровень срабатывания 10 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(Числатомы(набор(терм($l(AD)$), терм($l(AB)$), терм($l(BC)$), терм($l(CD)$))) числсвязь))", "не(равно(A D))", "не(равно(B D))", "не(равно(A B))", "не(равно(B C))", "не(равно(C D))". Этого почти достаточно.

Число приемов данного типа - 3.

3.7. Соотношение с участием числового атома "неизв".

3.7.1. Соотношение с участием числового атома "неизв".

Пример:

$\forall_{ABCD}(\angle(ABD) = \angle(DBC)$ & $D \in$ прямая(AC) & разныеточки(A, C) & разныепрямые(прямая(AB), прямая$(BC)) \rightarrow l(AB)l(CD) = l(AD)l(BC))$

Прием применяется в посылках задачи на доказательство либо на исследование. Среди атомов $l(AB)$, $l(CD)$, $l(AD)$, $l(BC)$ имеется атом типа "неизв".

Спецификация приема имеет вид "тип(косинус)". Справочник "заголовокприема" указывает уровень срабатывания 9 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм($l(AB)$), терм($l(CD)$), терм($l(AD)$), терм($l(BC)$)) неизв(1)))", "усм(актив(прямая(AC)))", "не(равно($A\ C$))", "не(усм(принадлежит(D прямая(AB))))", "не(усм(принадлежит(прямая(BA))))", "не(усм(принадлежит(C прямая(BD))))", "не(равно($A\ B$))", "не(равно($C\ D$))", "не(равно($A\ D$))", "не(равно($B\ C$))", "не(равно(терм(прямая(AB)) терм(прямая(BC))))". Этого почти достаточно.

Число приемов данного типа - 5.

3.7.3. Соотношение связывает числовой атом с атомами "неизв" и известными числовыми атомами.

3.7.3.1. Соотношение связывает числовой атом с атомами "неизв" и известными числовыми атомами.

Пример:

$\forall_{ABCD}(\angle(ABD) = \angle(DBC)$ & $D \in$ прямая(AC) & разныеточки(A, C) & разныепрямые(прямая(AB), прямая$(BC)) \rightarrow l(AB)l(CD) = l(AD)l(BC))$

Прием применяется в посылках задачи на доказательство либо на исследование. Среди атомов $l(AB)$, $l(CD)$, $l(AD)$, $l(BC)$ имеется не более одного, не известного в текущей задаче и не имеющего типа "неизв".

Спецификация приема имеет вид "тип(определение)". Справочник "заголовокприема" указывает уровень срабатывания 12 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм($l(AB)$), терм($l(CD)$), терм($l(AD)$), терм($l(BC)$)) новатом))", "не(известно(результат))", "усм(актив(прямая(AC)))", "не(равно($A\ C$))", "не(усм(принадлежит(D прямая(AB))))", "не(усм(принадлежит(прямая(BA))))", "не(усм(принадлежит(C прямая(BD))))", "не(равно($A\ B$))", "не(равно($C\ D$))", "не(равно($A\ D$))", "не(равно($B\ C$))", "не(равно(терм(прямая(AB)) терм(прямая(BC))))". Этого почти достаточно.

Число приемов данного типа - 4.

3.7.3.2. Соотношение связывает числовой атом с атомом "неизв" через известные параметры.

Пример:

$\forall_{ABC}$(прямая(AB) $\perp$ прямая(AC) & актив($\angle(ABC)$) $\rightarrow$ $l(AB) = \cos(\angle(ABC))l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Среди атомов $l(AB)$, $\angle(ABC)$, $l(BC)$ имеется хотя бы один известный и хотя бы один, имеющий тип "неизв".

Спецификация приема имеет вид "тип(натурстепень)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм($l(AB)$), терм($\angle(ABC)$), терм($l(BC)$))) и(неизв(1) известно(1))))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))","не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 9.

3.7.6. Соотношение связывает числовой атом "неизв" с определимыми атомами.

3.7.6.2. Соотношение связывает числовой атом "неизв" с определимым атомом через известные параметры.

3.8. Соотношение с участием применимых атомов.

3.9. Связь двух числовых атомов через определимые параметры.

3.9.2. Связь двух числовых атомов через известные параметры.

4. Общий случай вывода равенств для невырожденных числовых атомов (асимметричные типы).

В этом разделе рассматриваются типы приемов, у которых условия на различные присутствующие в теореме числовые атомы различны. Ввиду большой глубины вложенности подпунктов, по-прежнему будет использоваться обозначение их с помощью отделенных точками номеров.

4.1. Определение заданного неизвестного числового атома.

4.1.1. Определение заданного неизвестного числового атома.

Пример:

$\forall_{ABCEF}$(окружность(EF)вписана в фигура(ABC) $\rightarrow$ $l(EF)$периметр(фигура(ABC)) = $2S$(фигура(ABC)))

Прием применяется в посылках задачи на доказательство либо на исследование. Атомы $l(EF)$ и S(фигура(ABC)) известны, атом периметр(фигура(ABC)) не известен.

Спецификация приема имеет вид "тип(просмотртермов)", "терм(t)", где t - не известный атом. Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(EF)$))", "контекст(посылка(x1) вид(x1 актив(площадь(фигура(набор(ABC))))) множество(набор(ABC)))", "известно(терм($l(EF)$))", "известно(терм(площадь(фигура(набор(ABC)))))", "конец(не(известно(терм(периметр(фигура(набор(ABC))))))", "не(равно(E F))". Этого почти достаточно.

Число приемов данного типа - 23.

4.1.2. Определение заданного существенного числового атома.

4.1.2.1. Определение заданного существенного числового атома.

Пример:

$\forall_{ABC}$(актив($l(AB)$) & $B \in$ отрезок(AC) $\rightarrow l(AC) = l(AB) + l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Атомы $l(AB)$ и $l(BC)$ известны, атом $l(AC)$ не известен, причем характеризуется пакетным индикатором "существатом" как существенный.

Спецификация приема имеет вид "тип(посылки)", "терм(t)", где t - существенный атом. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(BC)$))", "конец(известно(терм($l(AB)$)))", "конец(известно(терм($l(BC)$)))", "конец(не(известно(терм($l(AC)$))))", "конец(легковидеть(существатом($l(AC)$)))", "не(равно(A C))", "не(равно(A B))", "не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 9.

4.1.2.3. Определение заданного применимого числового атома.

4.1.2.3.1. Определение заданного применимого числового атома.

Пример:

$\forall_{ABC}$(актив($\angle(ABC)$) & актив($\angle(BCA)$) $\rightarrow \angle(ABC) + \angle(BCA) + \angle(BAC) = \pi$)

Прием применяется в посылках задачи на доказательство либо на исследование. Углы ABC и BCA известны, а угол BAC - не известен. Пакетный индикатор усматривает, что он имеет тип "применимо".

Спецификация приема имеет вид "тип(дизъюнктчлен)", "терм(t)", где t - применимый атом. Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))",

"известно(терм(угол(ABC)))", "известно(терм(угол(BCA)))", "конец(легковидеть(применимо(угол(BAC))))", "конец(не(известно(терм(угол(BAC)))))", "усм(актив(прямая(BC)))", "усм(актив(прямая(AC)))", "не(усм(принадлежит(A прямая(BC))))". Требуется доработка.

Число приемов данного типа - 13.

4.1.2.3.2. Определение заданного числового атома типа "неизв".

4.1.2.3.2.1. Определение заданного числового атома типа "неизв".

Пример:

$\forall_{ABC}$(прямая(AB) $\perp$ прямая(AC) & актив($\angle(ABC)$) $\rightarrow$ $l(AC) =$ tg($\angle(ABC)$)$l(AB)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояния $l(AC)$ и $l(AB)$ известны, угол ABC имеет тип "неизв".

Спецификация приема имеет вид "тип(нормоператора)", "терм(t)", где t - атом "неизв". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AC)$))", "усм(актив($l(AB)$))", "известно(терм($l(AC)$))", "известно(терм($l(AB)$))", "неизв(терм($\angle(ABC)$))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))". Этого почти достаточно.

Число приемов данного типа - 21.

4.1.3. Определение заданного старого числового атома.

4.1.3.1. Определение заданного старого числового атома.

В качестве примера рассмотрим прием, теорема которого та же, что в предыдущем пункте. Он применяется в посылках задачи на доказательство либо на исследование. Расстояния $l(AB)$, $l(AC)$, $l(BC)$ известны. Угол $\angle(BAC)$ не известен, но уже встречается в задаче.

Спецификация приема имеет вид "тип(учетответа)", "терм(t)", где t - не известный атом. Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AB)$))", "усм(актив($l(AC)$))", "усм(актив($l(BC)$))", "известно(терм($l(AB)$))", "известно(терм($l(AC)$))", "известно(терм($l(BC)$))", "не(известно(терм($\angle(BAC)$)))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(A B))", "не(равно(A C))", "не(равно(B C))". Требуется доработка.

Число приемов данного типа - 6.

4.1.4. Определение заданного неизвестного числового атома "возмсвяз".

4.1.5. Определение заданного неизвестного числового атома, косвенно связанного со старыми объектами.

Пример:

$\forall_{ABCDEF}$(актив(окружность(AB)) & $C \in$ окружность(AB) &
$D \in$ окружность(AB) & $E \in$ окружность(AB) & $F \in$ окружность(AB) &
актив($\angle(CED)$) & разныестороны($E, F,$ прямая(CD)) $\rightarrow$
$\angle(CED) + \angle(CFD) = \pi$)

Прием применяется в посылках задачи на доказательство либо на исследование. Угол $\angle(CED)$ известен. Угол $\angle(CFD)$ не известен. Прямые CF и FD - стороны этого угла - уже рассматриваются в задаче.

Спецификация приема имеет вид "тип(модуль)", "терм(t)", где t - неизвестный атом. Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(прямая(CF)))", "усм(актив(прямая(DF)))", "известно(терм(угол(CED)))", "конец(не(известно(терм(угол(CFD)))))", "не(равно(C E))", "не(равно(C F))", "не(равно(D E))", "не(равно(D F))", "не(равно(E F))", "усм(актив(прямая(CE)))", "усм(актив(прямая(DE)))", "не(усм(принадлежит(D прямая(CE))))". Этого почти достаточно.

Число приемов данного типа - 3.

4.2. Связь между двумя заданными числовыми атомами через посредство группы определимых атомов.

4.2.5. Связь между двумя заданными числовыми атомами через посредство группы известных атомов.

4.2.5.1. Связь между двумя заданными числовыми атомами через посредство группы известных атомов.

Пример:

$\forall_{ABCD}$(прямая(AB) $\perp$ прямая(AC) & $D \in$ прямая(BC) & прямая(AD) $\perp$
прямая(BC) $\rightarrow l(BD)l(CD) = l(AD)^2$)

Прием применяется в посылках задачи на доказательство либо на исследование. Атом $l(AD)$ известен.

Спецификация приема имеет вид "тип(усмнечетное)", "терм(t_1)", "терм(t_2)", где t_1, t_2 - атомы, между которыми устанавливается связь. Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AD)$))", "известно(терм($l(AD)$))", "не(известно(результат))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(прямая(BC)))", "не(равно(B C))", "усм(актив(прямая(AD)))", "не(равно(A D))", "не(равно(B D))", "не(равно(C D))". Этого почти достаточно.

Число приемов данного типа - 6.

4.2.5.2. Связь между двумя заданными числовыми атомами, хотя бы один из которых имеет тип "неизв", через посредство группы известных атомов.

4.2.5.2.1. Связь между двумя заданными числовыми атомами, хотя бы один из которых имеет тип "неизв", через посредство группы известных атомов.

Пример:

$\forall_{ABC}$(актив($\angle(BAC)$) $\to 2l(AB)l(AC)\cos(\angle(BAC)) = l(AB)^2+l(AC)^2-l(BC)^2$)

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояния AB и AC известны. Хотя бы один из атомов $l(BC)$, $\angle(BAC)$ имеет тип "неизв".

Спецификация приема имеет вид "тип(усммножество)", "терм(t_1)", "терм(t_2)", где t_1, t_2 - заданные атомы, хотя бы один из которых имеет тип "неизв". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AB)$))", "усм(актив($l(AC)$))", "конец(известно(терм($l(AB)$)))", "конец(известно(терм($l(AC)$)))", "конец(или(неизв(терм($\angle(BAC)$)) неизв(терм($l(BC)$)))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(A B))", "не(равно(A C))", "не(равно(B C))". Требуется доработка.

Число приемов данного типа - 7.

4.2.5.2.2. Связь между двумя заданными старыми числовыми атомами, хотя бы один из которых имеет тип "неизв", через посредство группы известных атомов.

4.2.5.2.2.1. Связь между двумя заданными старыми числовыми атомами, хотя бы один из которых имеет тип "неизв", через посредство группы известных атомов.

Пример:

$\forall_{ABC}$(прямая(AB) $\perp$ прямая(AC) & актив($\angle(ABC)$) $\to l(AB) = \cos(\angle(ABC))l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Угол $\angle(ABC)$ известен. Расстояния $l(AB)$, $l(BC)$ уже рассматриваются в задаче, причем хотя бы одно из них имеет тип "неизв".

Спецификация приема имеет вид "тип(рациональное)", "терм(t_1)", "терм(t_2)", где t_1, t_2 - заданные атомы, хотя бы один из которых имеет тип "неизв". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AB)$))", "усм(актив($l(BC)$))", "конец(известно(терм($\angle(ABC)$)))", "конец(или(неизв(терм($l(AB)$)) неизв(терм($l(BC)$)))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(прямая(BC)))",

"не(усм(принадлежит(C прямая(AB))))","не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 4.

4.2.5.2.2.2. Связь заданного атома "неизв" с заданным старым атомом через посредство группы известных атомов.

Пример:

$\forall_{ABCDE}(C \in$ окружность(AB) & $D \in$ окружность(AB) & $E \in$ окружность(AB) & актив$(\angle(CED)) \to l(CD) = 2l(AB)\sin(\angle(CED)))$

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояние $l(CD)$ известно. Угол $\angle(CED)$ имеет тип "неизв", расстояние $l(AB)$ уже рассматривается в задаче.

Спецификация приема имеет вид "тип(младшиечлены)", "неизв(t)", "терм(r)", где t - атом типа "неизв"; r - старый атом, связываемый с атомом "неизв". Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(неизв(терм($\angle(CED)$))))", "усм(актив($l(CD)$))", "усм(актив($l(AB)$))", "конец(известно(терм($l(CD)$))))", "не(равно(C D))", "не(усм(принадлежит(D прямая(CE))))", "не(равно(A B))". Этого почти достаточно.

Число приемов заданного типа - 4.

4.2.5.2.2.3. Связь между двумя заданными атомами "неизв" через посредство группы известных атомов.

Пример:

$\forall_{ABCD}$(актив($l(AB)$) & прямая$(AB) \perp$ прямая(AC) & актив$(\angle(CBD))$ & $B \in$ отрезок$(AD) \to l(AB) = -\cos(\angle(CBD))l(BC))$

Прием применяется в посылках задачи на доказательство либо на исследование. Угол "$\angle(CBD)$" известен. Расстояния $l(AB)$, $l(CD)$.

Спецификация приема имеет вид "тип(величина)", "терм(t_1)", "терм(t_2)", где t_1, t_2 - атомы типа "неизв". Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(BC)$))", "конец(известно(терм($\angle(CBD)$)))", "конец(неизв(терм($l(AB)$)))", "конец(неизв(терм($l(BC)$)))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(прямая(BC)))", "усм(актив(прямая(BD)))", "не(равно(A D))", "не(усм(принадлежит(D прямая(BC))))", "не(равно(B D))", "не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 3.

4.2.5.2.3. Связь заданного атома "неизв" с заданным другим числовым атомом через группу известных атомов.

4.2.5.2.3.1. Связь заданного атома "неизв" с заданным другим числовым атомом через группу известных атомов.

Пример:

$\forall_{ABC}$(актив($l(AB)$) & актив($l(AC)$) & $B \in$ прямая(AC) & точкалуча(C, A, B) $\rightarrow$ $l(AB) = |l(AC) - l(BC)|$)

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояние $l(AC)$ известно, растояние $l(AB)$ имеет тип "неизв".

Спецификация приема имеет вид "тип(подфрагмент)", "терм(t)", "неизв(r)". Здесь r - атом "неизв", t - другой из атомов, между которыми устанавливается связь. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "неизв(терм($l(AB)$))", "конец(известно(терм($l(AC)$))", "усм(актив(прямая(AC))", "не(равно(A C))", "не(равно(B C))", "не(равно(A B))". Требуется доработка.

Число приемов данного типа - 5.

4.2.5.2.3.4. Связь заданного атома "неизв" с заданным новым числовым атомом через группу известных атомов.

4.2.5.4. Связь заданного существенного атома с заданным определимым через посредство группы известных атомов.

4.2.5.5. Связь заданного неизвестного старого атома с другим неизвестным атомом через группу известных атомов.

4.2.5.8. Связь между двумя заданными существенными числовыми атомами через посредство группы известных атомов.

В качестве примера рассмотрим прием, теорема которого - та же, что в предыдущем пункте. Он применяется в посылках задачи на доказательство либо на исследование. Числовые атомы периметр(фигура(ABC)) и S(фигура(ABC)) имеют тип "существатом", т.е. одноименный пакетный индикатор усматривает возможность связать их с числовыми атомами, входящими в уравнения с неизвестными. Расстояние EF известно.

Спецификация приема имеет вид "тип(пи)", "терм(t_1)", "терм(t_2)", где t_1, t_2 - числовые атомы типа "неизвпарам". Справочник "заголовокприема" указывает уровень срабатывания 10 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(EF)$))", "известно(терм($l(EF)$))", "легковидеть(существатом(периметр(фигура(набор(ABC)))))", "легковидеть(существатом(площадь(фигура(набор(ABC)))))", "не(известно(результат))", "не(равно(E F))". Этого почти достаточно.

Число приемов данного типа - 3.

4.2.5.12. Связь текущего атома с другим невырожденным атомом через группу известных атомов.

4.3. Связь заданной группы старых атомов с другими атомами.

4.3.1. Связь заданной группы старых атомов с другими атомами.

Пример:

$\forall_{ABCDEFab}$(актив(S(фигура(ACB))) & $C \in$ отрезок(AD) & $al(AC) = bl(CD)$ & $E \in$ отрезок(BD) & $F \in$ отрезок(BD) & актив(S(фигура(AEF))) $\rightarrow$ $bl(BD)S$(фигура(AEF)) $= (a+b)l(EF)S$(фигура(ACB)))

Прием применяется в посылках задачи на доказательство либо на исследование. Площади уже рассматриватся в задаче; для расстояний это не требуется.

Спецификация приема имеет вид "тип(сохрзначений)", "терм(t_1)", ..., "терм(t_n)", где t_i - старые атомы. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(равно(A D))", "не(равно(B D))", "не(равно(E F))", "не(равно(A C))", "не(равно(C D))", "не(равно(B E))", "не(равно(D E))", "не(равно(B F))", "не(равно(D F))". Этого почти достаточно.

Число приемов данного типа - 3.

4.3.6. Связь заданной группы старых атомов через численные параметры.

4.3.7. Связь группы заданных числовых атомов, выразимых через численные параметры, с другими атомами.

4.3.7.1. Связь группы заданных числовых атомов, выразимых через численные параметры, с другими атомами.

Пример:

$\forall_{ABCDEFGH}$(прямая(AB) $\parallel$ прямая(DE) & прямая(AC) $\parallel$ прямая(DF) & прямая(BC) $\parallel$ прямая(EF) & прямая(CG) $\perp$ прямая(AB) & прямая(FH) $\perp$ прямая(DE) & $G \in$ прямая(AB) & $H \in$ прямая(DE) & актив($l(CG)$) & актив($l(FH)$) & разныепрямые(прямая(AB), прямая(BC)) $\rightarrow$ $l(CG)l(DE) = l(FH)l(AB)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Каждое из расстояний $l(CG)$, $l(FH)$ либо известно, либо имеет тип "внешнеизв".

Спецификация приема имеет вид "тип(определениепараметра)", "терм(t_1)", ..., "терм(t_n)", где $t_1, \ldots, t_n$ - атомы, выразимые через численные параметры (т.е. известные либо типа "внешнеизв"). Справочник "заголовокприема" указывает уровень срабатывания 13 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "числатомы(набор(терм(расстояние(CG)) терм(расстояние(FH))) числпарам)", "не(известно(результат))", "усм(актив(прямая(AC)))",

"не(равно(A C))", "усм(актив(прямая(DF)))", "не(равно(D F))", "усм(актив(прямая(BC)))", "не(равно(B C))", "усм(актив(прямая(EF)))", "не(равно(E F))", "усм(актив(прямая(CG)))", "не(равно(C G))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(FH)))", "не(равно(F H))", "усм(актив(прямая(DE)))", "не(равно(D E))", "не(равно(терм(прямая(AC)) терм(прямая(DF))))", "не(равно(терм(прямая(BC)) терм(прямая(EF))))", "не(равно(терм(прямая(AB)) терм(прямая(BC))))". Требуется доработка.

Число приемов данного типа - 3.

4.3.9. Связь заданного старого атома с другими атомами.

4.3.9.1. Связь заданного старого атома с другими атомами.

Пример:

$\forall_{ABCDE}$(окружность(DE)описана около фигура(ABC) & актив($\angle(ABC)$) $\rightarrow$ $2\sin(\angle(ABC))l(DE) = l(AC)$)

Прием применяется в посылках задачи на исследование. Угол ABC уже встречается в задаче.

Спецификация приема имеет вид "тип(попытка)", "терм(t)", где t - старый атом. Справочник "заголовокприема" указывает уровень срабатывания 9 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(прямая(AB)))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(D E))", "не(равно(A C))". Требуется доработка.

Число приемов данного типа - 20.

4.3.9.2. Связь текущего атома с другими атомами.

Пример:

$\forall_{ABC}$(услвероятн(A, B, C)вероятность(B, C) = услвероятн(B, A, C) $\cdot$ вероятность(A, C))

Прием применяется в посылках задачи на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении выражения "услвероятн(A, B, C)". Проверяется существование в задаче выражения вида "услвероятн(D, E, F)", у которого D имеет общую переменную с B.

Спецификация приема имеет вид "тип(буфер)", "терм(t)", где t - текущий атом. Справочник "заголовокприема" указывает уровень срабатывания 5, вводит указатель "контрольвывода(услвероятн(A, B, C))" и фильтр "тип(исследовать)". Требуется доработка.

Число приемов данного типа - 12.

4.3.9.7. Связь заданного атома "Неизв" с другими атомами.

4.3.9.7.2. Связь заданного атома "неизв" с другими атомами.

4.3.9.7.2.1. Связь заданного атома "неизв" с другими атомами.

Пример:

$\forall_{ABCEF}$(окружность(EF) вписана в фигура(ABC) $\rightarrow$
$l(EF)$периметр(фигура(ABC)) $= 2S$(фигура(ABC)))

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояние EF имеет тип "неизв".

Спецификация приема имеет вид "тип(разборслучаев)", "терм(t)", где t - заданный атом типа "неизв". Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(EF)$))", "неизв(терм($l(EF)$))", "не(равно(E F))". Этого почти достаточно.

Число приемов данного типа - 13.

4.3.9.7.2.2. Связь заданного атома "неизв" со старыми атомами.

4.3.9.7.2.2.3. Соотношение связывает заданный числовой атом "неизв" с численными параметрами.

Пример:

$\forall_{ABC}$(актив($\angle(BAC)$) $\rightarrow 2l(AB)l(AC)\cos(\angle(BAC)) = l(AB)^2+l(AC)^2-l(BC)^2$)

Прием применяется в посылках задачи на доказательство либо на исследование. Угол BAC имеет тип "неизв". Каждое из расстояний AB, AC, BC либо известно, либо имеет тип "внешнеизв".

Спецификация приема имеет вид "тип(редакцияфильтра)", "терм(t)", где t - атом типа "неизв". Справочник "заголовокприема"указывает уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "неизв(терм(угол(BAC)))", "усм(актив($l(AB)$))", "усм(актив($l(AC)$))", "усм(актив($l(BC)$))", "конец(числатомы(набор(терм($l(AB)$) терм($l(AC)$)терм($l(BC)$)) числпарам))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(A B))", "не(равно(A C))", "не(равно(B C))". Требуется доработка.

Число приемов данного типа - 5.

4.3.10. Определение числового атома при заданных известных атомах.

4.3.10.2. Определение старого атома при заданных известных атомах.

Пример:

$\forall_{ABC}$(актив($\angle(BAC)$) & актив($l(AC)$) & актив($\angle(ABC)$) $\rightarrow \sin(\angle(BAC))l(AC)$
$= \sin(\angle(ABC))l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Каждый из углов BAC), ABC известен. Одно из расстояний AC, BC известно, а другое уже встречается в задаче.

Спецификация приема имеет вид "тип(подобл)", "терм(t_1)", ..., "терм(t_n)", где $t_1, \ldots, t_n$ - заданные известные атомы. В нашем примере - углы. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(BC)$))", "известно(терм($\angle(BAC)$))", "известно(терм($\angle(ABC)$))", "конец(числатомы(набор(терм($l(AC)$) терм($l(BC)$)) опред(актив)))", "не(известно(результат))", "усм(актив(прямая(AB)))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(A C))", "не(равно(B C))". Требуется доработка.

Число приемов данного типа - 3.

4.3.10.3. Определение атома "неизв" при заданных известных атомах.

Пример:

$\forall_{ABCD}(D \in$ отрезок(AC) & актив$(\angle(ABD))$ & актив$(\angle(DBC))$ & актив$(l(BD))$ & актив$(l(AB))$ & актив$(l(BC)) \rightarrow$
$l(BD)(l(AB)\sin(\angle(ABD)) + l(BC)\sin(\angle(DBC))) = l(BC)l(AB)\sin(\angle(ABD) + \angle(DBC)))$

Прием применяется в посылках задачи на доказательство либо на исследование. Каждый из углов ABD, DBC известен. Одно из расстояний D, AB, BC имеет тип "неизв", а другие - известны.

Спецификация приема имеет вид "тип(верхняяоценка)", "терм(t_1)", ..., "терм(t_n)", где $t_1, \ldots, t_n$ - заданные известные атомы. В нашем примере - углы. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "известно(терм($\angle(ABD)$))", "известно(терм($\angle(DBC)$))", "конец(числатомы(набор(терм($l(BD)$) терм($l(AB)$) терм($l(BC)$)) опред(неизв)))", "не(равно(A C))", "усм(актив(прямая(BD)))", "усм(актив(прямая(BC)))", "не(равно(A D))", "не(равно(D))", "не(усм(принадлежит(D прямая(AB))))", "не(усм(принадлежит(C прямая(BD))))", "не(равно(B D))", "не(равно(A B))", "не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 3.

4.3.11. Связь заданных атома "неизв" и старого атома с другими атомами.

4.3.11.1. Связь заданных атома "неизв" и старого атома с другими атомами.

Пример:

$\forall_{ABCEF}($окружность(EF) вписана в фигура$(ABC) \rightarrow$
$l(EF)$периметр(фигура$(ABC)) = 2S($фигура$(ABC)))$

Прием применяется в посылках задачи на доказательство либо на исследование. Числовой атом $S($фигура$(ABC))$ имеет тип "неизв". Расстояние EF уже рассматривается в задаче.

Спецификация приема имеет вид "тип(циклвариантов)", "неизв(r)", "терм(t)", где r - заданный атом типа "неизв", t - заданный старый атом. Справочник

"заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(неизв(терм(S(фигура(ABC)))))", "контекст(посылка(x1) вид(x1 актив(S(фигура(ABC)))) множество(набор(ABC)))", "усм(актив($l(EF)$))", "не(равно(E F))". Этого почти достаточно.

Число приемов данного типа - 5.

4.4. Связь заданного атома со старыми атомами.

4.4.1. Связь заданного атома со старыми атомами.

Пример:

$\forall_{ABCDEF}(C \in$ Окружность(ABF) & $D \in$ Окружность(ABF) &
$E \in$ Окружность(ABF) & актив$(\angle(CED)) \to l(CD) = 2l(AB)\sin(\angle(CED)))$

Прием применяется в посылках задачи на доказательство либо на исследование. Угол CED и расстояние AB уже встречаются в задаче.

Спецификация приема имеет вид "тип(дробнаявеличина)", "терм(t)", где t - заданный числовой атом. Справочник "заголовокприема" указывает уровень срабатывания 9 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AB)$))", "не(известно(результат))", "не(усм(принадлежит(D прямая(CE))))", "не(равно(C D))", "не(равно(A B))". Требуется доработка.

Число приемов данного типа - 11.

4.4.2. Связь заданного нового числового атома со старыми атомами.

4.4.2.2. Связь заданного нового атома "возмсвяз" со старыми атомами.

В качестве примера рассмотрим прием, теорема которого - та же, что в предыдущем пункте. Он применяется в посылках задачи на доказательство либо на исследование. Расстояние AC пока не рассматривается, но пакетный индикатор определяет, что оно имеет тип "возмсвяз". Расстояния AB и BC уже рассматриваются в задаче.

Спецификация приема имеет вид "тип(учетобоснований)", "терм(t)", где t - заданный новый числовой атом типа "возмсвяз". Справочник "заголовокприема" указывает уровень срабатывания 9 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AC)$))", "не(усм(актив($l(BC)$)))", "конец(легковидеть(возмсвяз($l(BC)$)))", "не(равно(A C))", "не(равно(A B))", "не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 3.

4.4.6. Выражение через заданный определимый атом и известные атомы некоторого старого атома.

4.4.6.2. Определение числового атома "неизв" с применением заданного определимого числового атома.

Пример:

$\forall_{ABC}$(актив($\angle(ABC)$) & актив($\angle(BCA)$) $\to \angle(ABC)+\angle(BCA)+\angle(BAC)=\pi$)

Прием применяется в посылках задачи на доказательство либо на исследование. Угол BAC имеет тип "определимо". Один из углов ABC, BAC известен, а другой имеет тип "неизв".

Спецификация приема имеет вид "тип(Длинанабора)", "терм(t)", где t - атом типа "определимо". Справочник "заголовокприема" указывает уровень срабатывания 9 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм($\angle(ABC)$) терм($\angle(BCA)$)) опред(неизв)))", "конец(легковидеть(определимо($\angle(BAC)$)))", "усм(актив(прямая(BC)))", "усм(актив(прямая(AC)))", "не(усм(принадлежит(A прямая(BC))))". Требуется доработка.

Число приемов данного типа - 3.

4.5. Выражение заданного числового атома через возможно определимые атомы.

4.5.1. Выражение заданного числового атома "неизв" через возможно определимые атомы.

4.5.1.2. Выражение заданного числового атома "неизв" через определимые атомы.

4.5.1.2.1. Выражение заданного числового атома "неизв" через определимые атомы.

$\forall_{ABC}$(актив($l(AB)$) & актив(прямая(AC)) & актив($l(BC)$) $\to$
$2l(AB)l(AC)\cos\angle(BAC)=l(AB)^2+(l(AC))^2-l(BC)^2$)

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояние BC имеет тип "неизв". Расстояния AB и AC, а также угол BAC имеют тип "определимо".

Спецификация приема имеет вид "тип(текпеременные)", "терм(t)", где t - заданный атом типа "неизв". Справочник "заголовокприема" указывает уровень срабатывания 8 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(неизв(терм($l(BC)$)))", "конец(легковидеть(определимо($l(AB)$) замечание(стоп $l(BC)$)))", "конец(легковидеть(определимо($l(AC)$) замечание(стоп $l(BC)$)))", "конец(легковидеть(определимо($\angle(BAC)$) замечание(стоп $l(BC$))))", "не(равно(A C))", "не(равно(A B))", "не(равно(B C))". Требуется доработка.

Число приемов данного типа - 7.

4.5.1.2.3. Определение заданного числового атома "неизв" с применением заданного определимого числового атома.

4.5.1.2.3.1. Определение заданного числового атома "неизв" с применением заданного определимого числового атома.

Пример:

$\forall_{ABC}$(актив($\angle(BAC)$) & актив($l(AC)$) & актив($l(BC)$) $\rightarrow \sin(\angle(BAC))l(AC) = \sin(\angle(ABC))l(BC)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояние $l(AC)$ имеет тип "неизв". Угол BAC и расстояние BC известны, угол ABC имеет тип "определимо".

Спецификация приема имеет вид "тип(числовойатом)", "терм(t)", "неизв(r)", где r - атом типа "неизв", r - атом типа "определимо". Справочник "заголовокприема" указывает уровень срабатывания 9 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(усм(актив($\angle(ABC)$)))", "известно(терм($\angle(BAC)$))", "неизв(терм($l(AC)$))", "известно(терм($l(BC)$))", "конец(легковидеть(определимо($\angle(ABC)$) замечание(стоп $l(AC)$)))", "не(усм(принадлежит(C прямая(AB))))", "не(равно(A C))", "не(равно(B C))". Требуется доработка.

Число приемов данного типа - 3.

4.6. Связь между числовыми атомами относительно заданных известных атомов.

4.6.2. Связь между старыми атомами относительно заданных известных атомов.

Пример:

$\forall_{ABC}$(актив($\angle(BAC)$) & $\angle(ABC) \rightarrow \sin(\angle(BAC) + \angle(ABC))l(AC) = \sin(\angle(ABC))l(AB)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояния AC и AB уже встречаются в задаче. Углы ABC и BAC известны.

Спецификация приема имеет вид "тип(Номера)", "терм(t_1)", ..., "терм(t_n)"где $t_1, \ldots, t_n$ - заданные известные атомы. Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AC)$))", "усм(актив($l(AB)$))", "конец(известно(терм($\angle(BAC)$)))", "конец(известно(терм($\angle(ABC)$)))", "не(известно(результат))", "усм(актив(прямая(AB)))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(A B))))", "не(равно(A C))", "не(равно(A B))". Требуется доработка.

Число приемов данного типа - 3.

4.6.3. Связь атомов "неизв" со старыми атомами относительно заданных известных атомов.

4.6.3.1. Связь атома "неизв" со старыми атомами относительно заданных известных атомов.

Пример:

$\forall_{ABC}$(прямая(AB) $\perp$ прямая(AC) & актив($\angle(ABC)$) $\rightarrow$ $l(AC) =$ tg($\angle(ABC)$)$l(AB)$)

Прием применяется в посылках задачи на исследование. Угол ABC известен. Хотя бы одно из расстояний AC, AB имеет тип "неизв". Оба они уже встречаются в задаче.

Спецификация приема имеет вид "тип(нормзадача)", "терм(t_1)", ..., "терм(t_n)", где $t_1, \ldots, t_n$ - заданные известные атомы. Справочник "заголовокприема" указывает уровень срабатывания 9 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив($l(AC)$))", "усм(актив($l(AB)$))", "конец(известно(терм($\angle(ABC)$)))", "конец(числатомы(набор(терм($l(AC)$) терм($l(AB)$)) неизв(1)))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))", "усм(актив(прямая(BC)))", "не(усм(принадлежит(C прямая(AB))))". Этого почти достаточно.

Число приемов данного типа - 3.

5. Вывод равенства старых объектов.

Пример:

$\forall_{ABac}(A \cap B = \{c\}$ & $a \in A$ & $a \in B \rightarrow a = c)$

Прием применяется без ограничений. Первые два антецедента идентифицируются с посылками, третий - обрабатывается проверочным оператором.

Спецификация приема имеет вид "тип(склейкаоперандов)". Справочник "заголовокприема" указывает уровень срабатывания 1, вводит фильтр "посылка" и указатели "равно(1)", "блокпроверок(3)", "операнд(B фикс(1 1))". Требуется небольшая доработка.

Число приемов данного типа - 38.

6. Вывод нечислового равенства.

6.1. Вывод нечислового равенства.

Пример:

$\forall_{fgijax}(f =$ таблица($\{i \rightarrow j; a\}$) & произведение(суффикс(x, f)) $= g \rightarrow$ произведение(x)(j) $= g(i)$)

Прием применяется без ограничений.

Спецификация приема имеет вид "тип(тела)". Справочник "заголовокприема" указывает уровень срабатывания 2, вводит фильтр "посылка" и указатели "равно(1)", "примечание(ориентацияравенства)", "список(фикс(1 2 1 1))". Этого почти достаточно.

Число приемов данного типа - 10.

6.2. Вывод нечислового равенства для текущего объекта.

Пример:

$\forall_{APfpq}(f = \text{функграфик}(A)$ & функционально(A) & $A = \text{set}_x(\exists_y(x = (p(y), q(y))$ & $P(y))) \rightarrow \text{Val}(f) = \text{set}_x(\exists_y(x = q(y)$ & $P(y))))$

Указатель "контрольвывода" инициирует попытку применения приема при усмотрении в задаче выражения $\text{Val}(f)$, не связанного внешними кванторами и описателями. Первые два антецедента идентифицируются с посылками, третий - выделен указателем "идентификатор".

Спецификация приема имеет вид "тип(трассперечисл)", "терм(t)", где t - текущий объект. Справочник "заголовокприема" указывает уровень срабатывания 2, вводит фильтр "свобоперанд(теквхожд)" и указатели "контрольвывода(Val(f))", "идентификатор(1 3)", "примечание(ориентацияравенства)", "кортежпеременных(y)", "внешнийквантор(фикс(3 2 2))", "отображение(p q P)". Требуется доработка.

Число приемов данного типа - 4.

6.3. Вывод равенства, связывающего между собой атомарные нечисловые объекты.

6.3.1. Вывод равенства, связывающего текущий атомарный объект с другими старыми атомарными объектами.

6.3.1.1. Вывод равенства, выражающего текущий атомарный объект через другие старые атомарные объекты.

Пример:

$\forall_{ABCD}(\text{прямая}(AB) \parallel \text{прямая}(CD)$ & $\text{прямая}(BC) \parallel \text{прямая}(AD)$ & $\text{разныепрямые}(\text{прямая}(AB), \text{прямая}(AD)) \rightarrow \text{вектор}(AB) = \text{вектор}(DC))$

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку применения приема при усмотрении в посылке подвыражения "вектор(AB)". Вектор DC уже рассматривается в задаче.

Спецификация приема имеет вид "тип(извлекается)", "терм(t)", где t - текущий атомарный объект. Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))",

"смактив(терм(вектор(DC)))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(CD)))", "не(равно(C D))", "усм(актив(прямая(BC)))", "не(равно(B C))", "усм(актив(прямая(AD)))", "не(равно(A D))", "не(равно(терм(прямая(AB)) терм(прямая(CD))))", "не(равно(терм(прямая(BC)) терм(прямая(AD))))", "не(равно(терм(прямая(AB)) терм(прямая(AD))))". Создаются также указатели "усм(1 2)", "блокпроверок(3)", "контрольвывода(вектор(AB))". Этого достаточно.

Число приемов данного типа - 6.

6.3.1.2. Вывод равенства, выражающего текущий атомарный объект через атомарный объект "неизв" и известные атомарные объекты.

Пример:

$\forall_{ABC}$(актив(вектор(BC)) $\rightarrow$ вектор(AC) = вектор(AB) + вектор(BC))

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку применения приема при усмотрении в посылке подвыражения "вектор(AC)". Среди векторов AC, AB, BC имеется один известный и один - типа "неизв". Как и в случае числовых атомов, "неизв" означает, что выражение либо содержит неизвестную внешней задачи на описание, либо имеет общее неизвестное атомарное выражение с уравнением, содержащим неизвестную внешней задачи на описание.

Спецификация приема имеет вид "тип(провменьшеилиравно)", "указатель(контрольвывода(t))", где t - текущий атомарный объект. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "конец(числатомы(набор(терм(вектор(AC)) терм(вектор(AB)) терм(вектор(BC))) и(неизв(1) известно(1))))". Создается также указатель "контрольвывода(вектор(AC))". Требуется доработка.

Число приемов данного типа - 4.

6.3.4. Выражение одного нечислового атомарного объекта через другие нечисловые объекты.

6.3.4.1. Выражение одного нечислового атомарного объекта через другие нечисловые атомарные объекты.

Пример:

$\forall_{ABCDEFG}(C$ = проекция(A, плоскость(EFG)) & D = проекция(B, плоскость(EFG)) & актив(вектор(AB)) $\rightarrow$ вектор(CD) = проекция(вектор(AB), плоскость(EFG)))

Прием применяется в посылках задачи на доказательство либо на исследование. Антецеденты идентифицируются с посылками.

Спецификация приема имеет вид "тип(остатокнабора)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))". Требуется доработка.

Число приемов данного типа - 4.

6.3.4.2. Вывод равенства для заданного старого нечислового атомарного объекта.

Пример:

$\forall_{ABCDQp}$(параллелепипед(p) & ребро(отрезок(QA), p) & ребро(отрезок(QB), p) & ребро(отрезок(QC), p) & разныеточки(A, B) & разныеточки(A, C) & разныеточки(B, C) & диагональ(отрезок(QD), p) $\rightarrow$ вектор(QD) = вектор(QA) + вектор(QB) + вектор(QC))

Прием применяется в посылках задачи на доказательство либо на исследование. Вектор QD уже рассматривается в задаче.

Спецификация приема имеет вид "тип(установка)", "терм(t)", где t - старый атомарный объект. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "контекст(посылка(x1) позиция(x2 x1) вид(x2 вектор(QD)))", "не(равно(A B))", "не(равно(A C))", "не(равно(B C))". Этого почти достаточно.

Число приемов данного типа - 3.

6.3.4.4. Вывод равенства, выражающего один старый атомарный объект черех другие старые атомарные объекты.

6.3.4.4.1. Вывод равенства, выражающего один старый атомарный объект через другие старые атомарные объекты.

Пример:

$\forall_{ABCDab}$(актив(вектор(AB)) & актив(вектор(AC)) & актив(вектор(AD)) & $al(BD) = bl(CD)$ & $D \in$ отрезок(BC) & $0 < a + b \rightarrow$ вектор(AD) = $(a/(a + b))$вектор(AB) + $(b/(a + b))$вектор(AC))

Прием применяется в посылках задачи на доказательство либо на исследование. Четвертый антецедент обрабатывается пакетным синтезатором. Расстояния BD, CD уже встречаются в задаче.

Спецификация приема имеет вид "тип(десдробь)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(равно(B C))", "усм(актив($l(BD)$))", "усм(актив($l(CD)$))", "не(равно(B D))", "не(равно(C D))". Этого достаточно.

Число приемов данного типа - 3.

6.3.4.4.3. Выражение неизвестного нечислового атомарного объекта через известные объекты.

6.3.4.4.3.1. Выражение неизвестного нечислового атомарного объекта через известные объекты.

Пример:

$\forall_{ABCDEKab}(K = (C, D, E)$ & координ(вектор$(AB), K) = (0, b)$ & вектор$(AB) = a$ & $\neg(b = 0) \rightarrow$ вектор$(CE) = (1/b)a)$

Прием применяется в посылках задачи на доказательство либо на исследование. Выражения a, b не содержат неизвестных, а выражение "вектор(CE)" - содержит.

Спецификация приема имеет вид "тип(целое)", "терм(t)", где t - неизвестный атомарный объект. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(известно(терм(вектор(CE))))", "известно(a)", "известно(b)". Этого достаточно.

Число приемов данного типа - 3.

6.4. Вывод равенства, дающего явное выражение для функциональной характеристики.

6.4.1. Вывод равенства, дающего явное выражение для функциональной характеристики.

Пример:

$\forall_{AXab}($функраспред$(X, A) = \lambda_x(a(x), x -$ число$)$ & $b(x) = da(x)/dx \rightarrow$
плотнраспред$(X, A) = \lambda_x(b(x), x -$ число$))$

Первый антецедент идентифицируется с посылкой, второй - выделен указателем "идентификатор" и вычисляет производную, так что $b(x)$ символа "производная" уже не содержит. Отсутствует посылка, дающая явное выражение для "плотнраспред(X, A)".

Спецификация приема имеет вид "тип(внимание)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтр "посылка". Он не прорабатывался.

Число приемов данного типа - 10.

7. Вывод двуместного отношения, не являющегося равенством.

Начиная с этого места, возвращаемся к обычной схеме нумерации подпунктов. Глубина вложенности далее будет сравнительно небольшой.

(a) Вывод двуместного отношения в задаче на исследование либо на доказательство.

Пример:

$\forall_{abc}(a \in b$ & $b \subseteq c \rightarrow a \in c)$

Прием применяется без ограничений в посылках задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(свертка)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))". Этого достаточно.

Число приемов данного типа - 416.

(b) Вывод двуместного отношения в задаче на описание, имеющей цель "пример".

Пример:

$\forall_{abc}(a \subseteq \mathbb{R}\ \&\ c \in a\ \&$ верхняягрань$(b, a) \rightarrow c \leq b)$

Прием применяется в задаче на описание, имеющей цель "пример". Антецеденты идентифицируются с посылками.

Спецификация приема имеет вид "тип(сдвигоператоров)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "тип(описать)", "цель(пример)". Этого достаточно.

Число приемов данного типа - 3.

(c) Вывод нечислового двуместного предиката, характеризующего текущий объект.

Пример:

$\forall_a(a \subseteq \mathbb{R}\ \&$ огрснизу$(a) \rightarrow$ нижняягрань$(\inf(a), a))$

Прием применяется в задачах на доказательство, на исследование, а также в задачах на описание, имеющих цель "пример". Попытка его применения инициируется усмотрением выражения $\inf(a)$.

Спецификация приема имеет вид "тип(стандтерм)", "терм(t)", где t - текущий объект. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтр "или(тип(доказать) тип(исследовать) и(тип(описать) цель(пример)))". Создается также указатель "контрольвывода($\inf(a)$)". Этого почти достаточно.

Число приемов данного типа - 10.

(d) Вывод неравенства.

i. Вывод неравенства для невырожденных числовых атомов.

A. Вывод неравенства для невырожденных числовых атомов.
Пример:

$\forall_{ABCD}($ромб$(ABCD)\ \&\ \angle(BAD) < \pi/2 \rightarrow \pi/2 < \angle(ABC))$

Прием применяется в посылках задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(элемент)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(усм(принадлежит(D прямая(AB))))". Требуется доработка.

Число приемов данного типа - 4.

ii. Вывод неравенства для текущего невырожденного числового атома.

Пример:

$\forall_{ab}(a \subseteq \mathbb{R}$ & огрснизу(a) & $b \in a \to 0 \leq b - \inf(a))$

Попытка применения приема инициируется при усмотрении в задаче выражения $\inf(a)$. Первые два антецедента обрабатываются проверочными операторами, последний - идентифицируется с посылкой.

Спецификация приема имеет вид "тип(нок)", "терм(t)", где t - текущий атом. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит указатель "контрольвывода($\inf(a)$)". Этого почти достаточно.

Число приемов данного типа - 6.

iii. Вывод неравенства для численных параметров в задаче на исследование, имеющей цель "известно".

Пример:

$\forall_{ab}(b \leq 0$ & $0 \leq a + b \to 0 \leq a)$

Прием применяется в задачах на исследование, имеющих цель "известно". Второй антецедент идентифицируется с посылкой, первый - обрабатывается проверочным оператором. Выражение a неконстантное и не содержит невырожденных числовых атомов. Все слагаемые выражения b имеют заголовок "минус".

Спецификация приема имеет вид "тип(утверждения)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "тип(исследовать)", "цель(известно)", "неизвестная(a)". Он не прорабатывался.

Число приемов данного типа - 3.

iv. Вывод неравенств в задаче на исследование, не имеющей цели "известно".

A. Вывод неравенства для известных неконстантных численных параметров в задаче на исследование, не имеющей цели "известно".

Пример:

$\forall_{ABa}$(разныеточки(A, B) & $l(AB) = a \to 0 < a)$

Прием применяется в задачах на исследование, не имеющих цели "известно". В случае планиметрических задач это - обычно задачи на построение. Выражение a не содержит неизвестных и неконстантное. В нем нет невырожденных числовых атомов.

Спецификация приема имеет вид "тип(котангенс)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "тип(исследовать)", "не(цель(известно))", "не(константа(a))", "известно(a)", "не(контекст(числовойатом(a x3) не(переменная(x3))))", "не(равно(A B))". Этого почти достаточно.

Число приемов данного типа - 3.

v. Вывод неравенства для выражения с известными числовыми параметрами нечисловых объектов.

A. Вывод неравенства для известного численного параметра нечислового объекта.

Пример:

$\forall_{ABCDabc}(l(AB)ab = l(CD)c \,\&\, 0 \leq c \,\&\, 0 < b \,\&$ разныеточки$(A, B) \rightarrow 0 \leq a)$

Прием применяется в посылках задачи на доказательство либо на исследование. Переменная a идентицируется с известным параметром.

Спецификация приема имеет вид "тип(областьграницы)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "переменная(a)", "известно(a)", "не(равно(A B))". Требуется доработка.

Число приемов данного типа - 3.

8. Вывод многоместного отношения.

(a) Вывод многоместного отношения.
Пример:

$\forall_{ABCDEF}($окружность(EF) вписана в фигура$(ABCD)$ & актив$(\angle(BAD)) \rightarrow$ биссектриса$(BADE))$

Прием применяется в посылках задачи на доказательство либо на исследование. Хотя бы один из углов BAE, DAE уже рассматривается в задаче.

Спецификация приема имеет вид "тип(унификация)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(прямая(AB)))", "усм(актив(прямая(AD)))", "не(усм(принадлежит(D прямая(AB))))". Требуется доработка. Впрочем, во многих случаях особых ограничений на вывод многоместного отношения накладывать не нужно, и тогда практически сразу создаются пригодные для использования приемы.

Число приемов данного типа - 48.

(b) Вывод многоместного отношения, характеризующего текущий объект.

Пример:

$\forall_{ABCD}($прямая$(AB) \perp$ прямая(BC) & прямая$(AB) \parallel$ прямая(CD) & прямая$(BC) \parallel$ прямая$(AD) \rightarrow$ прямоугольник$(ABCD))$

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку применения приема при усмотрении выражения "фигура($ABCD$)".

Спецификация приема имеет вид "тип(клавиатураоператора)", "терм(t)", где t - текущий объект. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(

актив(прямая(BC)))", "не(равно(B C))", "усм(актив(прямая(CD)))", "не(равно(C D))", "усм(актив(прямая(AD)))", "не(равно(A D))", "не(равно(терм(прямая(AB)) терм(прямая(CD))))", "не(равно(терм(прямая(BC)) терм(прямая(AD))))". Кроме того, создается указатель "контрольвывода(фигура(набор($ABCD$)))". Требуется доработка.

Число приемов данного типа - 10.

9. Вывод условия, однозначно определяющего неизвестный объект через известные объекты, но не являющегося равенством.

   Пример:

   $\forall_{ABCDE}$(актив(прямая(AB)) & актив(прямая(CD)) & $E \in$ прямая(AB) & $E \in$ прямая(CD) & разныепрямые(прямая(AB), прямая(CD)) & $E -$ точка $\rightarrow$ $E \in$ прямая(AB) $\cap$ прямая(CD))

   Прием применяется ва задачах на исследование, не имеющих цели "известно". В данном случае - в планиметрических задачах на построение. Переменная E - неизвестная. Выражения "прямая(AB)" и "прямая(CD)" неизвестных не содержат. Выводимая посылка впоследствии будет включена в ответ как определяющая точку E через пересечение двух известных прямых. Прием создает комментарии, указывающие на это обстоятельство.

   Спецификация приема имеет вид "тип(отрицаниеквантора)", "переменная(x)", где x - определяемый объект. Справочник "заголовокприема" указывает уровни срабатывания 2,3,5 и вводит фильтры "посылка", "тип(исследовать)", "не(цель(известно))", "неизвестная(E)", "известно(терм(прямая(AB)))", "известно(терм(прямая(CD)))", "не(равно(терм(прямая(AB)) терм(прямая(CD))))". Создаются также указатели "найдено(E)" и "примечание(найдено переменная(E))". Требуется доработка.

   Число приемов данного типа - 4.

10. Вывод отрицания равенства.

    Пример:

    $\forall_{ABC}(0 < l(AB) - l(AC) \rightarrow \neg(B = C))$

    Прием применяется в посылках задачи на доказательство либо на исследование.

    Спецификация приема имеет вид "тип(первыйсимвол)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтр "посылка". Требуется доработка.

    Число приемов данного типа - 4.

11. Вывод в посылках задачи на доказательство.

    (a) Вывод в посылках задачи на доказательство либо задачи на исследование, имеющей цель "противоречие", ориентированный на исключение переменной.

Пример:

$\forall_{abcdx}(0 < a \ \& \ c < 0 \ \& \ 0 < ax + b \ \& \ 0 < cx + d \to 0 < ad - bc)$

Прием применяется в посылках задачи на доказательство либо задачи на исследование, имеющей цель "противоречие". Два последних антецедента идентифицируются с посылками, два первых - обрабатываются проверочными операторами. Выражение x имеет параметр y, не входящий в выражения a, b, c, d. В случае задачи на доказательство переменная y не входит в условие. Во избежание зацикливания, используется специальный комментарий, устанавливающий приоритеты при исключении переменных.

Спецификация приема имеет вид "тип(объединение)", "переменная(x)", где x - выражение с исключаемой переменной. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) и(тип(исследовать)цель(противоречие)))", "не(контекст(список(x6 a b c d) входит(x5 x6)))", "или(тип(исследовать)контекст(условие(x6) не(входит(x5 x6))))", "не(контекст(примечание(группировки x6) позиция(x7 x6) или(входит(x7 a) входит(x7 b) входит(x7 c)входит(x7 d)) не(заголовок(x7 x5)) не(контекст(разряд(x6 x5 x8) постпозиция(x7 x8)))))", "или(не(заголовок(d 0))и(не(заголовок(x 1))не(заголовок(b 0))))". Указатель "контекст(входит(x5 параметры(x)))" идентифицирует исключаемую переменную x5. Создается также указатель "Примечание(группировки x5)". Этого почти достаточно.

Число приемов данного типа - 3.

(b) Вывод в посылках задачи на доказательство с ограничениями на сложность выводимого утверждения.
$\forall_{abc}(0 < a + b \ \& \ 0 \leq c - b \to 0 < a + c)$

Прием применяется в посылках задачи на доказательство. Выражение b неконстантное. Каждое неконстантное слагаемое какого-то одного из выражений a, c имеет в другом из этих выражений своего двойника, отличающегося лишь константным коэффициентом.

Спецификация приема имеет вид "тип(квадрат)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "тип(доказать)", "не(константа(результат))", "или(не(заголовок(a 0))не(заголовок(c 0)))". Он не прорабатывался.

Число приемов данного типа - 4.

(c) Вывод в посылках задачи на доказательство, ориентированной на сближение с выражениями ее условия.
Пример:

$\forall_{abc}(0 \leq a^2 + b^2 - c \ \& \ 0 \leq a + b \ \& \ 0 \leq c \to 0 \leq a + b - \sqrt{c})$

Прием применяется в посылках задачи на доказательство. Первый антецедент идентифицируется с посылкой. Выражение c константное. В условии встречается сумма, имеющая такие два слагаемые, что a, b суть их натуральные степени (возможно, с показателем 1).

Спецификация приема имеет вид "тип(вводтерма)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "тип(доказать)", "посылка". Он не прорабатывался.

Число приемов данного типа - 8.

Вывод группы отношений

Обычно приемы вывода выводят единственное утверждение. Вывод группы утверждений предпринимается в тех случаях, когда они создают какой-то целостный контекст, ради экономии числа срабатываний.

1. Вывод группы элементарных утверждений в задаче на доказательство либо на исследование.

   Пример:

   $\forall_{ABCD}(C \in \text{отрезок}(AB)\ \&\ D \in \text{отрезок}(AB)\ \&\ l(AC) = l(CD)\ \&\ l(CD) = l(BD)\ \&\ \text{разныеточки}(B,C) \to C \in \text{отрезок}(AD)\ \&\ D \in \text{отрезок}(BC))$

   Прием применяется в посылках задачи на доказательство либо на исследование.

   Спецификация приема имеет вид "тип(второйоперанд)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(равно($A\ B$))", "не(равно($C\ D$))", "не(равно($A\ C$))", "не(равно($B\ C$))", "не(равно($A\ D$))", "не(равно($B\ D$))". Требуется доработка.

   Число приемов данного типа - 23.

2. Вывод группы равенств для числовых атомов.

   (a) Вывод группы равенств для числовых атомов.

   Пример:

   $\forall_{ABCD}(\text{биссектриса}(BACD) \to \angle(CAD) = \angle(DAB)\ \&\ \angle(BAC) = 2\angle(CAD))$

   Прием применяется в посылках задачи на доказательство либо на исследование.

   Спецификация приема имеет вид "тип(оценкатерма)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))". Этого почти достаточно.

   Число приемов данного типа - 5.

Ввод в рассмотрение нового объекта

1. Ввод вспомогательного параметра.

   Пример:

   $\forall_{ABCab}(\angle(ABC) = a\ \&\ \text{прямая}(AB) \perp \text{прямая}(AC) \to b = l(AB))$

Прием применяется в посылках задачи на доказательство либо на исследование. Первый антецедент идентифицируется с посылкой. Расстояние AB встречается в посылке - уравнении, не имеющем вида $l(AB) = \ldots$. Выражение a имеет тип "внешнеизв". Расстояние AB не известно и не имеет типа "внешнеизв". Прием вводит вспомогательный параметр b.

Спецификация приема имеет вид "тип(коммутативно)". Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "контекст(равно(х3 терм($l(AB)$)) не(известно(х3)) не(внешнеизв(х3)))", "усм(актив(прямая(AB)))", "не(равно(A B))", "усм(актив(прямая(AC)))", "не(равно(A C))", "не(усм(принадлежит(C прямая(AB))))". Создается также указатель "вспомпараметр(b фикс(0))". Требуется доработка.

Число приемов данного типа - 24.

2. Ввод вспомогательной неизвестной.

   (a) Ввод вспомогательной неизвестной.

   Пример:

   $\forall_{ABa}$(актив(окружность(AB)) $\rightarrow l(AB) = a$ & $a -$ число)

   Прием применяется в задачах на исследование, имеющих цель "известно". В задаче рассматривается расстояние от точки A до некоторой точки C, не лежащей на окружности AB либо на ее хорде и не являющейся центром окружности, касающейся окружности AB. Это расстояние имеет тип "неизв". Центр окружности A не расположен на выделенных в задаче отрезке, интервале либо прямой. Не указано, что окружность AB вписана в некоторую фигуру. Отсутствует равенство радиуса AB выражению, содержащему только численные параметры. Прием вводит новую переменную a и регистрирует ее как вспомогательную неизвестную.

   Спецификация приема имеет вид "тип(эквивалентно)". Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "тип(исследовать)", "цель(известно)", "не(равно(A B))". Создается также указатель "вспомнеизвестная(a)". Справочник не прорабатывался.

   Число приемов данного типа - 3.

   (b) Ввод вспомогательной неизвестной для текущего объекта.

   Пример:

   $\forall_{ABa}(l(AB) = a$ & $a -$ число)

   Прием применяется в задачах на исследование, имеющих цель "известно". Указатель "контрольвывода" инициирует попытку его применения при усмотрении выражения $l(AB)$ в уравнении. Это уравнение содержит также неизвестную x внешней задачи на описание. Других неизвестных, кроме x и $l(AB)$, в нем нет. Существует еще одно уравнение, содержащее только неизвестные x и $l(AB)$. Отсутствует посылка вида $l(AB) = t$, такая, что t не имеет невырожденных числовых атомов. Прием вводит новую переменную a и регистрирует ее как вспомогательную неизвестную.

Спецификация приема имеет вид "тип(удалениепосылок)". Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "тип(
исследовать)", "цель(известно)", "заголовок(корень равно)", "не(контекст(неизвестная(x2) разряд(корень x2 x3)операнд(x4 x3)не(контекст(вид(x4 $l(AB)$))))))", "конец(не(контекст(посылка(x2) вид(x2 равно($l(AB)$ x3)) не(контекст(числовойатом(x3 x4) не(переменная(x4)))))))", "не(равно($A\ B$))". Создаются также указатели "вспомнеизвестная(a)" и "контрольвывода($l(AB)$)". Справочник не прорабатывался.

Число приемов данного типа - 11.

3. Ввод вспомогательных объектов.

   Пример:

   $\forall_{ABCD}(l(AB) = l(AC)$ & актив(прямая(BC)) & разныеточки(B, C) $\rightarrow$
   $D-$точка & $D \in$ отрезок(BC) & прямая(AD) $\perp$ прямая(BC) & $l(BD) = l(DC)$)

   Прием применяется в посылках задачи на доказательство либо на исследование. Вводится новая точка D - основание высоты равнобедренного треугольника ABC. Предварительно проверяется, что расстояние AB выражено через численные параметры и что в задаче рассматривается расстояние от точки A до некоторой точки, не лежащей на прямых AB, AC, BC. Проверяется также, что треугольник ABC не прямоугольный и что точка A не лежит на прямой BC.

   Спецификация приема имеет вид "тип(Минус)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "не(тип(
   преобразовать))", "или(не(тип(описать))не(цель(прямойответ))не(контекст(неизвестная(x1) не(контекст(ключ(цели параметры x2) входит(x1 x2))))))", "не(равно($B\ C$))", "не(равно($B\ D$))", "не(равно($C\ D$))". Он не прорабатывался.

   Число приемов данного типа - 200. Фактически, это все приемы для "дополнительного построения". Мотивация обычно индивидуальная и определяемая еще на этапе программирующего вывода.

4. Ввод вспомогательного обозначения.

   (a) Ввод вспомогательного обозначения для текущего подвыражения.

   Пример:

   $\forall_{fG}$(операция(G) = f)

   Прием применяется в задачах на доказательство. Указатель "контрольвывода" инициирует попытку его применения при усмотрении в задаче выражения "операция(G)", где G - группа. Выбирается новая переменная f, которая будет использоваться как обозначение для операции данной группы. Комментарий "заменяемое"к выводимому равенству блокирует попытки его переориентации и исключения.

Спецификация приема имеет вид "тип(нормтаблица)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтр "не(контекст(посылка(x1) вид(x1 равно(операция(G)x2)) переменная(x2)))". Создается также указатель "контрольвывода(операция(G))". Справочник не прорабатывался.

Число приемов данного типа - 5.

Регистрация объекта в активе

1. Регистрация рассматриваемого объекта в активе.

   Пример:

   $\forall_{AB}$(актив(прямая(AB)))

   Прием применяется в задачах на доказательство, на исследование, а также в задачах на преобразование, имеющих цель "класс". Указатель "контрольвывода" инициирует попытку его применения при усмотрении выражения "прямая(AB)".

   Спецификация приема имеет вид "тип(занесениепосылки)", "терм(t)", где t - рассматриваемый объект. Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "или(тип(доказать)тип(исследовать)и(тип(преобразовать)цель(класс)))", "не(заголовок(корень актив))", "не(См(прямая(AB)))", "свобоперанд(теквхожд)". Создается также указатель "контрольвывода(прямая(AB))". Этого достаточно.

   Число приемов данного типа - 7.

2. Регистрация в активе нового объекта.

   Пример:

   $\forall_{AB}$(актив($l(AB)$) $\rightarrow$ актив(прямая(AB)))

   Прием применяется в задачах на доказательство, на исследование, а также в задачах на преобразование, имеющих цель "класс".

   Спецификация приема имеет вид "тип(Контрользамены)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(контекст(усм(актив(прямая(AB)))))". Он не прорабатывался.

   Число приемов данного типа - 54.

3. Регистрация в активе нового объекта, связанного с текущим подтермом.

   Пример:

   $\forall_{ABCD}$(параллелограмм($ABCD$) $\rightarrow$ актив($l(AB)$))

   Прием применяется в задачах на доказательство, на исследование, а также в задачах на преобразование, имеющих цель "класс". Указатель "контрольвывода" инициирует попытку его применения при усмотрении выражения "площадь(фигура(набор($ABCD$)))". Расстояние AB в задаче пока не рассматривается.

Спецификация приема имеет вид "тип(измпозиции)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать)и(тип(преобразовать)цель(класс)))", "свобоперанд(теквхожд)". Он не прорабатывался.

Число приемов данного типа - 26.

Вывод дизъюнкции для разбора случаев

1. Разбор случаев в посылках.

   Пример:

   $\forall_{ABCDEF}$(актив($\angle(AEF)$) & актив($\angle(EFD)$) & прямая(AB) $\parallel$ прямая(CD) & $E \in$ прямая(AB) & $F \in$ прямая(CD) $\rightarrow$ однасторона($A, D,$ прямая(EF)) $\vee$ разныестороны($A, D,$ прямая(EF)))

   Прием применяется в посылках задачи на доказательство либо на исследование. Угол EFD имеет тип "неизв". Ни одна из выводимых альтернатив проверочными операторами не усматривается.

   Спецификация приема имеет вид "тип(проверка)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "усм(актив(прямая(AE)))", "усм(актив(прямая(EF)))", "усм(актив(прямая(DF)))", "усм(актив(прямая(AB)))", "не(равно($A\ B$))", "усм(актив(прямая(CD)))", "не(равно($C\ D$))", "не(усм(принадлежит(F прямая(AE))))", "не(усм(принадлежит(D прямая(EF))))". Создается также указатель "примечание(разборслучаев)". Справочник не прорабатывался.

   Число приемов данного типа - 61.

2. Разбор случаев в посылках задачи на доказательство, связанный с рассмотрением текущего подвыражения ее условия.

   Пример:

   $\forall_a(a \leq 0 \ \vee \ 0 < a)$

   Прием применяется в задачах на доказательство. Попытка его применения инициируется усмотрением в условии задачи выражения "модуль(a)". Это условие представляет собой неравенство, причем выбранный модуль - самый короткий из имеющихся в условии.

   Спецификация приема имеет вид "тип(конъюнктчлен)", "терм(t)", где t - текущее подвыражение условия. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "тип(доказать)", "условие". Создаются также указатели "контрольвывода(модуль(a))" и "примечание(разборслучаев)". Справочник не прорабатывался.

   Число приемов данного типа - 4.

3. Разбор случаев в посылках задачи на преобразование, связанный с рассмотрением текущего подвыражения ее условия.

Пример:

$\forall_{ab}(0 \le ab \to 0 \le a \ \& \ 0 \le b \ \vee \ a \le 0 \ \& \ b \le 0)$

Прием применяется в задачах на преобразование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении в условии подвыражения "$(ab)^c$", расположенного внутри радикала четной степени.

Спецификация приема имеет вид "тип(внешантецедент)", "терм(t)", где t - текущее подвыражение условия. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "тип(преобразовать)", "условие". Создаются также указатели "контрольвывода($(ab)^c$)" и "примечание(разборслучаев)". Справочник не прорабатывался.

Число приемов данного типа - 7.

Контроль противоречивости при разборе случаев

1. Усмотрение противоречия в посылках задачи на исследование, возникшей при разборе случаев.

   Пример:

   $\forall_{ABCDE}(\neg(E \in \text{фигура}(ABC)) \ \& \ D \in \text{отрезок}(AC) \ \& \ E \in \text{отрезок}(BD) \to \text{ложь})$

   Прием применяется в задачах на исследование, имеющих цель "контроль".

   Спецификация приема имеет вид "тип(известны)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(исследовать)", "цель(контроль)", "не(равно(B D))", "не(равно(A D))", "не(равно(C D))", "не(равно(B E))", "не(равно(D E))". Этого почти достаточно.

   Число приемов данного типа - 45.

Вывод утверждения существования

1. Вывод утверждения существования.

   Пример:

   $\forall_a(\text{огрсверху}(a) \to \exists_b(\text{верхняягрань}(b, a)))$

   Прием применяется в задачах, не имеющих типа "преобразовать". Антецедент идентифицируется с посылкой. Есть ряд ограничений, связанных с типом задачи и ее целевой установкой.

   Спецификация приема имеет вид "тип(продолжение)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "не(тип(преобразовать))", "или(не(тип(исследовать)) и(не(цель(Посылка)) не(цель(известно))))", "или(не(тип(описать)) не(цель(прямойответ))не(контекст(неизвестная(x3) не(контекст(ключ(цели параметры x4) входит(x3 x4))))))", "не(контекст(посылка(x4) вид(x4 верхняягрань(x3 a))))". Требуется доработка.

   Число приемов данного типа - 9.

Вывод кванторной импликации

1. Вывод кванторной импликации, определяющей значения серии невырожденных числовых атомов.

   Пример:

   $\forall_{AQnp}(\forall_i(i \in \{1,\ldots,n\} \to \text{услвероятн}(A(i), \bigcap_{j=1}^{i-1} A(j), Q) = p(i)) \to$
   $\forall_i(i \in \{1,\ldots,n\} \to \text{вероятность}(A(i), Q) = \prod_{j=1}^{i} p(j)))$

   Прием применяется в задачах на доказательство, исследование либо в задачах на описание, имеющих цель "пример" либо "известно ...". Антецедент идентифицируется с кванторной импликацией.

   Спецификация приема имеет вид "тип(решить)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать) и*тип(описать) или(цель(пример) Входит(известно цели)))". Создаются также указатели "внешнийквантор(фикс(1))"и "отображение(p, A)". Требуется доработка.

   Число приемов данного типа - 4.

2. Вывод кванторного тождества с целью преобразования его в рекуррентное соотношение.

   Пример:

   $\forall_{ABCabm}(\forall_n(n-\text{натуральное} \to \det(\lambda_{ij}(A(i,j), i \in \{1,\ldots,an+b\}\ \&\ j \in \{1,\ldots,an+b\})) = B(n))\ \&\ \text{set}_i(i \in \{1,\ldots,an+b\}\ \&\ \neg(A(1,i) = 0)) = \{;C\}\ \&\ l(C) = m \to \forall_n(n - \text{натуральное}\ \&\ 2 \le n \to B(n) = \sum_{k=1}^{m}(A(1,C(k))(-1)^{C(k)+1}\det(\lambda_{ij}((A(i+1,j)\ \text{при}\ j < C(k),\ \text{иначе}A(i+1,j+1)), i \in \{1,\ldots,an+b-1\}\ \&\ j \in \{1,\ldots,an+b-1\})))))$

   Прием применяется в задачах на преобразование. Первый антецедент идентифицируется с посылкой, два других выделены указателем "идентификатор". a, m - натуральные константы; b - целочисленная константа. Конечная сумма разворачивается в обычную.

   Спецификация приема имеет вид "тип(уровень)". Справочник "заголовокприема" указывает уровень срабатывания 7 и вводит фильтры "посылка", "тип(преобразовать)". Он не прорабатывался.

   Число приемов данного типа - 3.

Вывод при специальных целевых установках

Теоремы приемов таких типов часто возникают на этапе программирующего логического вывода как квазипротоколы, и уже на этом этапе имеется вся информация о том, какой прием должен по ним создаваться. Компилятор спецификаций в этих случаях играет роль передаточного звена.

1. Целевой вывод, инициируемый посылкой.

   Пример:

   $\forall_{abcdfguv}(g = \lambda_x(u(x), v(x))$ & Производная(f, g) & $a = \mathrm{set}_x(u(x) = 0$ & $x \in b) \rightarrow$ roots$(g, b) = a)$

   Прием применяется в посылках задачи на доказательство, имеющей условие вида "нижняягрань$(X,$ образ$(f, b))$" либо "Нижняягрань$(X,$ образ$(f, b))$". Такие задачи возникают при доказательстве неравенств с помощью производных. Первые два антецедента идентифицируются с посылками, третий выделен указателем "идентификатор". Уравнение в его правой части разрешается относительно x задачей на описание.

   Спецификация приема имеет вид "тип(См)". Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1. Он не прорабатывался.

   Число приемов данного типа - 109.

2. Целевой вывод, инициируемый текущим подвыражением условия.

   Пример:

   $\forall_{abcf}$(собствекторы(f, a, b) & ортогонализация$(b, c) \rightarrow$ ортогонализация$(b, c))$

   Прием применяется в задачах на описание. Указатель "контрольвывода" инициирует попытку его применения при усмотрении утверждения "ортканоничвид(f, x, y)" в условии задачи. Переменные x, y - неизвестные. Первый антецедент идентифицируется с посылкой, второй - обрабатывается синтезатором, выполняющим ортогонализацию системы векторов. Напомним, что "ортканоничвид(f, x, y)" означает, что y - результат приведения ортогональным преобразованием x2 квадратичной формы f над полем вещественных чисел к каноническому виду.

   Спецификация приема имеет вид "тип(ослабление)", "терм(t)", где t - текущее подвыражение. Справочник "заголовокприема" ограничивается указанием уровня срабатывания 1, вводом фильтра "условие" и указателя "контрольвывода(ортканоничвид(f, x, y))". Он не прорабатывался.

   Число приемов данного типа - 7.

3. Вывод следствий в задаче на исследование, имеющей цель "исключ".

   Пример:

   $\forall_{abc}(a \subseteq b$ & $b \subseteq c \rightarrow a \subseteq c)$

   Прием применяется в задачах на исследование, имеющих цель "исключ ...". Выражение b имеет переменные, указанные в этой цели, а выражения a, c - не имеют.

Спецификация приема имеет вид "тип(примечанализатор)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(исследовать)", "Входит(исключ цели)", "контекст(ключ(цели исключ x4) не(пересекаются(параметры(a)x4)) не(пересекаются(параметры(c)x4)) пересекаются(параметры(b)x4))". Этого достаточно.

Число приемов данного типа - 8.

4. Использование синтезаторов для устранения зависимости от исключаемых переменных в задаче на исследование, имеющей цели "длялюбого", "независит".

   Пример:

   $\forall_{abc}(a < b \ \& \ c \leq a \rightarrow c < b)$

   Прием применяется в задачах на исследование, имеющих цель "длялюбого"и цель "независит ...". Первый антецедент идентифицируется с посылкой, второй - обрабатывается пакетным синтезатором, которому передается в качестве комментария цель "независит ...". Выражение a содержит переменные, указанные в этой цели. Выражение b содержит неизвестные.

   Спецификация приема имеет вид "тип(точкиотрезка)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "тип(исследовать)", "цель(длялюбого)", "Входит(независит цели)". Он не прорабатывался.

   Число приемов данного типа - 4.

5. Вывод в задаче на описание, имеющей цель "независит", подготавливающий устранение зависимости текущего подвыражения условия от исключаемых переменных.

   Пример:

   $\forall_{abcdef}(0 \leq a + bc \ \& \ 0 \leq d + ef \ \& \ 0 < b \ \& \ 0 < e \rightarrow 0 \leq bec + bef + ae + bd)$

   Прием применяется в задачах на описание, имеющих цель "независит ...". Указатель "контрольвывода" инициирует попытку его применения при усмотрении в условии подвыражения $c + f$. Первые два антецедента идентифицируются с посылками, следующие два - обрабатываются проверочными операторами. Выражения a, b, d, e не содержат переменных, указанных в цели "независит ...", а выражения c, f - содержат. Выводима посылка позволяет получить неравенство для $c + f$ и воспользоваться им для обратного вывода, исключающего "запрещенные" перменные в условии.

   Спецификация приема имеет вид "тип(сокращение)", "терм(t), где t - текущий терм. Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "условие", "тип(описать)", "Входит(независит цели)", "или(не(заголовок(d 0)) не(заголовок(f 0)) не(заголовок(e 0))и(не(заголовок(c 0))не(заголовок(a 0))))", "или(не(заголовок(a 0)) не(заголовок(c 0)))". Создается также указатель "контрольвывода($c + f$)". Справочник не прорабатывался.

   Число приемов данного типа - 4.

Вывод определения

Пример:

$\forall_a(a \subseteq \mathbb{R}$ & огрмнож$(a) \to \exists_b(b -$ число & $\forall_x(x \in a \to |x| \leq b)))$

Прием применяется в задачах на доказательство, на исследование, либо в задачах на описание, имеющих цель "пример".

Спецификация приема имеет вид "тип(нормуглы)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать) и(тип(описать) цель(пример)))". Этого почти достаточно.

Число приемов данного типа - 24.

Координаты

1. Связь числовых атомов с координатами.

   (a) Связь числовых атомов с координатами.

   Пример:

   $\forall_{ABCDEKab}(K = (A, B, C)$ & коорд$(D, K) = (a, b)$ & $E \in$ прямая(AB) & прямая$(AC) \parallel$ прямая(DE) & $0 \leq b \to bl(AC) = l(DE))$

   Прием применяется в посылках задачи на доказательство либо на исследование. Первые два антецедента идентифицируются с посылками, третий и четвертый - выделены указателем "усм".

   Спецификация приема имеет вид "тип(формоперанды)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "усм(актив(прямая(AB)))", "не(равно$(A\ B)$)", "усм(актив(прямая(AC)))", "не(равно$(A\ C)$)", "усм(актив(прямая(DE)))", "не(равно$(D\ E)$)", "не(равно(терм(прямая(AC))) терм(прямая(DE))))". Этого почти достаточно.

   Число приемов данного типа - 9.

   (b) Связь старых числовых атомов с координатами.

   Пример:

   $\forall_{ABKabc}($прямкоорд(K) & коорд(вектор$(AB), K) = (a, b, c)$ & актив$(l(AB)) \to l(AB) = \sqrt{a^2 + b^2 + c^2})$

   Прием применяется в посылках задачи на доказательство либо на исследование. Выводимое равенство содержит неизвестные.

   Спецификация приема имеет вид "тип(Входит)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(известно(результат))", "не(равно$(A\ B)$)". Этого почти достаточно.

   Число приемов данного типа - 8.

2. Связь текущего числового атома с координатами.

(a) Связь текущего числового атома с координатами.

Пример:

$\forall_{ABKabc}$(прямкоорд(K) & коорд(вектор(AB), K) = $(a, b, c) \rightarrow$
$l(AB) = \sqrt{a^2 + b^2 + c^2}$)

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении в посылке выражения "$l(AB)$". Второй антецедент выделен указателем "идентификатор", его левая часть обрабатывается нормализатором "нормкоорд". Проверяется, что в задаче уже рассматривается вектор AB.

Спецификация приема имеет вид "тип(Операнды)", "терм(t), где t - текущий числовой атом. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(корень актив))", "не(равно(A B))". Создается также указатель "контрольвывода($l(AB)$)". Требуется доработка.

Число приемов данного типа - 36.

(b) Связь текущего числового атома с координатами, выраженными через численные параметры.

Пример:

$\forall_{Kabcdefgh}$(прямкоорд(K) & коорд(a, K) = (b, c, g) & коорд(d, K) = $(e, f, h) \rightarrow$
скалумнож(a, d) = $be + cf + gh$)

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении в посылке выражения "скалумнож(a, d)". Выражения b, c, e, f, g, h не содержат невырожденных числовых атомов. Два последних антецедента выделены указателем "идентификатор".

Спецификация приема имеет вид "тип(преобрфильтр)", "терм(t), где t - текущий числовой атом. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(контекст(список(x9 b e c f g h) числовойатом(x9 x10) не(переменная(x10))))". Создается также указатель "контрольвывода(скалумнож(a, d))". Этого почти достаточно.

Число приемов данного типа - 3.

(c) Определение текущего числового атома "неизв" через известные коорината.

Пример:

$\forall_{ABCKabcdpq}$(прямкоорд(K) & коорд(вектор(AB), K) = (a, b) &
коорд(вектор(AC), K) = $(c, d) \rightarrow \cos(\angle(BAC)) =$
$(ac + bd)/(\sqrt{a^2 + b^2} \cdot \sqrt{c^2 + d^2})$)

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения

при усмотрении в посылке выражения "$\angle(BAC)$", причем угол BAC имеет тип "неизв". Выражения a, b, c, d известны. Два последних антецедента выделены указателем "идентификатор".

Спецификация приема имеет вид "тип(полныепосылки)", "терм(t), где t - текущий числовой атом. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "неизв(терм(угол(BAC)))", "известно(a)", "известно(c)", "известно(b)","известно(d)". Создается также указатель "контрольвывода(угол(BAC))". Этого почти достаточно.

Число приемов данного типа - 6.

3. Вывод соотношений для координат объектов.

   (a) Вывод соотношений для координат объектов.

   Пример:

   $\forall_{ABCDEKabcd}(K = (A, B, C)$ & коорд$(D, K) = (a, b)$ & коорд$(E, K) = (c, d)$ & прямая$(DE) \parallel$ прямая$(AC) \to a = c)$

   Прием применяется в посылках задачи на доказательство либо на исследование.

   Спецификация приема имеет вид "тип(больше)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(константа(результат))", "усм(актив(прямая(DE)))", "не(равно($D\ E$))", "усм(актив(прямая(AC)))", "не(равно($A\ C$))", "не(равно(терм(прямая(DE))терм(прямая(AC))))". Этого почти достаточно.

   Число приемов данного типа - 39.

   (b) Вывод соотношения, связывающего известные координаты заданных объектов с неизвестными координатами других объектов.

   Пример:

   $\forall_{ABCKabcdxy}($коорд$(A, K) = (a, b)$ & коорд$(B, K) = (c, d)$ & $C \in$ прямая(AB) & коорд$(C, K) = (x, y) \to (c - a)(y - b) - (d - b)(x - a) = 0)$

   Прием применяется в посылках задачи на доказательство либо на исследование. Выражения a, b, c, d не содержат неизвестных, а хотя бы одно из выражений x, y - содержит. Все антецеденты, кроме третьего, идентифицируются с посылками.

   Спецификация приема имеет вид "тип(конъюнктоперанд)", "известно($a_1 \ldots a_k$)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "известно(a)", "известно(b)", "известно(c)", "известно(d)", "или(не(известно(x)) не(известно(y)))", "усм(актив(прямая(AB)))", "не(равно($A\ B$))". Этого почти достаточно.

   Число приемов данного типа - 3.

4. Выражение координатных наборов через числовые атомы.

   (a) Выражение координатных наборов через числовые атомы.

   Пример:

   $\forall_{ABCDEK}(K = (A, B, C, D, E)$ & $D \in$ прямая(AB) & точкалуча(A, B, D) & прямкоорд$(K) \rightarrow$ коорд$(D, K) = (l(AD), 0, 0))$

   Прием применяется в посылках задачи на доказательство либо на исследование. Второй и третий антецеденты выделены указателем "усм". Расстояние AD выражено через численные параметры. Координатный набор для коорд(D, K) пока не определен.

   Спецификация приема имеет вид "тип(удаление)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(фикс(0 1)набор))", "усм(актив(прямая(AB)))", "не(равно(A B))", "не(равно(A D))", "не(равно(B D))". Требуется доработка.

   Число приемов данного типа - 7.

   (b) Выражение координатных наборов через численные параметры.

      i. Выражение координатных наборов через численные параметры.

      Пример:

      $\forall_{ABCKabcdpq}($коорд$(A, K) = (a, b)$ & коорд$(C, K) = (c, d)$ & $B \in$ отрезок(AC) & $pl(AB) = ql(BC)$ & $\neg(p + q = 0) \rightarrow$ коорд$(B, K) = ((ap + cq)/(p + q), (bp + dq)/(p + q)))$

      Прием применяется в посылках задачи на доказательство либо на исследование. Первые два антецедента идентифицируются с посылками, четвертый - обрабатывается пакетным синтезатором. Выражения a, b, c, d, p, q не содержат невырожденных числовых атомов.

      Спецификация приема имеет вид "тип(стандравно)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(контекст(список(х5 a p c q b d) числовойатом(х5 х6) не(переменная(х6))))", "не(равно(A B))", "не(равно(B C))". Этого почти достаточно.

      Число приемов данного типа - 23.

      ii. Выражение координатных наборов через известные параметры.

      Пример:

      $\forall_{abcdefghijklmABK}(\neg(k = 0)$ & коорд$(l, K) = (c, d)$ & коорд$(l, K) = (a, b)$ & коорд(вектор$(lm), K) = (g, h)$ & коорд(вектор$(AB), K) = (e, f)$ & $j \in$ прямая(lm) & $j \in$ прямая(AB) & $i = g(b - d) + h(c - a)$ & $k = eh - fg \rightarrow$ коорд$(j, K) = (a + ei/k, b + fi/k))$

      Прием применяется в посылках задачи на доказательство либо на исследование. Второй и третий антецеденты идентифицируются с посылками; четвертый, пятый, восьмой и девятый выделены указателем "идентификатор". Выражения a, b, e, f, i, k не содержат неизвестных.

Спецификация приема имеет вид "тип(Синус)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(фикс(0 1)набор))", "известно(a)", "известно(e)", "известно(i)", "известно(k)", "известно(b)", "известно(f)", "усм(актив(прямая(lm)))", "не(равно(l m))", "усм(актив(прямая(AB)))", "не(равно(A B))". Этого почти достаточно.

Число приемов данного типа - 3.

5. Выражение текущего координатного набора через числовые атомы.

   (a) Выражение текущего координатного набора через числовые атомы.

   Пример:

   $\forall_{ABCDEFK}(K = (A, B, C)$ & $E \in$ прямая(AB) & точкалуча(A, B, E) & прямая$(DE) \parallel$ прямая(AC) & $F \in$ прямая(AC) & точкалуча(A, C, F) & прямая$(DF) \parallel$ прямая$(AB) \rightarrow$ коорд$(D, K) = (l(AE)/l(AB), l(AF)/l(AC)$ $))$

   Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку применения приема при усмотрении выражения "коорд(D, K)".

   Спецификация приема имеет вид "тип(регистрациячертежа)", "терм(t)", где t - текущий координатный набор. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(фикс(0 1)набор))", "усм(актив(прямая(AB)))", "не(равно(A B))", "не(равно(A E))", "не(равно(B E))", "усм(актив(прямая(DE)))","не(равно(D E))","усм(актив(прямая(AC)))", "не(равно(A C))", "не(равно(A F))", "не(равно(C F))", "усм(актив(прямая(DF)))", "не(равно(D F))", "не(равно(терм(прямая(DE)) терм(прямая(AC))))", "не(равно(терм(прямая(DF)) терм(прямая(AB))))". Создается также указатель "контрольвывода(коорд(D, K))". Этого достаточно.

   Число приемов данного типа - 4.

   (b) Выражение текущего координатного набора через численные параметры.

      i. Выражение текущего координатного набора через численные параметры.

      Пример:

      $\forall_{ABK}($коорд(вектор$(AB), K) = (a, b, c)$ & коорд$(A, K) = (d, e, f) \rightarrow$ коорд$(B, K) = (a + d, b + e, c + f))$

      Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку применения приема при усмотрении выражения "коорд(B, K)". Первый антецедент идентифицируется с посылкой, второй - выделен указателем "идентификатор". Выражения a, b, c, d, e, f не содержат невырожденных числовых атомов.

Спецификация приема имеет вид "тип(регистрациячертежа)", "терм(t)", где t - текущий координатный набор. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "конец(не(контекст(список(x7 a d b e c f) числовойатом(x7 x8) не(переменная(x8)))))". Создаются также указатели "идентификатор(2)" и "контрольвывода(коорд(B, K))".

Число приемов данного типа - 5.

ii. Выражение текущего координатного набора через известные параметры.

Пример:

$\forall_{ABCDKabcdpquv}$(прямая(CD) $\parallel$ прямая(AB) & однасторона(B, D, прямая(AC)) & коорд(вектор(AB), K) = (a, b) & $pl(AB) = ql(CD)$ & коорд(C, K) & $\neg(q = 0)$ & коорд(вектор(AC), K) = (u, v) & $\neg(bu - av = 0) \rightarrow$ коорд(D, K) = $(c + ap/q, d + bp/q)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку применения приема при усмотрении выражения "коорд(D, K)". Третий, пятый и седьмой антецеденты выделены указателем "идентификатор". Четвертый антецедент обрабатывается пакетным синтезатором. Выражения a, b, c, d, p, q не содержат неизвестных.

Спецификация приема имеет вид "тип(факториал)", "терм(t)", где t - текущий координатный набор. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(фикс(0 1)набор))", "известно(c)", "известно(a)", "известно(p)", "известно(q)", "известно(d)", "известно(b)", "усм(актив(прямая(CD)))", "не(равно(C D))", "усм(актив(прямая(AB)))", "не(равно(A B))", "не(равно(терм(прямая(CD)) терм(прямая(AB))))". Создается также указатель "контрольвывода(коорд(D, K))". Этого достаточно.

Число приемов данного типа - 4.

6. Выражение связанного с текущим термом кординатного набора через числовые атомы.

   (a) Выражение связанного с текущим термом кординатного набора через численные параметры.

      i. Выражение связанного с текущим термом кординатного набора через численные параметры.

      Пример:

      $\forall_{ABK}$(коорд(вектор(AB), K) = (a, b, c) & коорд(A, K) = $(d, e, f) \rightarrow$ коорд(B, K) = $(a + d, b + e, c + f)$)

      Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку применения приема при усмотрении выражения "сферкоорд(B, K)". Первый

антецедент идентифицируется с посылкой, второй - выделен указателем "идентификатор". Выражения a, b, c, d, e, f не содержат невырожденных числовых атомов.

Спецификация приема имеет вид "тип(числосочетаний)", "терм(t)", где t - текущий терм. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(заголовок(фикс(0 1)набор))", "конец(не(контекст(список(x7 $a\ d\ b\ e\ c\ f$) числовойатом(x7 x8) не(переменная(x8)))))". Создаются также указатели "идентификатор(2)" и "контрольвывода(сферкоорд(B, K))". Этого достаточно.

Число приемов данного типа - 12.

ii. Выражение связанного с текущим термом кординатного набора через известные параметры.

Пример:

$\forall_{abpKT}$(вниз(Скорость$(a, K, T), K)$ & Равндвиж$(a, T) \rightarrow$
координапрпути(Путь$(a, T)), K) = (0, 0, -1)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку применения приема при усмотрении выражения "напрпути(Путь(a, T))". Антецеденты идентифицируются с посылками.

Спецификация приема имеет вид "тип(поисктеор)", "терм(t)", где t - текущий терм. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(фикс(0 1)набор))". Создаются также указатели "блокпроверок(2)" и "контрольвывода(напрпути(Путь(a, T)))". Этого почти достаточно.

Число приемов данного типа - 8.

7. Вывод соотношений для отдельных координат и числовых атомов.

(a) Вывод соотношений для отдельных координат и числовых атомов.

Пример:

$\forall_{ABK}$(вправо(вектор$(AB), K) \rightarrow$ крд$(A, K, 3) =$ крд$(B, K, 3)$)

Прием применяется в посылках задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(текпосылка)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))". Этого почти достаточно.

Число приемов данного типа - 9.

(b) Вывод соотношений для отдельных координат и старых числовых атомов.

Пример:

$\forall_{ABK}$(вправо(вектор$(AB), K) \rightarrow$ крд$(B, K, 1) =$ крд$(A, K, 1) + l(AB)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояние AB уже рассматривается в задаче.

Спецификация приема имеет вид "тип(Разныестороны)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "усм(актив($l(AB)$))", "не(равно(A B))". Этого почти достаточно.

Число приемов данного типа - 11.

8. Вывод соотношения, связывающего текущую координату с другими координатами и числовыми атомами.

   (a) Вывод соотношения, связывающего текущую координату с другими координатами и числовыми атомами.

   Пример:

   $\forall_{ABCDKPQ}$(прямая(PQ) $\parallel$ плоскость(ABD) & $K = (A, B, C, D)$ & прямкоорд(K) $\rightarrow$ крд($Q, K, 2$) = крд($P, K, 2$))

   Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку применения приема при рассмотрении подвыражения "крд($Q, K, 2$)".

   Спецификация приема имеет вид "тип(коррекциязнака)", "терм(t)", где t - текущая координата. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "усм(актив(прямая(PQ)))", "не(равно(B D))", "не(равно(A D))не(равно(A B))". Создается также указатель "контрольвывода(крд(Q, $K, 2$))". Этого почти достаточно.

   Число приемов данного типа - 16.

   (b) Вывод соотношения, связывающего текущую координату со старымии координатами и числовыми атомами.

   Пример:

   $\forall_{ABK}$(прямкоорд(K) & вверх(вектор(BA), K) $\rightarrow$ $l(AB)$ = крд($A, K, 3$) $-$ крд($B, K, 3$))

   Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку применения приема при рассмотрении подвыражения "крд($B, K, 3$)". Выражения $l(AB)$, "крд($A, K, 3$)" уже встречаются в посылках.

   Спецификация приема имеет вид "тип(дуга)", "терм(t)", где t - текущая координата. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "усм(актив($l(AB)$))", "контекст(посылка(x1) позиция(x2 x1) вид(x2 крд($A, K, 3$)))", "не(равно(A B))". Создается также указатель "контрольвывода(крд($B, K, 3$))".Этого почти достаточно.

   Число приемов данного типа - 26.

9. Вывод ограничений на координаты объектов.

   (a) Вывод ограничений на координаты объектов.

   Пример:

   $\forall_{ABKabcdef}$(прямкоорд(K) & Вектор(A) & Вектор(B) & коорд$(A,K) = (a,b,c)$ & коорд$(B,K) = (d,e,f)$ & уголмежду$(A,B) < \pi/2 \rightarrow 0 < ad + be + cf$)

   Прием применяется в посылках задачи на доказательство либо на исследование. Четвертый и пятый антецеденты выделены указателем "идентификатор".

   Спецификация приема имеет вид "тип(наборслов)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))". Этого почти достаточно.

   Число приемов данного типа - 5.

   (b) Вывод ограничений на координаты объектов при контроле разбора случаев в задаче на исследование.

   Пример:

   $\forall_{ABCKabcdef}(B \in$ отрезок(AC) & коорд$(A,K) = (a,b)$ & коорд$(B,K) = (c,d)$ & коорд$(C,K) = (e,f) \rightarrow 0 \leq (c-a)(e-c) + (d-b)(f-d))$

   Прием применяется в задачах на исследование, имеющих цель "контроль". Антецеденты идентифицируются с посылками.

   Спецификация приема имеет вид "тип(дн)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "посылка", "тип(исследовать)", "цель(контроль)", "не(равно($A\ B$))", "не(равно($B\ C$)). Требуется доработка.

   Число приемов данного типа - 3.

10. Ввод нового объекта для вычисления координат.

    (a) Ввод новых объектов и задание их координатных наборов через старые параметры.

    Пример:

    $\forall_{ABCDEFG}$(ромб$(ABCD)$ & $E \in$ прямая(AC) & $E \in$ прямая(BD) & коорд$(E,K) = (a,b)$ & $F \in$ прямая(AB) & коорд$(F,K) = (c,d) \rightarrow G -$ точка & $G \in$ прямая(CD) & коорд$(G,K) = (2a-c, 2b-d)$)

    Прием применяется в посылках задачи на доказательство либо на исследование. Первый и четвертый антецеденты идентифицируются с посылками, шестой - выделен указателем "идентификатор". Выражения a, b, c, d не содержат неизвестных. Проверяется также наличие на прямой CD точки с известными координатами. Прием вводит новую точку G.

Спецификация приема имеет вид "тип(натуральное)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "усм(актив(прямая(A C)))", "не(равно(A C))", "усм(актив(прямая(B D)))", "не(равно(B D))", "усм(актив(прямая(A B)))", "не(равно(A B))". Требуется доработка.

Число приемов данного типа - 8.

(b) Ввод новых объектов и задание их координатных наборов через вспомогательные параметры.

Пример:

$\forall_{ABCEKabc}$(гиперболоид(E) & осьсимметрии(прямая(AB), E) & прямкоорд(K) $\rightarrow$ $C-$точка & центр(C, E) & $a-$число & $b-$число & $c-$число & коорд(C, K) = (a, b, c))

Прием применяется в посылках задачи на доказательство либо на исследование. Он вводит новые переменные C, a, b, c.

Спецификация приема имеет вид "тип(нормализатор)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))". Создаются также указатели "новыйсимвол(a фикс(0 3))", "новыйсимвол(b фикс(0 4))", "новыйсимвол(c фикс(0 5))", "новыйсимвол(C фикс(0 2))". Требуется доработка.

Число приемов данного типа - 10.

11. Ввод в рассмотрение координатного набора, выраженного через новые параметры.

(a) Ввод в рассмотрение координатного набора, выраженного через новые параметры.

Пример:

$\forall_{ABCKPQabx}$($K = (A, B, C)$ & прямкоорд(K) & актив($l(PQ)$) $\rightarrow$ $a-$число & $b-$число & коорд(P, K) = (a, b))

Прием применяется в посылках задачи на доказательство либо на исследование. Расстояние PQ имеет тип "внешнеизв". Для точки P пока не введен координатный набор. Прием вводит новые переменные a, b.

Спецификация приема имеет вид "тип(меткаперехода)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(контекст(посылка(x3) вид(x3 равно(коорд(P x4)x5)) заголовок(x5 набор)))", "не(заголовок(терм(коорд(P, K)) набор))". Создаются также указатели "новыйсимвол(a фикс(0 1))", "новыйсимвол(b фикс(0 2))". Требуется доработка.

Число приемов данного типа - 18.

(b) Ввод в рассмотрение координатного набора для явно упоминаемых в задаче координат.

Пример:

$\forall_{DKabc}$(коорд$(D,K) = a \to b -$ число & $c -$ число & $a = (b,c)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Выражение a не имеет заголовка "набор" и не равно выражению с таким заголовком. Отсутствует координатный набор для D в какой-либо системе координат. В задаче рассматривается расстояние от точки D до некоторой точки, для которой уже введен координатный набор. Прием вводит новые переменные b, c.

Спецификация приема имеет вид "тип(нормпрямая)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(a набор))", "не(контекст(посылка(x4) вид(x4 равно(a x5)) заголовок(x5 набор)))", "не(контекст(посылка(x4) вид(x4 равно(коорд(D x5)x6)) заголовок(x6 набор)))". Создаются также указатели "новаяпеременная(b)", "новаяпеременная(c)". Требуется доработка.

Число приемов данного типа - 8.

12. Ввод в рассмотрение связанного с текущим термом координатного набора, выраженного через новые параметры.

    (a) Ввод в рассмотрение связанного с текущим термом координатного набора, выраженного через новые параметры.

    Пример:

    $\forall_{Kaxyz}$(прямкоорд(K) & Вектор$(a) \to x -$ число & $y -$ число & $z -$ число & коорд$(a,K) = (x,y,z)$)

    Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "ориентация$(K,(a,b,c))$". Элементы набора идентифицируются без учета порядка. Прием вводит новые переменные x, y, z.

    Спецификация приема имеет вид "тип(норммаксимум)", "терм(t)", где t - текущий терм. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(коорд(a,K)) набор))". Создаются также указатели "контрольвывода(ориентация$(K,(a,b,c))$)", "новыйсимвол(x фикс(0 1))", "новыйсимвол(y фикс(0 2))", "новыйсимвол(z фикс(0 3))". Требуется доработка.

    Число приемов данного типа - 15.

    (b) Ввод в рассмотрение связанного с текущим числовым атомом "неизв" координатного набора, выраженного через новые параметры.

    Пример:

    $\forall_{ABKPQRab}$(прямкоорд(K) & $K = (P,Q,R)$ & актив(вектор$(AB)) \to a -$ число & $b -$ число & коорд$(B,K) = (a,b)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении имеющего тип "неизв" числового атома "скалумнож(вектор(AB), f)". Прием вводит новые переменные a, b.

Спецификация приема имеет вид "тип(тождество)", "терм(t)", где t - текущий числовой атом. Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(коорд(B, K)) набор))", "конец(неизв(теквхожд))". Создаются также указатели "контрольвывода(скалумнож(вектор(AB),f))", "новыйсимвол(a фикс(0 1))", "новыйсимвол(b фикс(0 2))". Требуется доработка.

Число приемов данного типа - 3.

13. Координаты множества объектов.

(a) Вывод уравнений для координат множества объектов.

i. Вывод уравнения для координат множества объектов.

A. Вывод уравнения для координат множества объектов.

Пример:

$\forall_{ABEKabcp}$(прямкоорд(K) & парабола(E) & вершина(A, E) & коорд(A, K) = (a, b) & фокпараметр(E) = p & направлпараболы(E, B) & коорд(B, K) = $(c, 0)$ & $\neg(c = 0)$ $\rightarrow$ коорд(E, K) = $\mathrm{set}_{xy}((y-b)^2 - 2p\mathrm{sg}(c)(x-a) = 0$ & $x -$ число & $y -$ число))

Прием применяется в посылках задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(значениепеременной)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(коорд(E, K)) класс))". Требуется доработка.

Число приемов данного типа - 37.

B. Вывод уравнения для координат множества объектов, не содержащего старых неизвестных параметров.

Пример:

$\forall_{ABEKabcdp}$(эллипс(E) & прямкоорд(K) & вершина(A, E) & вершина(B, E) & осьсимметрии(прямая(AB), E) & коорд(A, K) = (a, b) & коорд(B, K) = (c, d) & разныеточки(A, B) $\rightarrow$ $p -$ число & $0 < p$ & коорд(E, K) = $\mathrm{set}_{xy}(((c-a)(2x-a-c)+(d-b)(2y-b-d))^2 + p((c-a)(2y-b-d)-(d-b)(2x-a-c))^2 - ((c-a)^2+(d-b)^2)^2 = 0$ & $x -$ число & $y -$ число))

Прием применяется в посылках задачи на доказательство либо на исследование. Первые пять антецедентов идентифицируются с посылками. Выражения a, b, c, d не содержат неизвестных. Прием вводит новую переменную p.

Спецификация приема имеет вид "тип(верхнийкрай)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(коорд(E,K)) класс))", "известно(a)", "известно(b)", "известно(c)", "известно(d)", "не(равно(A B))". Создается также указатель "новаяпеременная(p)". Этого почти достаточно.

Число приемов данного типа - 13.

C. Вывод общего вида уравнения для координат множества объектов.

Пример:

$\forall_{ABEKabcd}$(прямкоорд(K) & линвторпорядка(E) & осьсимметрии(прямая(AB), E) & коорд(прямая(AB), K) = $\mathrm{set}_{xy}(x = 0$ & $x -$ число & $y -$ число) $\rightarrow$ $a -$ число & $b -$ число & $c -$ число & $d -$ число & коорд(E, K) = $\mathrm{set}_{uv}(au^2+bv^2+cv+d = 0$ & $u-$число & $v-$ число))

Прием применяется в посылках задачи на доказательство либо на исследование. Антецеденты идентифицируются с посылками. Уравнение для кривой E пока отсутствует. Прием вводит новые переменные a, b, c, d.

Спецификация приема имеет вид "тип(Подчинено)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(коорд(E, K)) класс))", "не(контекст(равно(x5 терм(фикс(E))) посылка(x6) вид(x6 равно(коорд(x5 x7)x8)) символ(x8 класс)))". Создаются также указатели "новаяпеременная(a)", "новаяпеременная(b)", "новаяпеременная(c)", "новаяпеременная(d)". Этого почти достаточно.

Число приемов данного типа - 25.

ii. Вывод уравнения для текущих координат множества объектов.

A. Вывод уравнения для текущих координат множества объектов.

Пример:

$\forall_{AKf}$(прямкоорд(K) & коорд(A, K) = $\mathrm{set}_{xy}(f(x, y) = 0$ & $x-$число & $y -$ число) $\rightarrow$ полкоорд(A, K) = $\mathrm{set}_{pq}(f(p\cos q, p\sin q) = 0$ & $0 \le p$ & $-\pi < q$ & $q \le \pi$ & $p -$ число & $q -$ число))

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "полкоорд(A, K)". Уравнение для полярных координат множества A пока отсутствует.

Спецификация приема имеет вид "тип(семействомножеств)", "терм(t)", где t - текущие координаты множества объектов. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))",

"не(заголовок(терм(полкоорд(A, K)) класс))". Создается также указатель "контрольвывода(полкоорд(A, K))". Этого почти достаточно.

Число приемов данного типа - 15.

B. Вывод уравнения для текущих координат множества объектов, не содержащего старых неизвестных параметров.

Пример:

$\forall_{ABCDKabcdefmnpqr}$(прямкоорд($K$) & прямая($AB$) $\perp$ прямая(CD) & коорд(C, K) = (d, e, f) & коорд(прямая(AB), K) = set_{xyz}(пропорцнаборы($(x+a, y+b, z+c), (p, q, r)$) & $x-$число & $y-$число & $z-$число) & $m = p(a+d)+q(b+e)+r(c+f)$ & $n = p^2+q^2+r^2$ & $D \in$ прямая(AB) $\rightarrow$ коорд(прямая(CD), K) = set_{xyz}(пропорцнаборы($(x-d, y-e, z-f), (mp-(a+d)n, mq-(b+e)n, mr-(c+f)n)$) & $x-$число & $y-$число & $z-$число))

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "коорд(прямая(CD), K)". Выражения a, b, c, d, p, q, r не содержат неизвестных.

Спецификация приема имеет вид "тип(текцвет)", "терм(t)", где t - текущие координаты множества объектов. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(коорд(прямая(CD,K))) класс))", "известно(a)", "известно(b)", "известно(c)", "известно(d)", "известно(e)", "известно(f)", "известно(p)", "известно(q)", "известно(r)", "усм(актив(прямая(CD)))", "не(равно(C D))". Создается также указатель "контрольвывода(коорд(прямая(CD),K))". Этого почти достаточно.

Число приемов данного типа - 5.

C. Вывод общего вида уравнения для текущих координат множества объектов.

Пример:

$\forall_{EKabcdef}$(эллипс(E) & прямкоорд(K) $\rightarrow$ $a-$число & $b-$число & $c-$число & $d-$число & $e-$число & $f-$число & коорд(E, K) = $\text{set}_{xy}(ax^2+bxy+cy^2+dx+ey+f=0$ & $x-$число & $y-$число) & $b^2-4ac<0$)

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "коорд(E, K)". Прием вводит новые переменные a, b, c, d, e, f.

Спецификация приема имеет вид "тип(фильтротрезков)", "терм(t)", где t - текущие координаты множества объектов. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(коорд(E, K)) класс))", "конец(не(легковидеть(опредкоорд(коорд(E, K)))))". Создаются также указатели "контрольвывода(коорд(E, K))", "новыйсимвол(a фикс(0 1))", "новыйсимвол(b

фикс(0 2))", "новыйсимвол(c фикс(0 3))", "новыйсимвол(d фикс(0 4))", "новыйсимвол(e фикс(0 5))", "новыйсимвол(f фикс(0 6))". Этого почти достаточно.

Число приемов данного типа - 23.

D. Вывод общего вида уравнения для текущих координат множества объектов, не содержащего старых неизвестных параметров.

Пример:

$\forall_{ABKabc}$(прямкоорд(K) & $l(AB) = c \to a -$ число & $b -$ число & коорд(окружность(AB), K) = $\text{set}_{xy}(x^2+y^2-2ax-2by+a^2+b^2-c^2 = 0$ & $x -$ число & $y -$ число))

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "коорд(окружность(AB),K)". Второй антецедент выделен указателем "идентификатор". Выражение c не содержит неизвестных. Прием вводит новые переменные a, b.

Спецификация приема имеет вид "тип(внеотрезка)", "терм(t)", где t - текущие координаты множества объектов. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(коорд(окружность(AB),K)) класс))", "известно(c)". Создаются также указатели "контрольвывода(коорд(окружность(AB),K))", "новыйсимвол(a фикс(0 1))", "новыйсимвол(b фикс(0 2))". Требуется доработка.

Число приемов данного типа - 6.

iii. Вывод параметрического описания для текущих координат множества объектов.

Пример:

$\forall_{ABKMabcd}$(Прямая(A) & $A \parallel B$ & Вектор(B) & коорд(B, K) = (c, d) & $M \in A$ & коорд(M, K) = $(a, b) \to$ коорд(A, K) = $\text{set}_{xy}(\exists_t(t-$число & $x = ct + a$ & $y = dt + b)))$

Прием применяется в посылках задачи на исследование, имеющей цель "вспомпараметр". Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "коорд(A, K)". Все антецеденты, кроме четвертого и шестого, выделенных указателем "идентификатор", идентифицируются с посылками.

Спецификация приема имеет вид "тип(Преобр)", "терм(t)", где t - текущие координаты множества объектов. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "тип(исследовать)", "цель(вспомпараметр)", "не(заголовок(терм(коорд(A, K)) класс))". Создается также указатель "контрольвывода(коорд(A, K))". Требуется доработка.

Число приемов данного типа - 3.

iv. Вывод уравнения для координат множества объектов, связанного с текущим термом.
Пример:
$\forall_{ABCDEKabcd}(E \in$ прямая(AB) & $C \in$ прямая(AB) & $D \in$ прямая(AB) & коорд$(C,K) = (a,b)$ & коорд$(D,K) = (c,d)$ & разныеточки$(C,D) \rightarrow$ коорд(прямая$(AB),K) = \text{set}_{xy}((d-b)x + (a-c)y + bc - ad = 0$ & $x -$ число & $y -$ число))
Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "классточки(E,p)".
Спецификация приема имеет вид "тип(перемещение)", "терм(t)", где t - текущий терм. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(коорд(прямая(AB),K)) класс))", "усм(актив(прямая(AB)))", "не(равно(C E))", "не(равно(D E))не(равно(C D))". Создается также указатель "контрольвывода(классточки(E x5))". Этого почти достаточно.

Число приемов данного типа - 3.

v. Вывод уравнений для координат множеств объектов.

A. Вывод уравнений для координат множеств объектов.

Пример:

$\forall_{ABCEKabcdem}$(прямкоорд(K) & асимптота(прямая$(AB),E)$ & асимптота(прямая$(AC),E)$ & коорд$(E,K) = \text{set}_{xy}(ax^2 + bx + cy^2 + dy + e = 0$ & $x -$ число & $y -$ число) & гипербола(E) & $m = \sqrt{-c/a}$ & разныепрямые(прямая(AB), прямая$(AC)) \rightarrow$ коорд(прямая$(AB),K) = \text{set}_{uv}(2acu + 2acmv + ad + mbc = 0$ & $u -$ число & $v -$ число) & коорд(прямая$(AC),K) = \text{set}_{uv}(2acu - 2acmv + ad - mbc = 0$ & $u -$ число & $v -$ число))

Прием применяется в посылках задачи на доказательство либо на исследование. Первые пять антецедентов идентифицируются с посылками.

Спецификация приема имеет вид "тип(арифмпрогрессия)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(коорд(прямая(AB),K)) класс))", "не(заголовок(терм(коорд(прямая(AC),K)) класс))", "не(контекст(равно(x6 терм(фикс(прямая(AB))))) посылка(x7) вид(x7 равно(коорд(x6 x8) x9)) символ(x9 класс)))", "не(контекст(равно(x6 терм(фикс(прямая(AC))))) посылка(x7) вид(x7 равно(коорд(x6 x8)x9)) символ(x9 класс)))", "не(равно(терм(прямая(AB)) терм(прямая(AC))))". Этого почти достаточно.

Число приемов данного типа - 5.

vi. Дополнительное построение и вывод уравнений для координат множеств объектов.

Пример:

$\forall_{ABCDEKabcd}$(прямкоорд(K) & равнгипербола(E) & асимптота(прямая(AB), E) & коорд(прямая(AB), K) = $\text{set}_{xy}(ax + by + c = 0$ & $x -$ число & $y -$ число) $\rightarrow C -$ точка & $D -$ точка & асимптота(прямая(CD), E) & $d -$ число & коорд(прямая(CD), K) = $\text{set}_{uv}(-bu + av + d = 0$ & $u -$ число & $v -$ число))

Прием применяется в посылках задачи на доказательство либо на исследование. Первые три антецедента идентифицируются с посылками, четвертый - выделен указателем "идентификатор". Прием вводит новые переменные C, D, d.

Спецификация приема имеет вид "тип(комментарий)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))". Создаются также указатели "новыйсимвол(d фикс(0 4))", "новыйсимвол(C фикс(0 4))", "новыйсимвол(D фикс(0 4))". Требуется доработка.

Число приемов данного типа - 11.

vii. Разбиение исследуемого множества объектов, заданного уравнением, на несколько подмножеств, тоже заданных уравнениями.

Пример:

$\forall_{ABEKfgh}(f(x, y, z) = gh$ & коорд(E, K) = $\text{set}_{xyz}(f(x, y, z) = 0$ & $x -$ число & $y -$ число & $z -$ число) $\rightarrow A -$ set & $B -$ set & $A \cup B = E$ & коорд(A, K) = $\text{set}_{xyz}(g = 0$ & $x -$ число & $y -$ число & $z -$ число) & коорд(B, K) = $\text{set}_{xyz}(h = 0$ & $x -$ число & $y -$ число & $z -$ число))

Прием применяется в задачах на исследование, имеющих цель "эллипсоид" (т.е. решаемых для исследования свойств поверхности, заданной своим уравнением). Первый антецедент выделен указателем "идентификатор", второй - идентифицируется с посылкой. Прием вводит новые переменные A, B.

Спецификация приема имеет вид "тип(нормусм)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "тип(исследовать)", "цель(эллипсоид)". Он не прорабатывался.

Число приемов данного типа - 3.

(b) Вывод соотношений для параметров уравнений координат множества объектов.

i. Вывод соотношений для параметров уравнений координат множества объектов.

A. Вывод соотношений для параметров уравнений координат множества объектов.

Пример:

$\forall_{ABCDEKMNabcdpq}(M \in$ прямая(AB) & $N \in$ прямая(CD) & $E \in$ прямая(MN) & $l(ME) = l(NE)$ & коорд(прямая(AB), K) = $\text{set}_{xy}(ax + by + c = 0$ & $x -$ число & $y -$ число) & коорд(прямая(CD), K) = $\text{set}_{uv}(au + bv + d = 0$ & $u -$ число &

v – число) & коорд$(E, K) = (p, q)$ & разныепрямые(прямая(AB), прямая$(MN)) \rightarrow 2ap + 2bq + c + d = 0)$

Прием применяется в посылках задачи на доказательство либо на исследование. Пятый, шестой и седьмой антецеденты идентифицируются с посылками.

Спецификация приема имеет вид "тип(терминал)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "усм(актив(прямая(AB)))", "усм(актив(прямая(CD)))", "усм(актив(прямая(MN)))", "не(равно$(M\ N)$)", "не(равно(терм(прямая(AB))) терм(прямая(MN))))". Требуется доработка.

Число приемов данного типа - 44.

B. Вывод соотношений для параметров содержащих неизвестные уравнений координат множеств объектов.

Пример:

$\forall_{EKabcdef}$(прямкоорд(K) & коорд$(E, K) = \text{set}_{xy}(ax^2+bxy+cy^2+dx+ey+f = 0$ & x – число & y – число) & парабола$(E) \rightarrow b^2 - 4ac = 0)$

Прием применяется в посылках задачи на доказательство либо на исследование. Уравнение параболы содержит неизвестные.

Спецификация приема имеет вид "тип(повторчисло)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(известно(фикс(2 2)))". Этого достаточно.

Число приемов данного типа - 17.

ii. Вывод соотношений для параметров текущего уравнения координат множеств объектов.

Пример:

$\forall_{ABCDKabcdef}$(коорд(прямая$(AD), K) = \text{set}_{xy}(ax + by + c = 0$ & x – число & y – число) & прямая$(AD) \perp$ прямая(BC) & коорд(прямая$(BC), K) = \text{set}_{uv}(du + ev + f = 0$ & u – число & v – число) & прямкоорд$(K) \rightarrow ad + be = 0)$

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "коорд(прямая(BC),K)". Первые два антецедента идентифицируются с посылками, третий - выделен указателем "идентификатор".

Спецификация приема имеет вид "тип(фокпараметр)", "терм(t)", где t - текущий терм для координат множества объектов. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "усм(актив(прямая(AD)))", "усм(актив(прямая(BC)))". Создается также указатель "контрольвывода(коорд(прямая$(BC)K$))". Этого почти достаточно.

Число приемов данного типа - 7.

iii. Связь числовых атомов с параметрами уравнений для координат множеств объектов.

A. Связь числовых атомов с параметрами уравнений для координат множеств объектов.

Пример:

$\forall_{ABCDEKabcdenpq}$(прямкоорд(K) & гипербола(E) & асимптота(прямая(AB), E) & асимптота(прямая(AC), E) & коорд(E, K) = $\text{set}_{xy}(ax^2 + bx + cy^2 + dy + e = 0$ & x − число & y − число) & $D \in$ Угол(BAC) & коорд(D, K) = (p, q) & $2cq + d = 0$ & актив($\angle(BAC)$) → $(a + c)\,\text{tg}(\angle(BAC)) - 2\text{sg}(c)\sqrt{-ac} = 0$)

Прием применяется в посылках задачи на доказательство либо на исследование. Первые шесть антецедентов идентифицируются с посылками.

Спецификация приема имеет вид "тип(функциональныйсимвол)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "усм(актив(прямая(AB)))", "усм(актив(прямая(AC)))", "не(усм(принадлежит(C прямая(AB))))". Этого почти достаточно.

Число приемов данного типа - 7.

iv. Связь текущего числового атома с параметрами уравнения для координат множества объектов.

A. Связь текущего числового атома с параметрами уравнения для координат множества объектов.

Пример:

$\forall_{EKabcdem}$(прямкоорд(K) & коорд(E, K) = $\text{set}_{xy}(ax^2+bx+cy^2+dy+e = 0$ & $x-$число & $y-$число) & эллипс(E) & $m = b^2c+d^2a-4ace \rightarrow$ малаяось(E) = $\min(\sqrt{m/(a^2c)}, \sqrt{m/(ac^2)})$ & $0 < ma$ & $0 < ac$)

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении выражения "малаяось(E)". Первые три антецедента идентифицируются с посылками.

Спецификация приема имеет вид "тип(монотоннозависит)", "терм(t)", где t - текущий числовой атом. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(известно(результат))". Создается также указатель "контрольвывода(малаяось(E))". Этого почти достаточно.

Число приемов данного типа - 29.

v. Вывод ограничения на параметры уравнения для координат множества объектов.

Пример:

$\forall_{ABCDEFGKabcdepq}(E \in$ прямая(AB), $E \in$ прямая(CD) & $F \in$ Угол(BED) & $G \in$ Угол(BED) & коорд(прямая(AB), K) =

$\text{set}_{xy}(ax + by + c = 0$ & $x -$ число & $y -$ число) & координ$(F, K) = (d, e)$ & координ$(G, K) = (p, q) \to 0 \leq (ad + be + c)(ap + bq + c))$

Прием применяется в посылках задачи на доказательство либо на исследование. Третий, четвертый и пятый антецеденты идентифицируются с посылками.

Спецификация приема имеет вид "тип(нормнабор)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "конец(не(легковидеть$(0 \leq (ad + be + c)(ap + bq + c))))$", "усм(актив(прямая$(AB)))$", "не(равно$(A\ B)$)", "усм(актив(прямая$(CD)))$", "не(равно$(C\ D)$)". Этого достаточно.

Число приемов данного типа - 22.

(c) Вывод соотношений для координат объектов с помощью уравнения координат множеств объектов.

i. Вывод соотношений для координат объектов с помощью уравнения координат множеств объектов.

Пример:

$\forall_{ABKPabc}$(координ$(A, K) = (a, b, c)$ & координ(прямая$(AB), K) = \text{set}_{xyz}(P(x, y, z)) \to P(a, b, c))$

Прием применяется в посылках задачи на доказательство либо на исследование. Второй антецедент идентифицируется с посылкой, первый - выделен указателем "идентификатор".

Спецификация приема имеет вид "тип(числоценка)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))". Требуется доработка.

Число приемов данного типа - 8.

(d) Выражение координат объекта через параметры уравнений для координат множеств объектов.

i. Выражение координат объекта через параметры уравнений для координат множеств объектов.

Пример:

$\forall_{ABKabcd}$(прямкоорд(K) & координ(окружность$(AB), K) = \text{set}_{xy}(ax^2 + ay^2 + bx + cy + d = 0$ & $x -$ число & $y -$ число$) \to$ координ$(A, K) = (-b/(2a), -c/(2a))$ & $\neg(a = 0))$

Прием применяется в посылках задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(равныедлины)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(заголовок(терм(координ(A, K)) набор))", "не(контекст(посылка(x5) вид(x5 равно(координ$(A$ x6)x7)) заголовок(x7 набор)))". Этого почти достаточно.

Число приемов данного типа - 17.

(e) Ввод в рассмотрение координат множества объектов, связанного с текущим термом.

Пример:

$\forall_{ABCDK}$(прямкоорд(K) $\rightarrow$ актив(коорд(прямая(AB), K)))

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "уголмежду(прямая(AB), прямая(CD))".

Спецификация приема имеет вид "тип(значениемн)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(контекст(посыка(x1) позиция(x2 x1)вид(x2 коорд(прямая(AB)x3))))". Создается также указатель "контрольвывода(уголмежду(прямая(AB), прямая(CD)))". Этого достаточно.

Число приемов данного типа - 5.

(f) Характеризация множества объектов, использующая уравнения для их координат.

i. Усмотрение вида множества объектов по уравнению для его координат.

Пример:

$\forall_{EKabcdef}$(прямкоорд(K) & коорд(E, K) = $\mathrm{set}_{xy}(ax^2 + bxy + cy^2 + dx + ey + f = 0$ & $x -$ число & $y -$ число) & $0 < 4ac - b^2$ & $(a + c)(4acf + bde - ae^2 - cd^2 - fb^2) < 0 \rightarrow$ эллипс(E))

Прием применяется в посылках задачи на доказательство либо на исследование.

Спецификация приема имеет вид "тип(подобны)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(контекст(посылка(x7) заголовок(x7 гипербола парабола эллипс) равно(E первыйтерм(x7))))". Требуется доработка.

Число приемов данного типа - 12.

ii. Усмотрение вида исследуемого множества объектов по параметрам уравнения для его координат.

Пример:

$\forall_{EKacdefg}$(прямкоорд(K) & коорд(E, K) = $\mathrm{set}_{xyz}(ax^2 + ay^2 + cz^2 + dx + ey + fz + g = 0$ & $x -$ число & $y -$ число & $z -$ число) & $ac < 0$ & $cd^2 + ce^2 + af^2 - 4acg = 0 \rightarrow$ Конус(E) & круглый(E))

Прием применяется в задачах на исследование, имеющих цель "эллипсоид". Такая цель - дополнение к цели "исследовать", указывающая, что требуется исследовать свойства поверхности, заданной своим уравнением.

Спецификация приема имеет вид "тип(Отрицание)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры

"тип(исследовать)", "цель(эллипсоид)", "не(контекст(посылка(x2) вид(x2 Конус(E))))", "или(не(заголовок(a 1)) не(заголовок(c 1)))". Требуется доработка.

Число приемов данного типа - 8.

iii. Текущий терм инициирует характеризацию множеств объектов с помощью уравнений для координат.

Пример:

$\forall_{ABCDEKabcdefpqrs}$(коорд(плоскость($CDE$), K) = set$_{xyz}$($px+qy+rz+s = 0$ & $x-$число & $y-$число & $z-$число) & коорд(A, K) = (a, b, c) & коорд(B, K) = (d, e, f) & $A-$точка & $B-$точка & $0 \leq (ap+bq+cr+s)(dp+eq+fr+s) \rightarrow$ однасторона(A, B, плоскость(CDE)))

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "классточки(A x7)". Указатель "контекст" определяет дополнительную идентификацию посылки, содержащей подтерм "полупространство(плоскость(CDE)B)" либо "обрполупространство(плоскость(CDE)B)".
Первый антецедент идентифицируется с посылкой.

Спецификация приема имеет вид "тип(модификатор)", "терм(t)", "заголовок($s_1 \ldots s_m$)", где t - текущий терм; $s_1, \ldots, s_n$ - такие заголовки, что ни одно из утверждений , получаемых из консеквента теоремы заменой его заголовка на один из этих заголовков, не усматривается из контекста. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(легковидеть(однасторона(A, B, плоскость(CDE))))", "не(легковидеть(разныестороны(A, B, плоскость(CDE))))". Создается также указатель "контрольвывода(классточки(A,x7))". Требуется доработка.

Число приемов данного типа - 7.

iv. Усмотрение стандартного представления исследуемого множества объектов по параметрам уравнения для его координат.

Пример:

$\forall_{ABEKabcdefp}$(прямкоорд(K) & коорд(E, K) = set$_{xy}$($ax^2+bxy+cy^2+dx+ey+f = 0$ & $x-$число & $y-$число) & $b^2-4ac = 0$ & $2cd-be = 0$ & $e^2-4cf = 0$ & $\neg(c = 0) \rightarrow A-$точка & $B-$точка & прямая(AB) = E & коорд(прямая(AB), K) = set$_{uv}$($bu+2cv+e = 0$ & $u-$число & $v-$число))

Прием применяется в задачах на исследование, имеющих цель "линия". Эта цель дополняет цель "исследовать" и указывает, что требуется исследовать свойства линии, заданной своим уравнением. Вводятся новые переменные A, B.

Спецификация приема имеет вид "тип(частное)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "тип(исследовать)", "цель(линия)", "не(контекст(посылка(x7)

вид(x7 равно(E прямая(x8 x9)))))", "или(не(заголовок(b 1))не(заголовок(c 1))и(или(не(заголовок(d 1))не(заголовок(e 1)))не(заголовок(a 1))))", "или(не(заголовок(f 0)) не(заголовок(e 1)))". Создаются также указатели "новыйсимвол(A фикс(0 1))", "новыйсимвол(B фикс(0 2))". Требуется доработка.

Число приемов данного типа - 11.

v. Усмотрение специальной системы координат, связанной с исследуемым множеством объектов, заданным уравнением.

Пример:

$\forall_{ABCEKQ}$(прямкоорд(K) & коорд(E, K) = $\mathrm{set}_{xy}(ax^2 + cy^2 + e = 0$ & $x-$число & $y-$число) & $0 < ac$ & $0 \le e(a-c)$ & $ce < 0 \to$ каноничкоорд(K, E))

Прием применяется в задачах на исследование, имеющих цель "точки" либо цель "линия". Первая из этих целей указывает на необходимость дать бескоординатное описание множества точек, вторая - дополняет цель "исследовать" и указывает, что требуется исследовать свойства линии, заданной своим уравнением.

Спецификация приема имеет вид "тип(прямыеуглы)", "цель($s_1 \dots s_n$)", где s_i - требуемые цели задачи на исследование. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "тип(исследовать)", "или(цель(точки)цель(линия))". Этого почти достаточно.

Число приемов данного типа - 3.

14. Две системы координат.

(a) Выражение координат объекта в одной системе координат через его координаты в другой системе.

i. Выражение координат объекта в одной системе через его координаты в другой системе.

Пример:

$\forall_{ABCDEFGKQabcd}(K = (B, C, D)$ & $Q = (E, F, G)$ & коорд(B, Q) = (a, b) & вектор(BC) = вектор(EG) & вектор(BD) = $-$вектор(EF) & коорд(A, Q) = (c, d) & сисккоорд(K) $\to$ коорд(A, K) = $(d - b, a - c)$)

Прием применяется в посылках задачи на доказательство либо на исследование. Первые два и последние два антецедента идентифицируются с посылками.

Спецификация приема имеет вид "тип(актив)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "посылка", "тип(исследовать)", "цель(линия)", "не(равно(K Q))", "коммент(коорд коорд A умножение(K, Q))". Этого почти достаточно. Символ "умножение" в комментарии используется как некий абстрактный коммутативный символ - для лексикографического упорядочения пары K, Q.

Число приемов данного типа - 4.

ii. Выражение координат объекта в одной системе через его координаты в другой системе при исследовании свойств множества объектов.

В качестве примера рассмотрим прием, теорема которого - та же, что в предыдущем пункте. Он применяется в задачах на исследование, имеющих цель "линия", т.е. при исследовании свойств линии, заданной своим уравнением. Антецеденты обрабатываются так же, как и выше.

Спецификация приема имеет вид "тип(контрольпрограммы)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "тип(исследовать)", "цель(линия)", "не(равно(Q K))", "коммент(коорд коорд D умножение(Q, K))". Создается также указатель "замечание(коорд коорд D умножение(Q, K))".

Число приемов данного типа - 3.

(b) Выражение координат текущего объекта в одной системе через его координаты в другой системе.

Снова повторим теорему из предыдущих двух пунктов. Теперь прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении в задаче выражения "коорд(D, Q)". Обработка антецедентов прежняя.

Спецификация приема имеет вид "тип(автоменю)", "терм(t)", где t - координаты текущего объекта. Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(равно(Q K))", "коммент(коорд коорд D умножение(Q, K))", "переменная(K)". Создается также указатель "замечание(коорд коорд D умножение(Q, K))".

Число приемов данного типа - 6.

(c) Выражение текущих координат множества объектов в одной системе через его координаты в другой системе.

Пример:

$\forall_{ABCKMTabcdefp}(T = (A, B, C)$ & коорд$(A, K) = (a, b)$ & коорд(вектор(AB), $K) = (c, d)$ & коорд(вектор$(AC), K) = (e, f)$ & коорд$(M, K) = \text{set}_{xy}(x -$ число & $y -$ число & $p(x, y)) \rightarrow$ коорд$(M, T) = \text{set}_{xy}(x -$ число & $y -$ число & $p(a + cx + ey, b + dx + fy)))$

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "коорд(M, T)". Первый и пятый антецеденты идентифицируются с посылками.

Спецификация приема имеет вид "тип(учетзнаков)", "терм(t)", где t - текущие координаты множества объектов. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(равно(T K))", "коммент(коорд перестановка M умножение(T K))", "переменная(T)", "не(контекст(равно(x7 терм(

коорд(M, T))) известно(x7) заголовок(x7 класс)))". Создается также указатель "контрольвывода(коорд(M, T))". Требуется доработка.

Число приемов данного типа - 6.

(d) Выражение координат исследуемого множества объектов в одной системе через его координаты в другой.

Пример:

$\forall_{ABCKMQabcdefpq}(K = (A, B, C)$ & коорд$(A, Q) = (a, b)$ & коорд(вектор(AB), $Q) = (c, d)$ & коорд(вектор$(AC), Q) = (e, f)$ & коорд$(M, K) = \text{set}_{xy}(x -$ число & $y -$ число & $q(x, y))$ & $p = cf - de$ & $\neg(p = 0)$ & прямкоорд$(Q) \rightarrow$ коорд$(M, Q) = \text{set}_{xy}(x -$ число & $y -$ число & $q((fx - ey + be - af)/p, (cy - dx + ad - bc)/p)))$

Прием применяется в задачах на исследование, имеющих цель "линия". Эта цель дополняет цель "исследовать" и указывает, что требуется исследовать свойства линии, заданной своим уравнением. Первый, пятый и восьмой антецеденты идентифицируются с посылками.

Спецификация приема имеет вид "тип(коэффиентмн)", "цель($s_1 \ldots s_n$)", где s_i - требуемые цели задачи на исследование. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "тип(исследовать)", "цель(линия)", "не(равно(Q K))", "коммент(коорд перестановка M умножение(Q K))", "переменная(K)". Этого почти достаточно.

Число приемов данного типа - 3.

(e) Ввод в рассмотрение координат объекта, для которого уже рассматриваются координаты в другой системе координат.

i. Ввод в рассмотрение координат элементов одной системы координат в другой системе, если текущие координаты объекта в одной системе не известны, а в другой - известны.

Пример:

$\forall_{ABCDKMQabcde}(K = (A, B, C, D)$ & коорд$(M, Q) = a \rightarrow c -$ число & $d -$ число & $e -$ число & коорд$(A, Q) = (c, d, e))$

Прием применяется в посылках задачи на доказательство либо на исследование. Указатель "контрольвывода" инициирует попытку его применения при усмотрении подвыражения "коорд(M, K)". Антецеденты идентифицируются с посылками. Выражение a не содержит неизвестных, а выражение "коорд(M, K)" - содержит. Для точки A пока не введен координатный набор в системе координат Q.

Спецификация приема имеет вид "тип(контрольодз)", "терм(t)", где t - текущие координаты. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать) тип(исследовать))", "не(известно(терм(коорд(M, K))))", "известно(a)", "не(заголовок(терм(коорд(A, Q))) набор))". Создается также указатель "контрольвывода(коорд(M, Q))". Этого почти достаточно.

Число приемов данного типа - 5.

(f) Ввод в рассмотрение новой системы координат.

i. Ввод в рассмотрение вспомогательной системы координат и задание в ней общего вида уравнения для множества объектов.

Пример:

$\forall_{ABEFGHKQabcpq}$(гипербола(E) & действось(прямая(AB), E) & прямкоорд(K) & коорд(прямая(AB), K) = $\mathrm{set}_{xy}(ax+by+c=0$ & $x-$число & $y-$число) $\to$ $F-$точка & $G-$точка & $H-$точка & $(F,G,H)=Q$ & $p-$число & $q-$число & $d-$число & $e-$число & $0<p$ & $0<q$ & коорд(F,K) = (d,e) & коорд(G,K) = $(d-b,e+a)$ & коорд(H,K) = $(d+a,e+b)$ & коорд(E,Q) = $\mathrm{set}_{uv}(pu^2-qv^2-pq=0$ & $u-$число & $v-$число) & $ad+be+c=0$)

Прием применяется в посылках задачи на доказательство либо на исследование. Четвертый антецедент выделен указателем "идентификатор", остальные - идентифицируются с посылками. Уравнение гиперболы E в задаче отсутствует. Прием вводит новые переменные F,G,H,Q,d,e,p,q.

Спецификация приема имеет вид "тип(номерперехода)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(контекст(посылка(x6) вид(x6 равно(коорд(E x7)x8)) заголовок(x8 класс)))". Создаются указатели для ввода новых переменных. Этого почти достаточно.

Число приемов данного типа - 25.

ii. Ввод в рассмотрение вспомогательной системы координат, в которой уравнение для исследуемого множества объектов имеет стандартный вид.

Пример:

$\forall_{ABCEKQabcdempq}$(прямкоорд(K) & коорд(E,K) = $\mathrm{set}_{xy}(ax^2+bx+cy^2+dy+e=0$ & $x-$число & $y-$число) & $ac<0$ & $m=b^2c+d^2a-4ace$ & $mc<0$ & $p=-b/(2a)$ & $q=-d/(2c)$ $\to$ $A-$ точка & $B-$ точка & $C-$ точка & $(A,B,C)=Q$ & прямкоорд(Q) & каноничкоорд(Q,E) & коорд(A,K) = (p,q) & коорд(C,K) = $(p+1,q)$ & коорд(B,K) = $(p,q+1)$ & коорд(E,Q) = $\mathrm{set}_{vu}(4a^2cu^2/m+4ac^2v^2/m=1$ & $u-$число & $v-$число) & мнимаяполуось(E) = $\sqrt{-m/(4a^2c)}$ & действполуось(E) = $\sqrt{-m/(4c^2a)}$)

Прием применяется в задачах на исследование, имеющих цель "точки" либо цель "линия". Первые два антецедента идентифицируются с посылками. Для E пока не введена каноническая система координат. Прием вводит новые переменные A,B,C,Q.

Спецификация приема имеет вид "тип(символвхождения)", "цель($s_1\ldots s_n$)", где s_i - требуемые цели задачи на исследование. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "тип(исследовать)", "или(цель(точки)цель(линия))", "или(не(заголовок(a 1)) не(заголовок(c 1)))". Создаются указатели для ввода новых переменных. Требуется доработка.

Число приемов данного типа - 30.

15. Первичный ввод системы координат.

    (a) Ввод системы координат, связанной с объектами задачи.

    Пример:

    $\forall_{ABCDEFGa}$(куб(a) & ребро(отрезок(AB), a) & ребро(отрезок(AC), a) & ребро(отрезок(AD), a) & разныеточки(B, C) & разныеточки(B, D) & разныеточки(C, D) $\rightarrow$ E – точка & F – точка & G – точка & K = (A, E, F, G) & прямкоорд(K) & $E \in$ прямая(AB) & $F \in$ прямая(AC) & $G \in$ прямая(AD) & точкалуча(A, B, E) & точкалуча(A, C, F) & точкалуча(A, D, G))

    Прием применяется в посылках задачи на доказательство либо на исследование. Первые четыре антецедента идентифицируются с посылками, остальные - обрабатываются проверочным оператором. Прямоугольная система координат в задаче не рассматривается, однако рассматривается какой-то вектор с началом A. Прием вводит новые переменные E, F, G, K.

    Спецификация приема имеет вид "тип(единица)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(равно(B C))", "не(равно(B D))", "не(равно(C D))". Создаются указатели для ввода новых переменных E, F, G, K.

    Число приемов данного типа - 3.

    (b) Первичный ввод системы координат, в которой задается общий вид уравнения для координат множества объектов.

    Пример:

    $\forall_{EKab}$(эллипс(E) & большаяось(E) = a & малаяось(E) = $b \rightarrow$ прямкоорд(K) & $0 < a$ & $0 < b$ & $0 \leq a - b$ & коорд(E, K) = $\text{set}_{xy}(4x^2/a^2 + 4y^2/b^2 = 1$ & x – число & y – число))

    Прием применяется в посылках задачи на доказательство либо на исследование. Прямоугольная система координат в задаче не рассматривается. Прием вводит новую переменную K.

    Спецификация приема имеет вид "тип(символы)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "посылка", "или(тип(доказать)тип(исследовать))", "не(контекст(посылка(x3) заголовок(x3 прямкоорд)))". Создается указатель для ввода новой переменной K. Требуется проработка.

    Число приемов данного типа - 9.

16. Задачи на преобразование, имеющие цель "класс".

    Напомним, что цель "класс" задачи на преобразование означает требование отсутствия в ответе описателей.

(a) Выражение через вспомогательные параметры координат объекта в задаче на преобразование, имеющей цель "класс".

Пример:

$\forall_{ABCDK}(K = (A, B, C)$ & $D -$ точка & прямкоорд$(K) \to a -$ число & $b -$ число & коорд$(D, K) = (a, b))$

Прием применяется в посылках задачи на преобразование, имеющей цель "класс". Переменная D идентифицируется с переменной, причем координатный набор для точки D пока не введен. Прием вводит новые переменные a, b и регистрирует их как вспомогательные параметры.

Спецификация приема имеет вид "тип(регистрациязадачи)". Справочник "заголовокприема" указывает уровень срабатывания 6 и вводит фильтры "посылка", "тип(преобразовать)", "цель(класс)". Создаются указатели для ввода новых переменных a, b. Требуется проработка.

Число приемов данного типа - 3.

(b) Ввод в рассмотрение системы координат в задаче на преобразование, имеющей цель "класс".

Пример:

$\forall_{ABCDEK}($прямая$(AB) \perp$ прямая(AC) & разныеточки(A, B) & разныеточки$(A, C) \to D -$ точка & $E -$ точка & $(A, D, E) = K$ & прямкоорд(K) & $E \in$ прямая(AC) & $D \in$ прямая(AB) & $\neg(A \in$ интервал$(CE))$ & $\neg(A \in$ интервал$(BD)))$

Прием применяется в посылках задачи на преобразование, имеющей цель "класс". Имеется указание на планиметрическую ситуацию, но в посылках не встречается символ "коорд". Прием вводит новые переменные D, E, K и регистрирует их как вспомогательные параметры.

Спецификация приема имеет вид "тип(нормкласс)". Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "тип(преобразовать)", "цель(класс)", "коммент(коорд)", "усм(актив(прямая$(AB)))$", "не(равно$(A\ B))$", "усм(актив(прямая$(AC)))$", "не(равно$(A\ C))$". Создаются указатели для ввода новых переменных D, E, K. Требуется проработка.

Число приемов данного типа - 3.

17. Задачи на исследование, имеющие цель "точки".

Цель "точки" указывает на необходимость получения бескоординатного описания множества точек.

(a) Переход от координатного задания множества к бескоординатному в задаче на исследование, имеющей цель "точки".

Пример:

$\forall_{ABCKa}($коорд$(A, K) = \text{set}_{xy}(x = a$ & $y -$ число$) \to B =$ тчкоорд$(K, (-a, 0))$ & $C =$ тчкоорд$(K, (-a, 1))$ & $A =$ прямая$(BC))$

Прием применяется в задачах на исследование, имеющих цель "точки". Отсутствует посылка вида $A = t$, где выражение t не содержит символа "точки". Прием вводит новые переменные B, C.

Спецификация приема имеет вид "тип(узелвывода)". Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит фильтры "тип(исследовать)", "цель(точки)", "не(контекст(посылка(х2) вид(х2 равно(A х3)) не(входит(точки х3))))". Создаются также указатели для ввода переменных B, C. Этого почти достаточно.

Число приемов данного типа - 6.

14.2.6 Вывод в условиях задачи на описание

Разрешение относительно неизвестных

1. Вывод условия, разрешенного относительно неизвестного подвыражения.

   Пример:

   $\forall_{abpqn}(a - \text{целое } \& \neg(p|b) \ \& \ b - \text{целое } \& \ n - \text{целое } \& \ 0 \leq n \ \& \ ap^n = b \rightarrow n = 0)$

   Последний антецедент идентифицируется с условием задачи на описание. Выражение n содержит неизвестные. Переменная p идентифицируется с натуральной константой.

   Спецификация приема имеет вид "тип(возрастает)", "неизвестная(t)", где t - переменная, идентифицируемая с неизвестным подвыражением. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "не(известно(n))".

   Число приемов данного типа - 3.

2. Вывод следствия условия, в котором исключено сложное понятие.

   Пример:

   $\forall_{abc}(\arcsin(a) + b = c \rightarrow a - \sin(c - b) = 0)$

   Прием применяется к условию задачи на описание. Выражения a, b содержат неизвестные, причем выражение a содержит символ "синус" либо "косинус". Выражение c неизвестных не содержит.

   Спецификация приема имеет вид "тип(нормМаксимум)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)". Требуется проработка.

   Число приемов данного типа - 19.

3. Вывод условия, ограничивающего значения неизвестного подвыражения конечным множеством.

   Пример:

$\forall_{abcdmn}(n-$натуральное & $0 < a$ & $d = [(c-m)/a]$ & $m \leq b$ & $an+b = c \rightarrow n \leq d)$

Первый и последний антецеденты идентифицируются с условиями задачи на описание. Выражение n содержит неизвестные. Выражения a, c, d - целочисленные константы. Четвертый антецедент определяет численную нижнюю оценку выражения b. Затем третий вычисляет d.

Спецификация приема имеет вид "тип(констцелое)", "неизвестная(n)", где n - переменная, идентифицируемая с содержащим неизвестные выражением. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "не(известно(n))". Требуется проработка.

Число приемов данного типа - 6.

4. Вывод условия, подготавливающего ограничение значений неизвестного подвыражения конечным множеством.

   Пример:

   $\forall_{abcdp}((a+b)/c -$ целое & $(d-b)/c -$ целое & $p = a+d \rightarrow p/c -$ целое$)$

   Первые два антецедента идентифицируются с условиями задачи на описание, третий - выделен указателем "идентификатор". Выражение c содержит неизвестные; p - десятичная константа.

   Спецификация приема имеет вид "тип(мощности)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "или(не(заголовок(d 0))и(не(заголовок(b 0)) не(заголовок(a 0))))", "или(не(заголовок(b 0))и(не(заголовок(a 0)) не(заголовок(c 1))))". Прием не прорабатывался.

   Число приемов данного типа - 3.

5. Вывод параметрического описания неизвестной.

   Пример:

   $\forall_x(\exists_y(x = \cos y$ & $0 \ldots y$ & $y \ldots \pi$ & $y -$ число$))$

   Указатель "контрольвывода" инициирует попытку применения приема при усмотрении в условии (уравнении) задачи на описание подвыражения $\sqrt{1-x^2}$. Данное подвыражение домножено на x. Переменная x - неизвестная.

   Спецификация приема имеет вид "тип(суммаряда)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "условие", "тип(описать)", "неизвестная(x)". Прием не прорабатывался.

   Число приемов данного типа - 6.

Разбор случаев в условиях задачи на описание

1. Разбор случаев в условиях задачи на описание, связанный со значением неизвестного подтерма.

   Пример:

   $\forall_{xn}(x \leq n \ \& \ x - \text{натуральное} \rightarrow \exists_m(m \in \{1, \ldots, n\} \ \& \ x = m))$

   Первый антецедент идентифицируется с условием задачи на описание, второй - обрабатывается проверочным оператором. Выражение x содержит неизвестные; переменная n идентифицирована с натуральной константой, меньшей 6. Задача имеет натуральную неизвестную, входящую в x. Все ее неизвестные, не входящие в x, численные. Квантор существования разворачивается в дизъюнкцию. Эта дизъюнкция сопровождается комментарием "разборслучаев".

   Спецификация приема имеет вид "тип(усмнатуральное)", "неизвестная(x)", где x - переменная, идентифицируемая с содержащим неизвестные выражением. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "не(известно(x))". Требуется проработка.

   Число приемов данного типа - 6.

2. Разбор случаев по параметрическим описаниям неизвестной.

   Пример:

   $\forall_{abcn}(a - \text{целое} \ \& \ b - \text{целое} \ \& \ c - \text{целое} \ \& \ x - \text{целое} \ \& \ \neg(a - \text{even}) \ \& \ ax^n + b = c \rightarrow \exists_k(k - \text{целое} \ \& \ x = 2k) \ \vee \ \exists_k(k - \text{целое} \ \& \ x = 2k + 1))$

   Последний антецедент идентифицируется с условием задачи на описание. Переменная x - неизвестная, n - натуральная константа. Усматривается четность либо нечетность c, а для x ни того, ни другого не усматривается. Выводимая дизъюнкция сопровождается комментарием "разборслучаев".

   Спецификация приема имеет вид "тип(простойоперанд)", "неизвестная(x)", где x - переменная, идентифицируемая с неизвестной. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "неизвестная(x)". Требуется проработка.

   Число приемов данного типа - 5.

3. Разбор случаев для исключения сложного подвыражения.

   (a) Разбор случаев в условиях задачи на описание, направленный на исключение текущего сложного подвыражения.

   Пример:

   $\forall_a(a \leq 0 \ \vee \ 0 < a)$

   Попытка применения приема инициируется при усмотрении в условии задачи на описание содержащего неизвестные подвыражения "модуль(a)".

Прием имеет целый ряд дополнительных фильтров. Выводимая дизъюнкция сопровождается комментарием "разборслучаев".

Спецификация приема имеет вид "тип(нормобъединение)", "терм(t)", где t - текущее сложное подвыражение. Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)". Создаются также указатели "контрольвывода(модуль(a))" и "примечание(разборслучаев)". Требуется проработка.

Число приемов данного типа - 18.

4. Разбор случаев в условиях задачи на описание, позволяющий установить связь между вспомогательными параметрами.

   Пример:

   $\forall_{abcdefghijkmn}(m - \text{целое}\ \&\ n - \text{целое}\ \&\ amb/c + d \le i\ \&\ i \le amb/c + e\ \&\ anb/c + g \le i\ \&\ i \le anb/c + h\ \&\ j = [(h-d)c/(ab)]\ \&\ k = [(e-g)c/(ab)] \to \exists_f(f \in \{0, \ldots, j+k\}\ \&\ n = m - j + f))$

   Неравенства идентифицируются с условиями задачи на описание. m, n - переменные; a, c - натуральные константы. Два последних равенства выделены указателем "идентификатор", причем после вычислений j, k оказываются десятичными константами. Квантор существования разворачивается в дизъюнкцию, каждый член которой фиксирует связь между параметрами m, n.

   Спецификация приема имеет вид "тип(усмпринадлежит)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)". Создаются также указатели "примечание(разборслучаев)" и "или(фикс(0) фикс(0 2 1))". Требуется проработка.

   Число приемов данного типа - 4.

5. Специальный разбор случаев в условиях задачи на описание.

   Пример:

   $\forall_a(a = 1\ \vee \neg(a - 1 = 0))$

   Попытка применения приема инициируется усмотрение в условии задачи на описания какого-то логарифма $\log_b c$, а также усмотрения комментария (нормлогарифм a), свидетельствующего о том, что какой-то ранее сработавший прием усмотрел целесообразность перехода к основанию логарифмов a. При этом не усматривается, что a отлично от единицы.

   Спецификация приема имеет вид "тип(учетконтекста)". Справочник "заголовокприема" указывает уровень срабатывания 1, вводит фильтры "условие", "тип(описать)" и указатель "примечание(разборслучаев)". Он не прорабатывался.

   Число приемов данного типа - 3.

Вывод ограничения на известные параметры

1. Вывод ограничения на известные параметры при редактировании ответа задачи на описание.

   Пример:

   $\forall_{abcd}(d = (b < c)\ \&\ b \le x\ \&\ x < c \to d)$

   Два последних антецедента идентифицируются с условиями задачи на описание, имеющей цель "редакция", т.е. решаемой для редактирования ответа. Переменная x - неизвестная; выражения b, c неизвестных не содержат, причем хотя бы одно из них неконстантное. Первый антецедент выделен указателем "идентификатор". Его правая часть обрабатывается нормализатором "стандменьше", предпринимающим попытки упростить неравенство для известных параметров. В частности, если это не приводит к существенным усложнениям, разрешить его относительно какого-либо параметра.

   Спецификация приема имеет вид "тип(одзоператора)", "оператор(S)", где S - название нормализатора, обрабатывающего условие на параметры. Справочник "заголовокприема" указывает уровень срабатывания 4 и вводит фильтры "условие", "тип(описать)", "цель(редакция)", "не(цель(описатель))", "неизвестная(x)", "известно(b)", "известно(c)", "или(не(константа(b)) не(константа(c)))", "или(заголовок(d ложь) не(константа(d)))", "коммент(связьпеременная x)". Создаются также указатели "примечание(стандменьше)", "замечание(связьпеременная x)", "идентификатор(1)", причем для обработки правой части первого антецедента вводится нормализатор "стандменьше". Этого почти достаточно.

   Число приемов данного типа - 3.

14.2.7 Исключение несущественных неизвестных

1. Исключение несущественных неизвестных в задаче на описание при невырожденном ограничении.

   Данная операция выполняется приемами с заголовком "связка". Прием удостоверяется в том, что все вхождения в условия задачи на описание некоторых несущественных неизвестных X идентифицируются с заданными в теореме приема утверждениями, после чего удаляет все такие условия, а вместо них помещает утверждения без неизвестных X, эквивалентные существовованию требуемых значений исключенных X.

   Исключаемые неизвестные суть переменные связывающей приставки квантора существования в теореме приема, эталоны для поиска содержащих эти неизвестные условий задачи - конъюнктивные члены утверждения под квантором существования.

   Пример:

   $\forall_{abc}(\exists_c(a < c\ \&\ c < b\ \&\ c - \text{число}) \leftrightarrow a < b)$

Прием применяется к группе условий задачи на описание, которая образована всеми условиями, содержащими несущественную (т.е. указанную в цели "параметры ...") неизвестную c. Эта неизвестная не входит в a, b.

Спецификация приема имеет вид "тип(удалениеусловия)". Справочник "заголовокприема" указывает уровень срабатывания 0 и вводит фильтры "условие", "корень", "тип(описать)", "не(входит(c a))", "не(входит(c b))". Этого почти достаточно.

Число приемов данного типа - 88.

2. Исключение несущественных неизвестных в задаче на описание при вырожденном ограничении.

   Отличие от предыдущего пункта состоит в том, что заменяющее условие тождественно истинное, т.е. консеквентом теоремы приема служит не эквивалентность с квантором существования в левой части, а сам квантор существования. Пример:

   $\forall_b(b - \text{set} \rightarrow \exists_a(a - \text{set} \ \& \ b \subseteq a))$

   Прием применяется к группе условий задачи на описание, которая образована всеми условиями, содержащими несущественную неизвестную a. Эта неизвестная не входит в b. Отсутствует цель "независит ...".

   Спецификация приема имеет вид "тип(обрывзадачи)". Справочник "заголовокприема" указывает уровень срабатывания 3 и вводит фильтры "условие", "корень", "тип(описать)", "не(входит(a b))", "не(Входит(независит цели))". Этого достаточно.

   Число приемов данного типа - 64.

14.2.8 Обратный вывод

Приемы данного раздела имеют заголовок "подборзначений". Их консеквент идентифицируется с одним либо несколькими условиями задачи на описание. Те антецеденты, которые играют роль заменяющих условий, выделены в приеме указателем "подборзначений($i_1 \ldots i_m$)". Прием лишь предпринимает попытку свести текущую задачу к другой задаче и решить последнюю. При неудаче происходит откат и продолжение сканирования текущей задачи.

1. Непосредственный подбор примера значений неизвестных, не входящих в невырожденные условия.

   Пример:

   $\forall_{ax}(x = \emptyset \rightarrow x \subseteq a)$

   Прием применяется к условию задачи на описание, имеющей цель "пример". x - неизвестная, причем любое другое содержащее x условие задачи либо имеет длину 1, либо имеет вид отрицания равенства x некоторому выражению, не

являющемуся символом пустого множества. Антецедент выделен указателем "подборзначений".

Спецификация приема имеет вид "тип(подборзначений)", "неизвестная(x)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "цель(пример)", "неизвестная(x)", "не(контекст(новоеусловие(x2) входит(x x2) не(длинатекста(x2 2)) не(входит(терм(x2) Одз(корень))) не(контекст(вид(x2 не(равно(x x3))) не(равно(x3 фикс(1 2)))))))". Создается также указатель "подборзначений(1)". Этого достаточно.

Число приемов данного типа - 45.

2. Подбор параметризованного примера значения неизвестной.

   Параметризованные примеры предоставляют больше степеней свободы для реализации прочих условий задачи.

   Пример:

   $\forall_f(\text{последовательность}(f, \mathbb{R})\ \&\ \exists_{ab}(a - \text{число}\ \&\ b - \text{число}\ \&\ \neg(a = b)\ \&\ f = \lambda_n((a \text{ при } n - \text{even},\ \text{иначе } b), n - \text{натуральное})))$

   Прием применяется к условию задачи на описание, имеющей цель "пример" либо не имеющей цели "полный". Переменная f - неизвестная. Отстутствует условие, содержащее как f, так и символ "частичнпредел". Второй антецедент выделен указателем "подборзначений".

   Спецификация приема имеет вид "тип(повтор)", "неизвестная(x)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "или(цель(пример) не(цель(полный)))", "неизвестная(f)". Создаются также указатели "подборзначений(2)", "замечание(2 серия)". Этого почти достаточно.

   Число приемов данного типа - 5.

3. Попытка подбора примера значения неизвестной задачи на описание.

   Приемы этого типа, в отличие от рассмотренных выше, предпринимают попытку подбора значения неизвестной без какого-либо предварительного учет прочих условий на эту неизвестную. Пример:

   $\forall_x(x = \pi/2 \to \neg(\sin x = 0))$

   Прием применяется к условию задачи на описание, имеющей цель "пример". Переменная x идентифицируется с неизвестной. Антецедент выделен указателем "подборзначений".

   Спецификация приема имеет вид "тип(смешаннаядробь)", "неизвестная(x)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "условие", "тип(описать)", "цель(пример)", "неизвестная(x)". Создается также указатель "подборзначений(1)". Этого достаточно.

   Число приемов данного типа - 8.

4. Подбор примера значений неизвестных с помощью вычислений.

   Приемы этого типа для получения примера обращаются к цепочке вспомогательных вычислений - с помощью вычислительных пакетов, пакетных синтезаторов, нормализаторов или вспомогательных задач. Пример:

   $\forall_{abcdx}(\text{смпредст}(a,b,c,d)\ \&\ d=0\ \&\ x=b \to \text{систпредст}(a,c))$

   Прием применяется к условию задачи на описание. Переменная x - неизвестная, a - конечный набор конечных списков десятичных чисел. Первые два антецедента выделены указателем "программа", последний - указателем "подборзначений". Первый антецедент обрабатывается вычислительным пакетом, определяющим систему b различных представителей для a. Если ее найти удается, то d становится равно 0. В противном случае d равно 1, а c - набор номеров множеств набора a, имеющих в совокупности меньше элементов, чем число данных множеств. На неизвестную x накладываются только такие дополнительные условия, которые имеют вид одноместных предикатов либо отрицаний равенств для x.

   Спецификация приема имеет вид "тип(десзапись)", "неизвестная(x)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "условие", "тип(описать)", "цель(пример)", "неизвестная(x)", "не(контекст(новоеусловие(x5) входит(x x5) не(длинатекста(x5 2)) не(контекст(вид(x5 не(равно(x x6))) не(равно(x6 фикс(3 2)))))))", "конец(не(контекст(ключ(цели независит x5) пересекаются(параметры(b) x5))))". Он не прорабатывался.

   Число приемов данного типа - 9.

5. Обратный вывод с переходом к более простым понятиям при подборе примера.

   Пример:

   $\forall_{mn}(m - \text{целое}\ \&\ n - \text{целое}\ \&\ 0 < n - m\ \&\ 0 < m \to \neg(n|m))$

   Прием применяется к содержащему неизвестные условию задачи на описание, имеющей цель "пример". Последние два антецедента выделены указателем "подоборзначений", причем обратные неравенства не усматриваются.

   Спецификация приема имеет вид "тип(перечислфрагментов)", "подборзначений(...)". Справочник "заголовокприема" указывает уровень срабатывания 1 и вводит фильтры "условие", "тип(описать)", "цель(пример)". Создается также указатель "подборзначений(3 4)". Требуется доработка.

   Число приемов данного типа - 8.

6. Обратный вывод в задаче на описание, использующий кванторную импликацию из контекста.

   Пример:

   $\forall_{Aabcft}(A(t)\ \&\ \forall_x(A(x) \to 0 \le a + f(x))\ \&\ 0 \le b - ac\ \&\ 0 < c \to 0 \le b + cf(t))$

   Прием применяется к условию задачи на описание, имеющей цель "пример". Второй антецедент идентифицируется с утверждением из контекста, третий

- выделен указателем "подборзначений". Переменные A, f функциональные. Указатель "контекст" определяет дополнительную идентификацию внутри консеквента второго антецедента подвыражения вида $h(x)$, а внутри текущего условия - подвыражения $h(t)$. Здесь h - какая-то обычная переменная.

Спецификация приема имеет вид "тип(тройнойинтеграл)", "подборзначений(...)". Справочник "заголовокприема" указывает уровень срабатывания 5 и вводит фильтры "условие", "тип(описать)", "цель(пример)", "корень". Создается также указатель "подборзначений(3)". Требуется проработка.

Число приемов данного типа - 3.

14.2.9 Создание комментариев

Пример:

$\forall_{ABC}(\text{актив}(\angle(BAC)) \ \& \ \text{актив}(l(BC)) \rightarrow \emptyset)$

Прием применяется в посылках задачина доказательство либо на исследование. Угол BAC известен, расстояние $l(BC)$ имеет тип "неизв", а расстояние $l(AB)$ - не известно. Прием вводит комментарий к посылкам (пассив $l(AB)$ 0 фикс(1) 4). Этот комментарий означает, что вес посылки, идентифицированной с первым антецедентом, должен быть понижен до 0 сразу, как только будет вычислено $l(AB)$.

Спецификация приема имеет вид "тип(значение)". Справочник "заголовокприема" указывает уровень срабатывания 1. Он не прорабатывался.

Число приемов данного типа - 3.

14.2.10 Пакетные операторы

Проверочный оператор

Пример:

$\forall_{abc}(a \in b \ \& \ a \in c \rightarrow a \in b \cap c)$

Прием относится к проверочному оператору "усмпринадлежит". Антецеденты обрабатываются тем же оператором.

Спецификация приема имеет вид "тип(спуск)", "оператор(P)", где P - название проверочного оператора. Справочник "заголовокприема" указывает уровень срабатывания 2 и вводит указатели "блокпроверок(1 2)", "дистрибразвертка(фикс(0 2))", "спуск", "комментарий(1 склейкаоперандов пересечение)", "комментарий(2 склейкаоперандов пересечение)". Этого достаточно.

Число приемов данного типа - 2294.

Нормализатор

1. Нормализатор общей стандартизации выражений.

(a) Прием нормализатора общей стандартизации.

Пример:

$\forall_{ab}(b \subseteq a \to a \cap b = b)$

Прием применяется в нормализаторе "нормпересечение". Антецедент обрабатывается проверочным оператором. Отсутствует комментарий "существпосылки", блокирующий применение приемов, имеющих существенные антецеденты.

Спецификация приема имеет вид "тип(норм)", "направл(N)", "оператор(P)", где P - название нормализатора. Справочник "заголовокприема" вводит фильтр "коммент(существпосылки)" и указатели "уровень(2)", "блокпроверок(1)". Этого достаточно.

Число приемов данного типа - 837.

(b) Вычисления в нормализаторе общей стандартизации.

Пример:

$\forall_{abcd}(d = b + c \to ab + ac = ad)$

Прием применяется в нормализаторе "нормплюс". Переменные b, c идентифицируются с десятичными константами. Выражение a не содержит символов "О", "плюсбеск", "минусбеск". Антецедент реализуется путем непосредственных вычислений.

Спецификация приема имеет вид "тип(усмубывает)", "направл(N)", "оператор(P)", "типданных($rx_1 \ldots x_k$)". Здесь P - название нормализатора; r - тип значения переменных $x_1, \ldots x_k$. Элементов последнего вида может быть несколько. Справочник "заголовокприема" вводит фильтры "десчисло(b)", "десчисло(c)", "постпозиция(фикс(0 1 2) фикс(0 1 1))". Создаются также указатели "уровень(2)", "программа(1)". Этого достаточно.

Число приемов данного типа - 15.

(c) Один шаг развертки операции над конечным семейством в нормализаторе общей стандартизации.

Пример:

$$\forall_{Pabf}(P(a) \to \bigcup_{i,i\in\{a;b\},P(i)} f(i) = f(a) \cup \bigcup_{i,i\in\{;b\},P(i)} f(i))$$

Прием применяется в нормализаторе "нормобъединениевсех". Список b непуст. Антецедент проверяется с помощью задачи на доказательство, решаемой до малого уровня. Переменные f, P функциональные.

Спецификация приема имеет вид "тип(облвхожд)", "направл(N)", "оператор(P)". Справочник "заголовокприема" вводит фильтр "коммент(существпосылки)". Создаются также указатели "уровень(2)", "усматривается(1)". Этого почти достаточно.

Число приемов данного типа - 5.

(d) Устранение вложенных операций в нормализаторе общей стандартизации.

Пример:

Теорема приема - "коммутативно(плюс)"; заголовок - "замена(спускоперандов нормплюс)".

Спецификация приема имеет вид "тип(нормцелаячасть)". Создается указатель "уровень(1)". Этого достаточно.

Число приемов данного типа - 5.

(e) Лексикографическое упорядочение операндов в нормализаторе общей стандартизации.

В качестве примера рассмотрим прием, у которого теорема - та же, что и выше. Заголовок приема - "замена(лексупорядочение нормплюс)".

Спецификация приема имеет вид "тип(Сокращение)". Создается указатель "уровень(3)". Этого достаточно.

Число приемов данного типа - 5.

(f) Прием "усмотрение из посылок" для прямой ориентации равенства.

Пример:

$\forall_{ab}(a = b \to a = b)$

Прием применяется в нормализаторе "нормкоорд". Выражение a имеет заголовок "коорд". Антецедент идентифицируется с посылкой, причем выражение a не является подвыражением выражения b. При идентификации частей равенства перестановка их не допускается.

Спецификация приема имеет вид "тип(поразныестороны)", "оператор(P)", "символ(s)", где s - заголовок преобразуемого терма. Создаются фильтры "заголовок(a коорд)", "не(вхождениетерма(b a))". Вводится также указатель "коммутативно(фикс(1))". Этого достаточно.

Число приемов данного типа - 23.

(g) Прием нормализатора общей стандартизации, использующий посылку.

Пример:

$\forall_{ab}(a - b = 0 \to b - a = 0)$

Прием применяется в нормализаторе "нормплюс". Антецедент идентифицируется с посылкой. Выражение $-a$ идентифицируется с подсуммой, образованной всеми слагаемыми со знаком "минус". Аналогичным образом идентифицируется $-b$.

Спецификация приема имеет вид "тип(нижнийпредел)", "направл(N)", "антецедент(i)". Создается указатель "нормзнака(b минус плюс)". Требуется доработка.

Число приемов данного типа - 5.

(h) Прием нормализатора общей стандартизации, учитывающий дополнительную целевую установку.

Пример:

$\forall_{ab}(\sqrt{a^2b} = |a|\sqrt{b})$

Прием применяется в нормализаторе "нормстепень", имеющем комментарий "сбросмодуля".

Спецификация приема имеет вид "тип(сигнум)", "направл(N)". Справочник "заголовокприема" никаких фильтров не создает. Он не прорабатывался.

Число приемов данного типа - 3.

(i) Стандартизация константного выражения в нормализаторе общей стандартизации.

Пример:

$\forall_{abcde}(a/b = e \rightarrow ad/(bc) = ed/c)$

Прием применяется в нормализаторе "нормдробь". Выражения a, b суть десятичные константы. Оператор "сокращдроби" предпринимает попытку сократить дробь a/b и убеждается, что результат e отличен от исходной дроби.

Спецификация приема имеет вид "тип(регтеор)", "оператор(P)", "направл(N)". Справочник "заголовокприема" создает указатели "идентификатор(1)", "операнд(a фикс(0 1 1))", "операнд(b фикс(0 1 2))". Он не прорабатывался.

Число приемов данного типа - 15.

2. Нормализатор общей стандартизации утверждений.

(a) Непосредственное усмотрение истинности либо ложности в нормализаторе общей стандартизации.

Пример:

$\forall_{ab}(a < b \rightarrow a < b)$

Прием применяется в нормализаторе "нормменьше". Антецедент обрабатывается проверочным оператором, которому передается комментарий "нормализатор". Отсутствует комментарий "нормусм". Эти комментарии вводятся для предовращения зацикливаний при встречных обращениях из проверочных операторов к нормализаторам.

Спецификация приема имеет вид "тип(первыйтерм)", "оператор(P)". Справочник "заголовокприема" никаких фильтров не создает, хотя и вводит указатель "блокпроверок(1)". Требуется доработка.

Число приемов данного типа - 9.

(b) Прием нормализатора общей стандартизации утверждений.

Пример:

$\forall_{abc}(a \in b \cap c \leftrightarrow a \in b \ \& \ a \in c)$

Прием применяется в нормализаторе "нормпринадлежит".

Спецификация приема имеет вид "тип(второйтерм)", "оператор(P)", "направл(N)". Справочник "заголовокприема" никаких фильтров и указателей не создает; впрочем, в этом случае они и не нужны. Требуется небольшая доработка.

Число приемов данного типа - 95.

3. Нормализатор сокращенной перезаписи.

(a) Прием нормализатора сокращенной перезаписи.

Пример:

$\forall_{abc}(a^c b^c = (ab)^c)$

Прием применяется в нормализаторе "упрощумножение". Есть ряд ограничений, связанных с контекстом применения нормализатора.

Спецификация приема имеет вид "тип(нормупростить)", "направл(N)". Справочник "заголовокприема" ограничивается вводом указателя "общаястепень(общаястепень c a^c b^c)". Требуется доработка.

Число приемов данного типа - 101.

(b) Общая стандартизация в нормализаторе сокращенной перезаписи.

Иногда нормализатор сокращенной перезаписи создает ситуации, в которых могут применяться приемы общей стандартизации. Такие приемы тоже заносятся в данный нормализатор.

Пример:

$\forall_{abc}((a/b)/c = a/(bc))$

Прием применяется в нормализаторе "упрощдробь".

Спецификация приема имеет вид "тип(указательтипа)", "оператор(P)", "направл(N)". Справочник "заголовокприема" никаких фильтров и указателей не создает; впрочем, в этом случае они и не нужны.

Число приемов данного типа - 5.

(c) Лексикографическое упорядочение операндов в нормализаторе сокращенной перезаписи.

Пример:

"коммутативно(плюс)". Заголовок приема - "замена(лексупорядочение упрощплюс)".

Прием применяется в нормализаторе "упрощплюс".

Спецификация приема имеет вид "тип(нормпроизводная)". Справочник "заголовокприема" никаких фильтров и указателей не создает; впрочем, в этом случае они и не нужны.

Число приемов данного типа - 4.

(d) Устранение вложенных операций в нормализаторе сокращенной перезаписи.

Пример:

"коммутативно(пересечение)". Заголовок приема - "замена(спускоперандов упрощпересечение)".

Спецификация приема имеет вид "тип(вхождениемыши)". Справочник "заголовокприема" никаких фильтров и указателей не создает; впрочем, в этом случае они и не нужны.

Число приемов данного типа - 4.

(e) Вычисления с константами в нормализаторе сокращенной перезаписи.

Пример:

$\forall_{ab}(a = b^2 \to \sqrt{a} = b)$

Прием применяется в нормализаторе "упрощумножение". Переменная a идентифицируется с десятичной константой. Антецедент выделен указателем "программа". Он предпринимает попытку извлечь квадратный корень.

Спецификация приема имеет вид "тип(верхнийпредел)", "оператор(P)", "направл(N)". Справочник "заголовокприема" создает указатель "идентификатор(1)". Он не прорабатывался.

Число приемов данного типа - 3.

4. Нормализатор приведения к заданным заголовкам.

(a) Непосредственное преобразование к нужному заголовку.

Пример:

$\forall_{ab}(-b^2 + a^2 = (a + b)(a - b))$

Пример применяется в нормализаторе "видумножение". Дополнительные слагаемые преобразуемого терма отсутствуют.

Спецификация приема имеет вид "тип(нормзаголовок)", "оператор(P)", "направл(N)". Справочник "заголовокприема" создает фильтр "коммент(длинаменее)" и указатель "модификатор". Этого почти достаточно.

Число приемов данного типа - 161.

(b) Попытка группировки нескольких корневых операндов.

Пример:

$\forall_{abcde}(e = a(b + c)^2 + d \to ab^2 + 2abc + ac^2 + d = e)$

Прием применяется в нормализаторе "видумножение". Антецедент выделен указателем "идентификатор". Для обработки его правой части предпринимается рекурсивное обращение к нормализатору "видумножение", с блокировкой раскрывания скобок. Проверяется, что результат e - произведение, дробь либо степень (с точностью до знака).

Спецификация приема имеет вид "тип(группировки)", "оператор(P)", "направл(N)". Справочник "заголовокприема" создает фильтры "постпозиция(фикс(0 1 3)фикс(0 1 1))", "или(не(заголовок(b 1))не(заголовок(c 1)))". Он не прорабатывался.

Число приемов данного типа - 37.

(c) Изменение заголовка неизвестного подвыражения, обеспечивающее возможность изменения заголовка всего выражения.

Пример:

$\forall_{acd}(a = d \to c \cap d = c \cap a)$

Прием применяется в нормализаторе "видобъединение". Выражение d содержит неизвестные (они указаны в комментарии нормализатора), отлично от переменной и не имеет заголовка "объединение". Антецедент выделен указателем "идентификатор". Его правая часть обрабатывается нормализатором "нормобъединение". Результат a либо имеет заголовок "объединение", либо короче выражения d.

Спецификация приема имеет вид "тип(числоповторений)", "норм(Q)", "оператор(P)", "направл(N)". Здесь Q - заголовок оператора, используемого при обработке подвыражения. Справочник "заголовокприема" создает фильтры "не(известно(d))", "не(переменная(d))", "не(заголовок(d объединение))", "или(заголовок(a объединение) короче(a d))", "не(подобные-термы(a копия(d)))". Создается также указатель "идентификатор(1)" и вводится обращение к нормализатору, обрабатывающему правую часть антецедента. Этого достаточно.

Число приемов данного типа - 4.

(d) Дополнительное преобразование к требуемому заголовку корневых операндов после того, как само выражение уже имеет нужный заголовок.

Пример:

$\forall_{abc}(a = b \to ac = bc)$

Прием применяется в операторе "видумножение". Антецедент выделен указателем "идентификатор". Он обращается к оператору "видумножение" для доразложения a на множители.

Спецификация приема имеет вид "тип(внутрзамена)", "оператор(P)", "направл(N)". Справочник "заголовокприема" создает указатель "идентификатор(1)", но и только. Он не прорабатывался.

Число приемов данного типа - 7.

(e) Усиленное преобразование к заданному заголовку при наличии специальной цели.

Пример:

$\forall_{abx}(ax^3+b=(\sqrt[3]{a}x+\sqrt[3]{b})(\sqrt[3]{a^2}x^2-\sqrt[3]{a}x\sqrt[3]{b}+\sqrt[3]{b^2}))$

Прием применяется в нормализаторе "видумножение", имеющем комментарий "нормИнтеграл x". Такой комментарий означает, что обращение к нормализатору произошло при вычислении неопределенного интеграла по x. Проверяется, что выражения a, b не содержат x.

Спецификация приема имеет вид "тип(подобныечлены)", "оператор(P)", "направл(N)". Справочник "заголовокприема" никаких фильтров не создает. Он не прорабатывался.

Число приемов данного типа - 3.

(f) Вычисления с константами в нормализаторе приведения к заданным заголовкам.

Пример:

$\forall_{abcd}(c=a+b\to a+b+d=c+d)$

Прием применяется в нормализаторе "видумножение". Переменные a, b идентифицируются с десятичными константами. Антецедент, выделенный указателем "идентификатор", реализует их сложение.

Спецификация приема имеет вид "тип(элементнабора)", "оператор(P)", "направл(N)". Справочник "заголовокприема" никаких фильтров не создает. Он не прорабатывался.

Число приемов данного типа - 4.

(g) Преобразование в нормализаторе приведения к заданным заголовкам, подготавливающее возможность непосредственного перехода к этим заголовкам.

Пример:

$\forall_{abc}(\sqrt{3}c\sin a-c\cos a+b=2c\sin(a-\pi/6)+b)$

Прием применяется в нормализаторе "видумножение". Выражение b либо равно 0, либо имеет вид $2c\cdot A$, где заголовок A - синус либо косинус.

Спецификация приема имеет вид "тип(транзитивно)", "оператор(P)", "направл(N)". Справочник "заголовокприема" никаких фильтров не создает. Он не прорабатывался.

Число приемов данного типа - 8.

(h) Корневая свертка выражения в нормализаторе приведения к заданным заголовкам.

Пример:

$\forall_{ab}(\sin a\cos b+\sin b\cos a=\sin(a+b))$

Прием применяется в нормализаторе "видумножение".

Спецификация приема имеет вид "тип(конгруэнтны)", "оператор(P)", "направл(N)". Справочник "заголовокприема" создает указатель "модификатор". Требуется доработка.

Число приемов данного типа - 5.

(i) Попытка варьирования выражения в нормализаторе приведения к заданным заголовкам.

Пример:

$\forall_{abcdef}(d = \sin(b/2)\ \&\ e = \cos(b/2)\ \&\ f = 2ade + c \to a\sin b + c = f)$

Прием применяется в нормализаторе "видумножение". Антецеденты выделены указателем "идентификатор", причем последний из них выпооняет рекурсивное обращение к нормализатору для разложения на множители проварьированного выражения. Предварительно проверяется, что хотя бы одно из слагаемых выражения c имеет своим сомножителем d либо e. Проверяется, что попытка разложения проварьированного выражения оказалась успешной.

Спецификация приема имеет вид "тип(нормлогарифм)", "оператор(P)", "направл(N)". Справочник "заголовокприема" каких-либо фильтров и рекурсивного обращения не создает. Он не прорабатывался.

Число приемов данного типа - 9.

(j) Блокировка дальнейших преобразований выражения с неизвестными.

Пример:

$\forall_{abcx}(a\sin x\cos x + b\sin x + b\cos x + c = a\sin x\cos x + b\sin x + b\cos x + c)$

Прием применяется в нормализаторе "видумножение". Выражение x содержит неизвестные; выражения a, b, c - не содержат. Усматривается вид левой части уравнения, для решения которого имеются приемы. Поэтому дальнейшие преобразования не нужны, и указатель "выход" определяет немедленную выдачу результата.

Спецификация приема имеет вид "тип(веществмногочлен)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 3.

(k) Предварительная стандартизация в нормализаторе приведения к заданным заголовкам.

Пример:

$\forall_{abcdef}(f = c\ \&\ 0 \le c \to a + bc^d/e = a + bf^d/e)$

Прием применяется в нормализаторе "видумножение". Заголовок выражения c - символ "плюс". Первый антецедент выделен указетелем "идентификатор". Его правая часть обрабатывается нормализатором "видумножение". Проверяется успешность попытки обращения.

Спецификация приема имеет вид "тип(стандартизация)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 123.

5. Нормализатор стандартной формы.

Напомним, что такой нормализатор обеспечивает преобразование выражения к заданной стандартной форме (например, к виду многочлена, д.н.ф., к.н.ф. и т.п.) и упрощение его в рамках данной стандартной формы.

(a) Преобразование к стандартной форме.

Пример:

$\forall_{abc}(ab + ac = a(b + c))$

Замена выполняется справа налево. Допускается произвольное число слагаемых. Прием применяется в нормаизаторе "стандплюс", выполняющем раскрывание скобок и приведение подобных членов. В зависимости от опций обращения, могут применяться также другие преобразования (например, переход от произведения тригонометрических функций к сумме).

Спецификация приема имеет вид "тип(стандлогарифм)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 34.

(b) Упрощение в стандартной форме.

Пример:

$\forall_{ab}(a \subseteq b \to a \cup b = b)$

Прием применяется в нормализаторе "стандобъединение".

Спецификация приема имеет вид "тип(общаяплоскость)", "оператор(P)", "направл(N)". Справочник "заголовокприема" создает фильтры "коммент(существпосылки)", "коммент(нормализация)". Этого достаточно.

Число приемов данного типа - 128.

(c) Устранение вложенных операций в нормализаторе стандартной формы.

Пример:

Теорема приема - "коммутативно(умножение)", заголовок - "замена(спуск-операндов стандплс)". Прием устраняет вложенные умножения в нормализаторе раскрывания скобок.

Спецификация приема имеет вид "тип(нормминимум)", "оператор(P)". Справочник "заголовокприема" указывает только уровень срабатывания 1. Этого достаточно.

Число приемов данного типа - 4.

(d) Лексикографическое упорядочение в нормализаторе стандартной формы.

Пример:

Теорема приема - "коммутативно(умножение)", заголовок - "замена(лексупорядочение стандплс)". Прием лексикографически упорядочивает сомножители в нормализаторе раскрывания скобок.

Спецификация приема имеет вид "тип(норминфимум)", "оператор(P)". Справочник "заголовокприема" указывает только уровень срабатывания 4. Этого достаточно.

Число приемов данного типа - 4.

(e) Вычисления с константами в нормализаторе стандартной формы.

Пример:

$\forall_{abcd}(c = ab \to abd = cd)$

Прием применяется в нормализаторе раскрывания скобок "стандплюс". Переменные a, b идентифицируются с десятичными константами. Антецедент выделен указателем "идентификатор". Он выполняет умножение этих констант.

Спецификация приема имеет вид "тип(многочлень)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 13.

(f) Специальная стандартизация в нормализаторе стандартной формы.

Пример:

$\forall_{abcde}(0 < b - c \;\&\; b = cd + e \;\&\; 0 \leq a \to a^{b/c} =^d a^{e/c})$

Прием применяется в нормализаторе "стандплюс". Переменные b, c идентифицируются с натуральными константами, причем c - четное. Первые два антецедента выделены указателем "программа", причем второй выполняет деление с остатком. Происходит вынесение натуральной степени из-под радикала.

Спецификация приема имеет вид "тип(нормугол)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 7.

6. Нормализатор упрощения относительно неизвестных.

(a) Прием нормализатора упрощения относительно неизвестных.

Пример:

$\forall_{abc}(c - \text{rational} \;\&\; \neg(\text{знаменатель}(c) - \text{even}) \to a^c b^c = (ab)^c)$

Прием применяется в нормализаторе "уравнумножение". Выражения a, b не содержат неизвестных, выражение c - содержит.

Спецификация приема имеет вид "тип(сокращнеизв)", "оператор(P)", "направл(N)", "неизвестные(...)". Справочник "заголовокприема" вводит фильтры "не(известно(c))", "известно(a)", "известно(b)" и указатели "блокпроверок(1 2)", "общаястепень(общаястепень c a^c b^c)". Этого почти достаточно.

Число приемов данного типа - 31.

(b) Стандартизация операнда в нормализаторе упрощения относительно неизвестных.

Напомним, что нормализатор упрощения относительно неизвестных - корневой. Поэтому имеет смысл предпринимать рекурсивные обращения для обработки подтермов. Пример:

$\forall_{abc}(c = a \to a/b = c/b)$

Прием применяется в нормализаторе "уравндробь". Антецедент выделен указателем "идентификатор", и его правая часть обрабатывается оператором "нормуравн", который по заголовку выражения a определяет конкретный используемый нормализатор упрощения относительно неизвестных.

Спецификация приема имеет вид "тип(унисборка)", "оператор(P)", "направл(N)". Справочник "заголовокприема" вводит фильтры "не(равно(a c))", "входит(нормуравн комментарии)". Этого почти достаточно.

Число приемов данного типа - 11.

(c) Общая стандартизация в нормализаторе упрощения относительно неизвестных.

Пример:

$\forall_{abc}(c \cdot a/b = ac/b)$

Прием применяется в нормализаторе "уравнумножение".

Спецификация приема имеет вид "тип(лексупорядочение)", "оператор(P)", "направл(N)". Справочник "заголовокприема" никаких фильтров и указателей не создает; впрочем, в этом случае они и не нужны.

Число приемов данного типа - 8.

7. Нормализатор упрощения выражений путем перебора группировок.

Пример:

$\forall_{abc}(c \setminus a \cup c \setminus b = c \setminus (a \cap b))$

Прием применяется в нормализаторе "группмножество". Он не выполняет немедленной замены, а лишь регистрирует в некотором накопителе результат упрощения выражения, возникшего при указанной группировке. По окончании перебора различных таких группировок нормализатор выберет группу замен, дающую наибольшее упрощение.

Спецификация приема имеет вид "тип(нормгрупп)", "оператор(P)", "направл(N)". Справочник "заголовокприема" создает единственный фильтр "постпозиция(фикс(0 1 2) фикс(0 1 1))". Этого достаточно.

Число приемов данного типа - 20.

8. Нормализатор явного разрешения утверждений с неизвестными.

   (a) Общая стандартизация утверждения в нормализаторе явного разрешения относительно неизвестных.

   Пример:

   $\forall_{ab}(0 < a \to 0 < ab \leftrightarrow 0 < b)$

   Прием применяется в нормализаторе "уравнменьше".

   Спецификация приема имеет вид "тип(контрсерия)", "оператор(P)", "направл(N)". Справочник "заголовокприема" никаких фильтров не создает; впрочем, в этом случае они и не нужны.

   Число приемов данного типа - 34.

   (b) Явное разрешение утверждения относительно неизвестных в нормализаторе уравнений.

   Пример:

   $\forall_{abc}(b < a + c \leftrightarrow b - c < a)$

   Прием применяется в нормализаторе "уравнменьше". Переменная a идентифицируется с непустой суммой всех содержащих неизвестные слагаемых. Выражение b неизвестных не содержит.

   Спецификация приема имеет вид "тип(внешкадр)", "оператор(P)", "направл(N)". Справочник "заголовокприема" вводит фильтры "не(известно(a))", "известно(b)", а также указатель "перечень(a не(известно(a)))". Этого достаточно.

   Число приемов данного типа - 77.

   (c) Группировка всех неизвестных в одной части двуместного отношения.

   Пример:

   $\forall_{ab}(a < b \leftrightarrow 0 < b - a)$

   Прием применяется в нормализаторе "уравнменьше". Выражения a, b содержат неизвестные.

   Спецификация приема имеет вид "тип(русшрифт)", "оператор(P)", "направл(N)". Справочник "заголовокприема" вводит фильтры "не(известно(a))", "не(известно(b))". Этого почти достаточно.

   Число приемов данного типа - 11.

(d) Дизъюнктивно-конъюнктивная декомпозиция утверждения в нормализаторе явного разрешения.

Пример:

$\forall_{ab}(0 < ab \leftrightarrow a < 0 \ \& \ b < 0 \ \vee \ 0 < a \ \& \ 0 < b)$

Прием применяется в нормализаторе "уравнменьше".

Спецификация приема имеет вид "тип(внешконтроль)", "оператор(P)", "направл(N)". Справочник "заголовокприема" никаких фильтров не создает; впрочем, в этом случае они и не нужны.

Число приемов данного типа - 36.

(e) Преобразование условия к виду, для которого существует стандартный разрешающий прием.

Пример:

$\forall_{ab}(\sqrt{3}\sin a - \cos a < b \leftrightarrow 2\sin(a - \pi/6) < b)$

Прием применяется в нормализаторе "уравнменьше". Выражение a не содержит неизвестных, выражение b - содержит.

Спецификация приема имеет вид "тип(выч)", "оператор(P)", "направл(N)", "неизвестные(...)". Справочник "заголовокприема" вводит фильтры "не(известно(a))", "известно(b)", "известно(c)". Этого почти достаточно.

Число приемов данного типа - 18.

(f) Непосредственное усмотрение истинности либо ложности условия.

Пример:

$\forall_{abc}(c < a \ \& \ 0 \leq b - a \rightarrow \neg(b < c))$

Прием применяется в нормализаторе "уравнменьше". Переменная c - неизвестная; выражения a и b не содержат неизвестных. Первый антецедент идентифицируется с посылкой, второй - обрабатывается проверочным оператором.

Спецификация приема имеет вид "тип(Выч)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 9.

(g) Эквивалентное преобразование условия, приводящее к цепи упрощений для подвыражений с неизвестными.

Пример:

$\forall_{abcde}(c = a \ \& \ e = d/2 \rightarrow a < b \leftrightarrow \neg(\cos e = 0) \ \& \ c < b \ \vee \ \cos e = 0 \ \& \ a < b)$

Прием применяется в нормализаторе "уравнменьше". Указатель "контекст" определяет дополнительную идентификацию в преобразуемом терме подвыражения $\cos d$, не являющегося основанием четной степени. Антецеденты выделены указателем "идентификатор". Правая часть первого из них

обрабатывается нормализатором "половинныйугол", выражающему a через тангенс $d/2$. Второй антецедент упрощает выражение $d/2$, обрабатывая его нормализаторами "нормдробь" и "стандплюс". Выражения a, d содержат неизвестные. Условие либо содержит выражение $\sin d$, не являющееся основанием четной степени, либо содержит $\operatorname{tg} e$. Оно не содержит других не известных тригонометрических операций, кроме $\sin d$, $\cos d$, $\operatorname{tg} d$, $\operatorname{tg} e$. Степени этих операций не выше 3.

Спецификация приема имеет вид "тип(второйсимвол)", "оператор(P)", "направл(N)", "неизвестные(...)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 4.

(h) Преобразование с помощью нормализатора приведения к заданным заголовкам, обеспечивающее декомпозицию утверждения.

Пример:

$\forall_{abc}(c = b - a \to a < b \leftrightarrow 0 < c)$

Прием применяется в нормализаторе "уравнменьше". Правая часть антецедента обрабатывается нормализатором разложения на множители "видумножение", а после этого - нормализатором выражений с неизвестными "уравнплюс". Неверно, что одна из частей неравенства нулевая, а другая имеет своим заголовком один из символов "умножение", "степень", "дробь". Неверно, что одна из частей неравенства - переменная, не являющаяся неизвестной. Целесообразность выполнения замены после разложения на множители оценивается оператором "фильтрмножителей".

Спецификация приема имеет вид "тип(сложить)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 10.

(i) Исключение сложных подвыражений с неизвестными.

Пример:

$\forall_{abcde}(0 < b \ \& \ \neg(b - 1 = 0) \to a \log_b c + d \leq e \leftrightarrow 0 < b - 1 \ \& \ 0 \leq b^e - c^a b^d \ \vee \ b - 1 < 0 \ \& \ b^e - c^a b^d)$

Прием применяется в нормализаторе "уравнменьшеилиравно". Хотя бы одно из выражений b, c содержит неизвестные. Фильтры приема проверяют, что после упрощения заменяющей части не остается степеней с показателями, содержащими неизвестные.

Спецификация приема имеет вид "тип(учетприменения)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 17.

(j) Стандартная схема вычислений в нормализаторе разрешения относительно неизвестных.

Пример:

$\forall_{abkmnrx}(r \in \{1,\ldots,\min(m,n)\}\ \&\ \neg(a(r,r)=0) \to \lambda_{ij}(a(i,j), i \in \{1,\ldots,m\}\ \&\ j \in \{1,\ldots,n\})x = \lambda_{pq}(b(p,q), p \in \{1,\ldots,m\}\ \&\ q \in \{1,\ldots,k\}) \leftrightarrow \lambda_{ij}(((0 \text{ при } j=r, \text{ иначе } (a(i,j)a(r,r)-a(i,r)a(r,j))/a(r,r)) \text{ при } \neg(i=r), \text{ иначе } a(i,j)), i \in \{1,\ldots,m\}\ \&\ j \in \{1,\ldots,n\})x = \lambda_{pq}(((b(p,q)a(r,r) - a(p,r)b(r,q))/a(r,r) \text{ при } \neg(p=r), \text{ иначе } b(p,q)), p \in \{1,\ldots,m\}\ \&\ q \in \{1,\ldots,k\}))$

Прием применяется в нормализаторе решения матричных уравнений "уравнматр". Он выполняет вычитание кратных заданной строки.

Спецификация приема имеет вид "тип(смрасст)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 15.

9. Нормализатор вычисления.

   (a) Непосредственное вычисление.

   Пример:

   $\forall_{abkn}(\neg(a-1=0)\ \&\ k - \text{целое}\ \&\ n - \text{целое}\ \&\ 0 \le n-k\ \&\ \neg(b=0) \to \sum_{m=k}^{n} a^{bm} = (a^{b(n+1)} - a^{bk})/(a^b - 1))$

   Прием применяется в нормализаторе вычисления конечных сумм "нормсуммавсех".

   Спецификация приема имеет вид "тип(буква)", "оператор(P)", "направл(N)". Справочник "заголовокприема" создает все необходимые указатели.

   Число приемов данного типа - 223.

   (b) Сведение к вычислению более простого выражения.

   Пример:

   $\forall_{afgh}(\sum_{x,f(x)} ag(x)/h(x) = a\sum_{x,f(x)} g(x)/h(x))$

   Прием применяется в нормализаторе "нормсуммавсех". Конечная сумма в заменяющей части обрабатывается тем же нормализатором.

   Спецификация приема имеет вид "тип(переменная)", "оператор(P)", "направл(N)". Справочник "заголовокприема" создает все необходимые указатели, однако не создает нужного фильтра. Требуется доработка.

   Число приемов данного типа - 356.

   (c) Декомпозиция вычисляемого выражения.

   Пример:

   $\forall_{abc}(\text{скалумнож}(a, b+c) = \text{скалумнож}(a,b) + \text{скалумнож}(a,c))$

   Прием применяется в нормализаторе "значскалумнож". Так как нормализатор некорневой, рекурсивные обращения для обработки скалярных произведений в заменяющей части не требуются.

   Спецификация приема имеет вид "тип(менее)", "оператор(P)", "направл(N)". Справочник "заголовокприема" в отдельных случаях требует доработки.

   Число приемов данного типа - 109.

(d) Попытка варьирования выражения в нормализаторе вычисления.

Пример:

$\forall_{abcfgx}(0 < f(x)\ \&\ c = \lim_{x\to b\setminus a}(\ln f(x)\cdot g(x)) \to \lim_{x\to b\setminus a}(f(x)^{g(x)}) = \exp c)$

Прием применяется в нормализаторе "нормпредел". Правая часть второго антецедента обрабатывается тем же нормализатором, причем результат c не содержит символа "предел".

Спецификация приема имеет вид "тип(клавиатура)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 35.

(e) Разбор случаев в нормализаторе вычисления.

Пример:

$\forall_{abcdfg}(a = \lim_{x\to c\setminus b} f(x)\ \&\ (a = \infty\ \vee\ a = -\infty)\ \&\ d = \lim_{x\to c\setminus b} g(x)\ \&\ d - \text{число} \to \lim_{x\to c\setminus b}(f(x)g(x)) = \text{разборслучаев}(d = 0\ \vee\ 0 < d\ \vee\ d < 0))$

Прием применяется в нормализаторе "нормпредел". Первый и третий антецеденты выделены указателем "идентификатор". Их правые части обрабатываются тем же самым нормализатором. Проверяется, что выражение d неконстантное. Прием указывает альтернативы для параметров, содержащихся в d. Если бы разбор случаев по этим альтернативам выполнялся непосредственно в рамках данного приема, то результат имел бы вид условного выражения. Вообще говоря, такой результат имел бы промежуточный характер и передавался бы внешнему приему. Однако, обычно внешний прием рассчитывает на безусловный результат, и при получении условного попросту не сработает. Поэтому указание на разбор случаев инициирует откат к повторной обработке "самого исходного" предела, где и разбираются данные альтернативы. Окончательный ответ получается последующей склейкой частных результатов в корневой точке дерева вычислений нормализатора.

Спецификация приема имеет вид "тип(кодтекста)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 12.

(f) Стандартизация в нормализаторе вычисления.

Пример:

$\forall_{abcdef}(\sum_{i,f(i)} \cos(ai/b + ci/d + e(i)) = \sum_{i,f(i)} \cos((a/b + c/d)i + e(i)))$

Прием применяется в нормализаторе "нормсуммавсех". Он выполняет группировку относительно варьируемого параметра, необходимую для вычисления суммы.

Спецификация приема имеет вид "тип(указатель)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 121.

(g) Ввод комментария, передающего информацию внешнему процессу.

Это - еще одна разновидность приема, инициирующего разбор случаев в пакетном нормализаторе. В отличие от приведенного выше способа разбора случаев, где откат выполнялся до корневой точки вычислений нормализатора, здесь инициирующий разбор случаев комментарий передается текущей задаче, из которой имелось обращение к нормализатору. Пример:

$\forall_{abdefgh}(\neg(a=0)\ \&\ 0\le b\ \&\ 0\le h\ \&\ \neg(e=0)\ \&\ \lim_{c\to g\setminus f} d(c)=\infty\ \&\ (c\to g\setminus f)\to \text{контекст}(ab^{d(c)}+eh^{d(c)}))$

Прием применяется в нормализаторе "асимптоценка". Переменная c не входит в выражения a, b, e, h. Не удается сравнить по величине b и h. Прием вводит комментарий (Случай $b=h\ \vee\ 0<b-h\ \&\ b-h<0$) к текущей задаче. При неудачной попытке найти асимптотическую оценку эта задача инициирует разбор случаев по данному комментарию.

Спецификация приема имеет вид "тип(набор)", "оператор(P)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 5.

10. Специальный нормализатор.

Ряд нормализаторов не попал в приводимую предварительную их классификацию. Они используются для достижения различных специальных целей (например, приведения выражения к виду суммы простейших дробей).

(a) Прием специального нормализатора.

Пример:

$\forall_{abcdefg}(c=d-b\ \&\ \neg(c=0)\to e/(f(a+bg)(a+dg))=e/(fcg(a+bg))-e/(fcg(a+dg)))$

Прием применяется в нормализаторе "простейшиедроби". При обращении к нему задается комментарий (переменная g), указывающая на варьируемую переменную. Проверяется, что выражение a представляет собой многочлен от g, степень которого больше 1.

Спецификация приема имеет вид "тип(Исключение)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался. В действительности он должен играть роль передаточного звена, так как вся информация о цели применения преобразования становится известной еще на этапе вывода теоремы.

Число приемов данного типа - 337.

11. Нормализатор упрощения дизъюнкции.

Для приведения к стандартному виду дизъюнкций неравенств создан нормализатор "склейканеравенств". Он применяется приемами, предпринимающими разбор случаев при решении неравенств, для завершающего упрощения заменяющего утверждения.

(a) Прием нормализатора упрощения дизъюнкции.

Пример:

$\forall_{abc}(a\&b \ \vee \ a\&c \leftrightarrow a\&(b \ \vee c))$

Прием применяется в нормализаторе "склейканеравенств". Утверждение a не содержит неизвестных.

Спецификация приема имеет вид "тип(нормфикс)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 36.

12. Нормализатор выделения заданных подтермов.

В некоторых ситуациях бывает необходимо так преобразовать выражение, чтобы заданные переменные встречались только внутри подтермов заданного вида. Для этой цели создаются специальные нормализаторы, которым входные данные - вид подтермов и список переменных - передаются через специальный комментарий. Пока имеется единственный такой нормализатор - "извлечение". Он используется при формальном интегрировании и в некоторых других случаях. Пример приема:

$\forall_{ab}((\sin a)^{2b} = (1 - (\cos a)^2)^b)$

Прием применяется в нормализаторе "извлечение". Входной комментарий указывает выделяемый подтерм "$\cos a$".

Спецификация приема имеет вид "тип(дискрвеличина)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 72.

13. Нормализатор ограничений на известные параметры.

После того, как задача решена и значения неизвестных найдены, бывает полезно упростить сопровождающие условия на известные параметры. Для этой цели могут использоваться специальные нормализаторы. Пока создано два нормализатора такого типа - "стандменьше" и "стандменьшеилиравно". Они упрощают сопровождающие строгие либо нестрогие неравенства. В простейших случаях эти неравенства разрешаются относительно каких-либо известных параметров, но не обязательно. Пример приема:

$\forall_{abc}(b - \text{rational} \ \& \ \neg(\text{знаменатель}(b) - \text{even}) \ \& \ \neg(\text{числитель}(b) - \text{even}) \ \& \ 0 < b \rightarrow 0 \leq a^b + c \leftrightarrow 0 \leq a + c^{1/b})$

Прием применяется в нормализаторе "стандменьшеилиравно". Выражения b, c константные, выражение a - неконстантное.

Спецификация приема имеет вид "тип(нормквадратура)", "оператор(P)", "направл(N)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 115.

Прием синтезатора

Пример:

$\forall_{abc}(0 < c \ \& \ c \le b \ \& \ 0 < a \to a/b \le a/c)$

Прием применяется в синтезаторе "верхняяоценка". Выражение a константное. Второй антецедент, выделенный указателем "значения", обращается к синтезатору "нижняяоценка" для получения нижней оценки c знаменателя b.

Спецификация приема имеет вид "тип(синтезатор)", "оператор(P)". Справочник "заголовокприема" не прорабатывался.

Число приемов данного типа - 341.

Пакетный анализатор

Приемы пакетных анализаторов делятся на два больших класса. В первом из них прием выполняет вывод следствий, во втором - осуществляет тождественное либо эквивалентное преобразование. Заголовок приема в обоих случаях один и тот же - "внутрвывод(P)", где P - название анализатора. Во втором случае добавляется указатель "внутрпреобр".

1. Прием анализатора, выводящий следствия.

   Пример:

   $\forall_{ABCpq}(\text{актив}(\angle(ACB)) \ \& \ \text{актив}(l(AC)) \ \& \ \text{актив}(l(BC)) \ \& \ p = \sin(\angle(ACB) + \angle(ABC)) \ \& \ q = \sin(\angle(ABC)) \to pl(AC) = ql(BC))$

   Прием применяется в анализаторе "синусы", использующем теорему синусов и ряд теорем, которые могут оказаться полезными в сочетании с ней. Проверяется, что каждое из выражений для угла ACB и расстояний AC, BC либо известно, либо имеет тип "внешнеизв". При этом хотя бы одно из них не известно. Проверяется также, что каждое из выражений p, q, получаемых из правых частей соответствующих антецедентов нормализаторами общей стандартизации, имеет не более одного символа "угол".

   Спецификация приема имеет вид "тип(Неизвестные)", "оператор(P)". Справочник "заголовокприема" не прорабатывался.

   Число учтенных в логическом ассемблере приемов данного типа - 2. В действительности их несколько сотен.

2. Прием анализатора, выполняющий тождественное либо эквивалентное преобразование.

   Пример:

   $\forall_{ABC}(0 \le \pi/2 - \angle(ABC) \ \& \ \neg(a = 0) \to a \sin(\angle(ABC)) = b \leftrightarrow \angle(ABC) = \arcsin(b/a))$

   Прием применяется в анализаторе "синусы". Выражения a, b не содержат неизвестных.

Спецификация приема имеет вид "тип(быстрхарактеристика)", "оператор(P)". Направление замены в данном случае всегда - слева направо. Справочник "заголовокприема" не прорабатывался.

Число учтенных в логическом ассемблере приемов данного типа - 3. В действительности их намного больше.

14.2.11 Прочие приемы

Помимо перечисленных выше типов приемов, имеется множество типов приемов для различных справочников, реализованных на ГЕНОЛОГе. Обычно их синтез хорошо алгоритмизирован и представляет собой чисто техническую задачу. Мы их здесь не приводим.

Кроме того, некоторые приемы сканирования задачи имеют общелогический характер и по сути своей уникальны. Создавать для них процедуры автоматического синтеза не имеет смысла. Такие приемы вынесены в оглавлении типов приемов в пункт "Общелогический прием".

Глава 15

Компилятор спецификаций

Первый шаг по преобразованию спецификации приема в его описание на ГЕНОЛОГе делается справочником "заголовокприема". Он создает бланк блока приема и заносит в него следующие элементы:

1. (заголовок A) - A есть заголовок приема.
2. (условие A) - A есть набор фильтров приема.
3. (прием A) - A есть описание приема без элемента "условие(...)".
4. (примечание A) - A есть спецификация приема.
5. (приемы A) - A есть накопитель четверок (теорема приема - заголовок приема - спецификация приема - описание приема) для созданных по спецификации приемов (иногда их бывает несколько, из-за различий в способе идентификации). Этот элемент создается еще до обращения к справочнику "заголовокприема". Он нужен внешней процедуре, обратившейся к синтезу приемов.

В действительности справочник "заголовокприема" определяет лишь малую часть фильтров и указателей приемов. Он вводит лишь те элементы описания приема, которые привязаны к специфике его типа. В конце почти любой из его программ расположено обращение к процедуре "схемапосылок", которая и продолжает работу по созданию описания приема. Собственно, с этой точки и начинается та часть программы, которую будем называть компилятором спецификаций.

Заметим, что при перечислении типов приемов мы не приводили программ справочника "заголовокприема", а лишь указывали наиболее существенные фильтры и указатели приема. Часть их создавалась справочником до обращения к процедуре "схемапосылок", часть - определялась компилятором спецификаций. Сами программы справочника обычно не сложны и легко вычитываются непосредственно на ЛОСе.

Компилятор спецификаций организован как конвейер: блок приема последовательно обрабатывается процедурами "схемапосылок", "схемаидентификации", "схемапреобразований", "схеманормализации", "фильтрыприема", "учетприема". Каждая из них вносит свой вклад в блок приема, после чего обращается к следующей. Последняя процедура регистрирует результат в элементе (приемы ...).

Заметим, что конвейер работает в режиме перечисления, так что по завершении создания очередной версии приема происходит откат и продолжение работы над следующей версией.

Подробное описание процедур компилятора спецификаций дано в 7 томе монографии "Компьютерное моделирование логических процессов". Оно имеет характер прослеживания ЛОС-программ этих процедур и полезно лишь как справочная информаци при работе с системой. Здесь мы ограничимся каратким описанием действий этих процедур.

1. Процедура "схемапосылок(x1 x2 x3)". x1 - теорема приема, x2 - спецификация приема, x3 - блок приема. Перечисляются возможные наборы указателей обработки антецедентов теоремы. Каждый такой вариант регистрируется в копии блока приема, после чего эта копия передается оператору "схемаидентификации", продолжающему создание приема. При откате возобновляется работа с исходной версией блока приема.

2. Процедура "схемаидентификации(x1 x2 x3)". Этот и следующие операторы конвейера не перечисляющие. x1 - теорема приема, x2 - спецификация приема, x3 - блок приема. Вводятся указатели идентификации, после чего предпринимается обращение к оператору "схемапреобразований", продолжающему создание приема.

3. Процедура "схемапреобразований(x1 x2 x3)". x1 - теорема приема, x2 - спецификация приема, x3 - блок приема. Вводятся указатели, определяющие дополнительные преобразования приема, после чего предпринимается обращение к оператору "схеманормализации", продолжающему создание приема.

4. Процедура "схеманормализации(x1 x2 x3)". x1 - теорема приема, x2 - спецификация приема, x3 - блок приема. Вводятся нормализаторы приема, после чего предпринимается обращение к оператору "фильтрыприема".

5. Процедура "фильтрыприема(x1 x2 x3)". x1 - теорема приема, x2 - спецификация приема, x3 - блок приема. Оператор осуществляет расстановку в блоке приема дополнительных фильтров и указателей, после чего обращается к оператору "учетприема".

6. Процедура "учетприема(x1 x2 x3)". x1 - теорема приема, x2 - спецификация приема, x3 - блок приема. Осуществляется завершающее редактирование описания приема и регистрация его в накопителе (приемы ...) из блока приема.

Таким образом, лишь оператор "схемапосылок" в этой цепочке является перечисляющим, и создание нескольких версий приема означает, что имеется несколько заслуживающих внимания способов обработки антецедентов.

Выйти на программы процедур компилятора спецификаций можно через раздел "Синтез приемов" - "Цикл создания приема по спецификации (Компилятор спецификаций)" оглавления программ.

Глава 16

Тестирование приемов

После того, как возникли теорема приема и его спецификация, компилятор спецификаций предлагает одну либо несколько версий описания этого приема на ГЕНОЛОГе. Каждую из них компилятор ГЕНОЛОГа может преобразовать в ЛОС-программу приема. Таким образом, цикл создания приема, начинающийся в действительности значительно раньше, чем появились теорема приема и его спецификация, как бы завершается. Предшествующие этапы этого цикла, такие, как создание спецификаций по теореме; характеризация теорем, на основе которой создаются спецификации, а также вывод самих теорем, будут рассмотрены в следующих разделах книги.

Однако, хотя после работы компилятора ГЕНОЛОГа программа приема и получена, не следует спешить помещать ее в "основную" базу приемов решателя. Во-первых, прием может оказаться избыточным - решатель и без него будет справляться со всеми задачами, для решения которых он, предположительно, был создан. Более того, "холостой ход" ненужных попыток применения новых приемов может существенно замедлять работу. Во-вторых, даже если есть задачи, для решения которых новый прием действительно необходим, нужно убедиться, что его применение не нарушит решения других задач - сильно замедлит либо вовсе приведет к выдаче отказа. В таких случаях необходима некоторая "доводка" приема - изменение уровня срабатывания, варьирование его типа и т.п.

Таким образом, после получения первой, пока "сырой", версии программы приема, необходимы еще два этапа: создание тестовых задач, доказывающих, что без нового приема решатель с ними не справится, и доводка приема путем "прогонки" решателя по всему задачнику, поиска критических точек - отказов либо сильных замедлений - и устранение их путем уточнения приема.

Хотя некоторые этапы цикла автоматического синтеза приемов будут рассмотрены лишь последующих разделах, дадим его краткое описание сейчас.

Входным данным для цикла служит некоторая теорема, выбранная в базе теорем решателя. Прежде всего, запускается цикл программирующего вывода. Выводятся различные следствия из данной теоремы. Это обеспечивается своего рода решателем, состоящим из множества (сейчас их уже более тысячи, хотя обучение данного решателя лишь начато) приемов вывода теорем. Такие приемы часто ставят вспомогательные задачи и привлекают для их решения весь потенциал решателя. Для получения новых теорем из исходной теоремы могут использоваться дополнительные теоремы, извлекаемые приемами программирующего вывода из произвольных разделов базы теорем. Чтобы ускорить их поиск, применяются многочисленные справочники поиска теорем.

Цикл вывода ориентирован на получение теорем, представляющих интерес для создания приемов. Фактически, он реализует переход от фундаментальных теорем предметной области к "техническим" теоремам, каковыми являются теоремы приемов.

Впрочем, граница между программирующим и исследовательским выводами весьма условна, и в процессе обучения "теоремного"решателя прорабатываются и цепочки "открытия"фундаментальных теорем, например, теоремы Пифагора, формулы Кардано, и т.п. Иногда, по аналогии, система находит новые любопытные утверждения, доказывая свою способность отдавать больше, чем в нее было заложено. "Технических" следствий теоремы она обычно выводит десятки, а иногда даже сотни. Хотя они и не все равноценные, часть таких следствий порождает вполне осмысленные и необходимые для усиления решателя приемы.

В процессе вывода теорем они снабжаются определенными характеристиками, предвосхищающими цели их использования при решении задач. Часть таких характеристик возникает путем непосредственного рассмотрения теоремы, часть - создаются приемами вывода теорем и объясняют, с какой целью теорема была получена.

По завершении цикла вывода теорем запускается процедура, создающая спецификации теорем на основе сопровождающих их характеристик. Она будет рассмотрена в последующих разделах. Обычно по одной теореме создаются несколько (иногда - один-два десятка) приемов. Все это часто приводит к огромному количеству гипотетически полезных приемов, созданных по единственной исходной теореме. Из них необходимо выбрать те, которые в действительности нужны.

Прежде, чем перейти к обработке спецификации компилятором спецификаций и далее - компилятором ГЕНОЛОГа, предпринимается проверка неизбыточности приема. Для этого генерируется очень простой тестовый пример, ориентированный на применение создаваемого приема. Если оказывается, что решатель успешно справляется с ним и без нового приема, то прием не создается. Иначе - он создается и регистрируется в буфере базы приемов. Одновременно в буфере задачника регистрируется задача, доказывающая целесообразность создания приема. Эти действия выполняются справочником "задачи".

Каждый прием справочника "задачи" сначала создает для синтезируемого приема тестовую задачу и убеждается в том, что старых средств для ее решения недостаточно, после чего обращается к процедуре "регприем". Эта процедура завершает создание нового приема, вплоть до получения его ЛОС-программы; проверяет, что тестовая задача с помощью нового приема уже решается, и регистрирует в буферах оглавлений новый прием и новую задачу. Перед тем, как рассматривать программы справочника "задачи", дадим краткое описание процедуры "регприем".

16.1 Процедура "регприем"

Обращение к процедуре имеет вид "регприем(x1 x2 x3 x4)". Здесь x1 - теорема приема, x2 - спецификация приема, x3 - тестовая задача, x4 - набор сопровождающих информационных элементов (опций обращения).

Предпринимается создание приемов по x1 и x2, регистрация их (несколько приемов могут появиться вследствие различных схем обработки антецедентов) в буфере базы

приемов, а также регистрация задачи x3 в буфере задачника. Если источник приемов (т.е. теорема, по которой создавались теорема приема и спецификация) пока не зарегистрирован в базе теорем, то он регистрируется в ее буфере. Создаются перекрестные ссылки между источником и приемами. Предпринимается тестирование созданных приемов на задаче x3. Иногда приемы могут быть слегка модифицированы. Если результаты тестирования негативные и модификация приемов не помогла, то эти приемы удаляются.

Опции обращения к процедуре

Эти опции (элементы набора x4) бывают следующих типов:

1. (ответ A). A есть указатель на способ контроля эффективности созданного приема при решении задачи x3. Такие способы перечисляются ниже.
2. (теорема A). A есть пара (теорема, представляющая собой источник приема, возможно, пока не зарегистрированная в базе теорем - набор ее характеристик). О характеристиках теорем рассказывается в следующем, восьмом томе монографии.
3. "стоп". Прием вводится, но не компилируется.
4. "уровень(A)". При получении отказа предпринимается попытка понизить уровень срабатывания приема до величины A.
5. "усм". При наличии указателя приема "усм(...)" вместо задачи на преобразование решается задача на исследование, в которой упрощаемый терм A представлен фиктивной посылкой "фикс(A)".
6. "урвн(A). Уровень обращения к тестовой задаче повышается до A.

Указатели способа контроля эффективности созданных приемов на тестовой задаче

1. "отказ(A)" - получение ответа, отличного от "отказ", при максимальном уровне обращения к задаче, равном A.
2. "короче(A)" - получение ответа, не более длинного, чем терм A.
3. "макслож(A)" - каждое подвыражение ответа, имеющее максимальную сложность, не длиннее всех максимально сложных подвыражений терма A.
4. "элементарно" - получение ответа, представляющего собой элементарный терм.
5. "легковидеть" - тестирование проверочного оператора, нормализатора либо синтезатора. Детали обращения уточняются фиктивной посылкой "процедура(...)", добавляемой в список посылок тестовой задачи. Фактически вместо решения задачи происходит обращение к соответствующему пакетному оператору.
6. "логсимвол($A_1 \dots A_n$)" - ответ не должен содержать логических символов A_1, $\dots$, A_n.
7. "отображение(A)" - последние термы описателей "отображение" в ответе задачи должны быть короче терма A.

8. "свобоперанд" - получение ответа, не содержащего связанных переменных.

9. "числовойатом" - получение ответа, не содержащего невырожденных числовых атомов.

10. "числатом" - получение ответа, содержащего только невырожденные числовые атомы.

11. "перемОценка(A X)" - сложность подвыражений ответа, содержащих переменные терма X, не более A.

12. "общнорм" - проверяется выполнение требований, накладываемых на преобразование общей стандартизации для тождеств.

13. "константа" - получение константного терма.

14. "существует" - ответ не должен квантора "существует".

15. "класс" - ответ не должен содержать описателя "класс".

16. "содержится($A_1 \dots A_n$)" - ответ должен содержать хотя бы один из символов $A_1, \dots, A_n$.

17. "единствхожд" - ответ не содержит описателя "отображение", имеющего более одного вхождения варьируемой переменной в последнем терме.

18. "заголовок(A)" - ответ должен иметь заголовок A.

19. "первыйсимвол(A)" - ответ должен представлять собой терм, первый операнд которого имеет заголовок A.

20. "внутрпреобр(A)" - отсутствуют вложенные вхождения символа операции A.

21. "связприставка(A)" - отсутствуют описатели со связывающей приставкой длины, не меньшей A.

22. "Оценка(A)" - сложность ответа не превосходит сложности терма A.

23. "станд(A)" - получение ответа B, такого, что результаты применения оператора "станд" к термам A, B различны.

24. "длинаменее(A)" - набор конъюнктивных членов ответа задачи на описание, после отбрасывания сопровождающих по о.д.з. утверждений, должен иметь длину, меньшую чем A (A - символьный номер).

25. "глуб" - ответ задачи имеет единственный конъюнктивный член длины более 2, содержащий неизвестную. Глубина вхождения в него этой неизвестной равна 1. Допускается случай, когда анализируется только конъюнктивный член вида "неопр(t)", причем оказывается, что глубина вхождения неизвестной в t равна 1.

26. "варьир(A)" - ответ не содержит описателей "отображение", у которых глубина вхождения варьируемой переменной в предпоследний терм не менее A.

27. "связка(A)" - ответ не должен содержать описателей "класс", у которых число вхождений варьируемой переменной не менее A

28. "независит($x_1 \dots x_n$)" - ответ не должен содержать переменных $x_1, \dots, x_n$.

29. "вароценка(A)" - вычисленная без учета символа "вариант" сложность ответа не превосходит такой же сложности терма A.

30. "сокращ(A)" - наиболее сложное подвыражение ответа не должно иметь сложность, большую сложности терма A, а в случае равенства сложностей - должно быть короче A.

31. "декомпозиция(A)" - либо сложность ответа меньше сложности терма A, либо эти сложности равны, а наиболее сложные подтермы ответа образуют по своим переменным декомпозицию множества переменных терма A.

32. "разделены(A_1, A_2)" - переменные списка A_1 в ответе не встречаются в одном операнде отношения с переменными списка A_2.

33. "единствсущ" - ответ имеет единственный подтерм максимальной сложности.

34. "числатомы" - в ответе не должно содержаться более одного вхождения невырожденного числового атома.

35. "числзнач" - в ответе не должно содержаться повторных вхождений невырожденных числовых атомов.

36. "крд" - в ответе нет невырожденных числовых атомов, отличных от координатных.

37. "нормкрд" - в ответе нет координатных числовых атомов.

38. "квантглубина(A)" - кванторная глубина ответа должна быть меньше, чем кванторная глубина терма A.

39. "разбиение(A)" - список элементарных утверждений ответа представляет собой декомпозицию терма A по его параметрам.

40. "упрощописатель" - результирующий терм упрощает описатели исходного терма.

41. "уменьшсложн" - сложности исходного и результирующего термов равны, но результирующий терм имеет единственный самый сложный подтерм, а исходный - несколько, и они образуют декомпозицию первого по своим переменным.

42. "глубина(x)" - глубина переменной x в ответе должна равняться 1. Берутся конъюнктивные члены ответа. Предварительно, если он есть, отбрасывается квантор существования.

43. "переменные(A)" - ответ должен содержать только такие переменные, которые входят в терм A.

44. "нормкоорд(A)" - переход к координатам более простых объектов, чем объекты в координатном выражении A.

45. "меньше(A)" - сложность результирующего терма меньше сложности терма A.

46. "числнеизв(A)" - число содержащих неизвестную конъюнктивных членов ответа не меньше A (A - символьный номер).

47. "известны" - сложность максимальных известных подвыражений ответа меньше сложности таких выражений исходного условия.

48. "принадлежит(A)" - для каждой переменной x терма A ответ должен содержать подтерм вида "принадлежит(r фикс(x))".

49. "коорд" - ответ должен содержать координаты либо разряды координат.

Описание программы процедуры "регприем" можно найти в 7 томе монографии "Компьютерное моделирование логических процессов".

16.2 Справочник "задачи"

Справочник "задачи" получает следующие входные данные: x1 - теорема приема, x2 - спецификация приема, x3 - пара (теорема, являющаяся источником приема, возможно, пока не зарегистрированная в базе теорем - набор ее характеристик). Текущий логический символ - тип приема. Предпринимается создание одной либо нескольких задач для тестирования полезности приема, попытка решения этих задач старыми средствами и сравнение с результатом применения нового приема. Если создание приема целесообразно, то он сохраняется в буфере базы приемов. Одновременно в буфере задачника регистрируется задача, доказавшая целесообразность ввода прием. Если прием был создан, то результатом обращения к справочнику служит символ 1, иначе - 0.

Заметим, что на момент обращения к справочнику описания приема на ГЕНОЛОГе и тем более его ЛОС-программы пока нет. Прием создается лишь после того, как была сгенерирована тестовая задача и выяснилось, что старых средств для ее решения недостаточно.

Переходя к описанию программ справочника "задачи", заметим, что они зарегистрированы в ветви оглавления программ "Синтез приемов" - "Синтез приемов с генерацией задач для тестирования их неизбыточности" - "Приемы справочника ЗАДАЧИ (создание тестовой задачи, проверка полезности приема и завершающая регистрация приема, задачи и теоремы)". Эта ветвь воспроизводит ветвь оглавления программ, в которой перечисляются типы приемов. Обе ветви имеют одинаковые названия соответствующих пунктов. Однако, на текущий момент далеко не все типы приемов охватываются справочником "задачи", и его оглавление существенно беднее. В процессе обучения данный справочник будет пополняться новыми программами.

Тестовые примеры обычно имеют одношаговый характер - они должны решаться непосредственно применением нового приема. Пока они чрезвычайно просты. Во многих ситуациях это приводит к отбрасыванию полезных приемов, так как их полезность может быть выявлена лишь на более изощренных задачах, а простые примеры легко решаются старыми средствами. Работа в этом направлении лишь начата. Во всяком случае, тестирование приемов уже сейчас оказывается мощным фильтром, способным сводить поток из сотен потенциально возможных приемов к нескольким десяткам предположительно полезных.

Полное описание программ справочника "задачи" можно найти в 7 томе монографии "Компьютерное моделирование логических процессов". Здесь мы приведем лишь сокращенное описание.

16.3 Тождественная замена

Общая стандартизация

1. Общая стандартизация. Тип приема - "общнорм".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы приема, а условием - терм "фикс(t)", где t - заменяемая часть теоремы приема. Цели задачи - "упростить", "задачи". Находится ответ t_1 копии этой задачи, решаемой при максимальном уровне 8. Рассматривается вспомогательная задача на преобразование Z', посылками которой служат антецеденты теоремы, а условием - терм "фикс(r)", где r - заменяющая часть теоремы. Цели ее - "упростить", "задачи". Находится ответ r_1 задачи Z', решаемой при максимальном уровне 8. Проверяется, что выражения t_1, r_1 различны. Проверяется также, что либо r_1 короче t_1, либо оценка сложности r_1 меньше оценки сложности t_1, либо эти оценки равны, причем для каждого имеющего максимальную сложность подтерма терма t_1 имеется более короткий подтерм такой же сложности в терме r_1. Проверяется, что если t_1 бесповторно, а r_1 небесповторно, то терм t_1 имеет параметр, не входящий в r_1. Тогда предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опций "усм", "(ответ общнорм)".

2. Общая стандартизация выражения, использующая явно идентифицированный антецедент для подмножества операндов ассоциативно - коммутативной операции. Тип приема - "уникопия".

   В спецификации приема находится элемент "переменная(x)". Рассматривается заменяемая часть t теоремы приема. В ней выделяется операция f, одним из операндов которой служит переменная x. Выбираются переменные y, z, не входящие в теоремы приема, и создается терм r вида $f(y, z)$. Определяется результат t_1 обработки нормализатором "норм" результата подстановки r вместо x в t. Определяется также список S результатов этой подстановки в антецеденты теоремы приема. Создается задача на преобразование Z с посылками S и условием t_1. Цели ее - "одз", "упростить". Находится ответ t_2 копии данной задачи, решаемой до максимального уровня 6. Определяется результат p_1 обработки нормализатором "норм" результата подстановки r вместо x в заменяющую часть теоремы приема. Проверяется, что p_1 короче, чем t_2. Тогда предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ короче(p_1))".

Преобразование описателей

1. Исключение описателя "класс".

   (a) Исключение описателя "класс". Тип приема - "цепьвоглавлении".

Создается задача на преобразование Z, посылки которой суть антецеденты теоремы приема, а условие - заменяемый терм теоремы. Цели задачи - "упростить", "одз". Копия этой задачи решается до максимального уровня 7. Проверяется, что ответ имеет связанные переменные, а заменяющий терм теоремы приема - не имеет. Тогда предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ свобоперанд)".

2. Стандартизация описателя "класс".

   (a) Переход от параметрического задания класса к непосредственному. Тип приема - "константа".

   Проверяется, что в заменяемый терм теоремы приема входит квантор существования. Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - заменяемый терм. Цели задачи - "упростить", "одз". Копия этой задачи решается до максимального уровня 5. Проверяется, что ответ содержит квантор существования, а заменяющий терм теоремы - не содержит. Тогда предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ существует)".

3. Упрощение выражения под описателем относительно варьируемой переменной. Тип приема - "измзнака".

   В спецификации приема находится элемент "переменная(x)". Антецеденты теоремы разбиваются на список S_1 утверждений, содержащих x, и список S_2 остальных утверждений. Выбирается не входящая в теорему переменная y. Проверяется, что переменная x имеет больше одного вхождения в заменяемый терм t и единственное вхождение в заменяющий. Формируется выражение r вида "отображение(x $A(x)$ t)", где $A(x)$ - конъюнкция утверждений списка S_1 и терма "значение(y x)". Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - выражение r. Цели задачи - "упростить", "одз". Копия этой задачи решается до максимального уровня 5. Проверяется, что ответ имеет вид "отображение(x $B(x)$ $p(x)$)", причем переменная x входит в $p(x)$ неоднократно. Тогда предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ единствхожд)".

4. Определение характеристики отображения.

   (a) Непосредственное определение характеристики отображения. Тип приема - "усмпростое".

   Создается задача на преобразование Z, посылки которой суть антецеденты теоремы приема, а условие - заменяемый терм теоремы. Цели задачи - "упростить", "одз". Копия этой задачи решается до максимального уровня 7. Проверяется, что ответ содержит символ "отображение", а заменяющий терм теоремы - не содержит. Тогда предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(отображение))".

   (b) Операции над семействами.

i. Непосредственное вычисление операции над семейством.

A. Непосредственное вычисление операции над семейством. Тип приема - "эллипсоид".

Аналогично предыдущему. Создается задача на преобразование Z, посылки которой суть антецеденты теоремы приема, а условие - заменяемый терм теоремы. Цели задачи - "упростить", "одз". Копия этой задачи решается до максимального уровня 7. Проверяется, что ответ содержит символ "отображение", а заменяющий терм теоремы - не содержит. Тогда предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(отображение))".

B. Шаг развертки операции над конечным семейством в обычную операцию. Тип приема - "раскрытьскобки".

Рассматривается заменяемый терм T теоремы. Проверяется, что он имеет вид "$f(\lambda_x(t(x), A(x)))$", где x - единственная переменная. Если $A(x)$ содержит символ "префикс", то определяется результат T' замены в T подтерма $A(x)$ на "$x \in \{1, 2, 3, 4\}$". В противном случае $A(x)$ заменяется на "$x = 1 \vee x = 2 \vee x = 3 \vee x = 4$". Создается задача на преобразование Z, посылки которой суть антецеденты теоремы приема, а условие - терм T'. Цели задачи - "упростить", "одз". Копия этой задачи решается до максимального уровня 5. Проверяется, что ответ содержит символ "отображение". Тогда предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(отображение))".

ii. Занесение внешнего члена под знак операции над семейством.

A. Занесение внешнего члена под знак операции над семейством. Тип приема - "сдвигзапятой".

Проверяется, что заменяющий терм имеет вид $f(\lambda_x(t(x), A(x)))$, где x - единственная переменная. Выбирается новая переменная y и рассматривается заголовок g заменяемой части (согласно типу приема, он является двуместной ассоциативно-коммутативной операцией, обобщением которой на произвольные конечные списки служит символ f). Затем создается задача на преобразование Z, условием которой является выражение $g(t(2), f(\lambda_x(t(x), y(x))))$. Единственная посылка - константа "истина"; цели - "упростить", "одз". Копия задачи Z решается до максимального уровня 5. Если ответ не имеет вида операции над описателем "отображение", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ первыйсимвол(отображение))".

B. Занесение внешнего члена под знак операции над семейством - случай продолжения ряда значений. Тип приема - "простойцикл".

Проверяется, что заменяющий терм имеет вид $f(\lambda_i(t(i), A(i)))$, где i - единственная переменная. Формируется терм T вида $f(\lambda_i(t(i), i - \text{целое } \& \ 2 \leq i \ \& \ i \leq 20))$. Находится заголовок g заменяемого терма. Согласно типу приема, он является двуместной ассоциативно-коммутативной операцией, обобщением которой на произвольные

конечные списки служит символ f. В качестве выражения Q поочередно рассматриваются выражение $t(1)$ и $t(21)$. Для каждого из них создается задача на преобразование Z, условием которой служит выражение $g(Q,T)$ а единственной посылкой - константа "истина". Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 5. Если ответ содердит символ "или" либо не является операцией над описателем "отображение", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опций "(ответ первыйсимвол(отображение))", "(ответ логсимвол(или))". Если первая попытка удачная, вторая не выполняется.

iii. Переход к операциям над семействами, имеющими более простой вид общего члена.

A. Сведение операции над семейством к операциям над семействами, имеющими более простой вид общего члена. Тип приема - "фильтр-радикалов".

Рассматривается список S антецедентов теоремы. Если спецификация приема имеет элемент "указатель(занесениепосылки(i A))", то рассматривается i - й антецедент B. Определяется список X общих переменных термов A, B, и антецедент B заменяется в списке S на кванторную импликацию "$\forall_X(A \to B)$". После коррекции списка S проверяется, что заменяемый терм имеет вид "$f(\lambda_x(t(x), Q(x)), \ldots)$". Создается задача на преобразование Z, посылками которой служат утверждения списка S, а условием - заменяемый терм теоремы. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 5. Если ответ содержит описатель "$\lambda_y(r(y), P(y))$", у которого выражение $r(y)$ не короче выражения $t(x)$, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отображение($t(x)$))".

5. Определение характеристики класса.

(a) Непосредственное определение характеристики класса. Тип приема - "орграф".

Создается задача на преобразование Z, посылки которой суть антецеденты теоремы, а условие - заменяемый терм. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 5. Если ответ содержит символ "класс", а заменяющий терм теоремы - не содержит, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(класс))".

Сближение подвыражений задачи

1. Задачи на преобразование.

(a) Переход в задаче на преобразование к уже имевшимся подтермам. Тип приема - "поглощается".

Рассматриваются заменяемое выражение t_1 и заменяющее t_2. Определяется тип s значения выражений с заголовком выражения t_1. Находится раздел, к которому относится s, и просматриваются символы f двуместных операций этого раздела. Справочник поиска теорем "констнорм" определяет по f теорему T, представляющую собой равенство выражения вида $f(A_1, A_2)$ некоторому константному выражению. Левая часть этого равенства содержит единственную переменную x, причем эта переменная входит в нее ровно дважды. Определяется результат R замены в выражении $f(A_1, A_2)$ первого вхождения переменной x на t_1, а второго - на t_2. Определяется список S, полученный добавлением к антецедентам теоремы приема результатов подстановки в антецеденты теоремы T выражения t_1 вместо переменной x. Создается задача на преобразование Z, посылками которой служат утверждения S, а условием - выражение R. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 7. Если ответ не является константным термом, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ константа)". Если же ответ на копию задачи Z был константным, то сразу выход из справочника по значению 0.

Выражения с неизвестными

1. Группировка относительно выражения с неизвестными. Тип приема - "нормнеизв".

   В спецификации находится элемент "неизвестные(x)". Создается список S - объединение списка антецедентов с утверждениями, сопровождающими по о.д.з. для этих же антецедентов и заменяемого выражения теоремы приема. Список S разбивается на подсписок S_1 утверждений, содержащих x, и подсписок S_2 утверждений, не содержащих x. Выбирается переменная a, не встречающаяся в теореме, и к списку S_1 добавляется равенство заменяемого выражения этой переменной. Затем создается задача на описание Z, посылками которой служат утверждения S_2, а условиями - утверждения S_1. Цели задачи - "пример", "прямойответ", "одз", "неизвестые x". Копия задачи Z решается до максимального уровня 7, причем используется средний ограничитель трудоемкости. При получении отказа предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(7))".

2. Группировка в одном подвыражении всех неизвестных условия задачи на описание либо посылки задачи на исследование. Тип приема - "путьвоглавлении".

   В спецификации находится элемент "терм(t)". Проверяется, что терм t имеет единственную переменную x. Список антецедентов разбивается на подсписок S_1 всех содержащих переменную x утверждений и подсписок S_2 утверждений, не содержащих x. Выбирается переменная a, не встречающаяся в теореме, и к списку S_1 добавляется равенство заменяемого выражения этой переменной. Затем создается задача на описание Z, посылками которой служат утверждения S_2, а условиями - утверждения S_1. Цели задачи - "полный", "явное", "прямойответ", "одз", "неизвестые x". Копия задачи Z решается до максимального уровня 7. При получении отказа предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(7))".

3. Уменьшение глубины неизвестной. Тип приема - "точкаокружности".

   В спецификации находится элемент "неизвестные(x)". Создается список S - объединение списка антецедентов с утверждениями, сопровождающими по о.д.з. для этих же антецедентов и заменяемого выражения теоремы приема. Список S разбивается на подсписок S_1 утверждений, содержащих x, и подсписок S_2 утверждений, не содержащих x. К списку S_1 добавляется терм "неопр(t)", где t - заменяемое выражение. Затем создается задача на описание Z, посылками которой служат утверждения S_2, а условиями - утверждения S_1. Цели задачи - "попыткаспуска", "прямойответ", "одз", "неизвестые x". Копия задачи Z решается до максимального уровня 7. Среди конъюнктивных членов ответа выбирается терм "неопр(r)". Если глубина вхождений переменной x в r больше единицы, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ глуб)".

4. Сведение неизвестного подвыражения условия задачи на описание либо посылки задачи на исследование к более простым неизвестным выражениям.

   (a) Сведение неизвестного подвыражения условия задачи на описание либо посылки задачи на исследование к более простым неизвестным выражениям. Тип приема - "записьприема".

   Проверяется, что оценки сложности заменяемого и заменяющего выражений равны, причем самый сложный подтерм заменяемого терма T - он сам, а заменяющий терм имеет единственный самый сложный подтерм, более короткий, чем терм T. Создается задача на описание Z, имеющая своей единственной посылкой константу "истина", а списком условий - набор антецедентов теоремы, пополненный термом "неопр(T)". Цели задачи - "попыткаспуска", "прямойответ", "одз", "неизвестные X", где X - список параметров терма T. Копия задачи Z решается до максимального уровня 7. Если оценка сложности ответа больше, чем оценка сложности терма T, либо эти оценки равны, причем либо ответ имеет более одного подтерма максимальной сложности, либо его единственный подтерм максимальной сложности не короче терма T, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ сокращ(T))".

   (b) Декомпозиция неизвестного подвыражения в условии задачи на описание. Тип приема - "схемавариантов".

   Проверяется, что оценки сложности заменяемого терма P и заменяющего терма Q равны. Составляется список $x_1, \ldots, x_k$ всех параметров заменяемого терма. Проверяется, что терм P имеет единственный самый сложный подтерм, причем этот подтерм совпадает с самим термом P. Проверяется, что терм Q имеет более одного подтерма максимальной сложности, причем ни один из этих подтермов не содержит всех переменных $x_1, \ldots, x_k$. Создается задача на описание Z, имеющая своей единственной посылкой константу "истина". Условиями служат все антецеденты теоремы и терм "неопр(набор($P, x_1, \ldots, x_k$))". Цели задачи - "попыткаспуска", "прямойответ", "одз", "неизвестные $x_1 \ldots x_k$". Копия задачи Z решается до максимального уровня 7. Если оценка сложности ответа больше оценки сложности терма P, либо эти оценки равны, но набор имеющих наибольшую

сложность подтермов ответа не образует декомпозицию по переменным терма P, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ декомпозиция(P))".

Константные подвыражения

1. Стандартизация с помощью вычислений. Тип приема - "исключприем".

   Составляется список $x_1, \ldots, x_k$ заголовков левых частей равенств в антецедентах теоремы. Проверяется, что эти заголовки суть различные переменные. Составляет также список $t_1, \ldots, t_k$ правых частей этих равенств. Путь P - остаток антецедентов. Выбираются переменные y, z, не входящие в заменяемый терм T, и составляется список S_1 термов "терм(y)", "позиция(z y)", "вид(z T)". Если спецификация приема содержит терм "см(A_1 ... A_n)", то термы $A_1, \ldots, A_n$ добавляются к списку S. Создается вспомогательная задача S на преобразование группы фильтров ГЕНОЛОГА - тройка (S_0, S_1, S_2), где S_0 - набор термов вида "истинно(F)"для всевозможных элементов F списка P, S_2 - одноэлементный набор (пример y).

   Целевой элемент (пример y) означает, что в задаче требуется найти пример значения неизвестной y (терма, набора термов, задачи и т.п.). Этот ответ будет зарегистрирован в целевом элементе (ответ ...), создаваемом оператором "преобрфильтр". Заметим, что пока эти целевые элементы введены лишь ради данного пункта генератора тестовых задача. Обслуживающие их приемы в приведенном выше описание программы оператора "преобрфильтр" были опущены.

   Далее начинается цикл из одного либо двух обращений к оператору "преобрфильтр". Каждый раз обрабатывается копия S' задачи S. Рассматривается элемент (ответ R), возникающий в задаче S'. Внутри терма R находится подтерм Q с тем же заголовком, что у заменяемого терма T теоремы приема. Определяется подстановка U вместо параметров терма T, переводящая его в терм Q. Находятся результаты $t'_1, \ldots, t'_k$ применения U к термам $t_1, \ldots, t_k$, упрощенные вспомогательными задачами на преобразование. Определяются результат B подстановки термов $t'_1, \ldots, t'_k$ вместо переменных $x_1, \ldots, x_k$ в заменяющий терм теоремы, а также результат R' замены подтерма Q терма R на терм B. Находится результат R'' упрощения терма R' вспомогательной задачей на преобразование относительно утверждений списка P, к которым применена подстановка U.

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы приема, а условием - терм R. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 4. Если терм R'' короче полученного ответа, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ короче(R''))".

   В противном случае предпринимается вторая (и последняя) попытка обращения к оператору "преобрфильтр". Перед этим отслеживается прием D тождественной либо эквивалентной замены, сработавший во время решения копии задачи Z. Рассматриваются варианты идентификации заменяемой части его

теоремы с заменяемой частью T теоремы синтезируемого приема. Для каждого из них рассматривается фильтр приема D, содержащий переменные заменяемой части его теоремы, и во второй элемент тройки S' заносится отрицание такого фильтра, приведенное к переменным теоремы синтезируемого приема. Таким образом создается указание на поиск примера R, для которого прием D не срабатывал бы.

Числовые атомы

1. Выражение числового атома через численные параметры.

   (a) Выражение числового атома через численные параметры с помощью равенств в посылках.

      i. Использование равенства из контекста, явно выражающего числовой атом через численные параметры. Тип приема - "величинаугла".

      Создается задача на преобразование Z, посылками которой служат антецеденты теоремы и утверждение "число(t)", где t - заменяющий терм теоремы. Условием задачи служит заменяемый терм. Цели задачи - "упростить". Копия задачи Z решается до максимального уровня 6. Если ответом служит терм, отличный от переменной, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ короче(t))".

2. Приведение подобных членов с числовыми атомами.

   (a) Приведение подобных членов с невырожденными числовыми атомами. Тип приема - "кривая".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, пополненные условиями на о.д.з. для заменяемого терма. Условием задачи служит заменяемый терм. Цель задачи - "фикс". Копия задачи Z решается до максимального уровня 6. Если ответ имеет не менее двух вхождений невырожденных числовых атомов, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ числатомы)".

   (b) Приведение подобных членов с числовыми атомами при ограничениях на типы атомов коэффициентов. Тип приема - "меньшеилиравно".

   Действия такие же, как в предыдущем пункте.

   (c) Догруппировка относительно числового атома. Тип приема - "нормкосинус".

   Рассматривается заменяемый терм t теоремы приема. В нем имеется единственный невырожденный числовой атом A. Каждая не входящая в A переменная терма t заменяется в нем на копию выражения A, в которой все переменные переобозначены на новые переменные. Затем создается задача на преобразование Z, посылками которой служат антецеденты теоремы и условия на о.д.з. для терма t. Условием задачи является терм t (преобразованный, как указано выше). Цели задачи - "нормтеорема", "фикс". Копия задачи Z решается до максимального уровня 7. Если ответ имеет более

одного вхождения одного и того же невырожденного числового атома, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ числзнач)".

3. Использование пропорциональной линейной комбинации числовых атомов, усматриваемой в посылках.

   (a) Выражение линейной комбинации числовых атомов через численные параметры при помощи посылки, представляющей собой равенство для пропорциональной линейной комбинации тех же атомов. Тип приема - "контрольнормализации".

   Составляется список S антецедентов теоремы, пополненный условиями на о.д.з. этих антецедентов и консеквента. Если в списке S встречается равенство $x = t$, где x - переменная, не входящая в t, а терм t не содержит символа "плюс", то оно исключается, а в остальных утверждениях списка S вместо x подставляется t. Создается задача на преобразование Z, посылками которой служат утверждения S, а условием - заменяемый терм теоремы. Единственная цель - "неопр". Копия задачи Z решается до максимального уровня 6. Если ответ содержит невырожденный числовой атом, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ числовойатом)".

   (b) Выражение дроби с линейной комбинацией числовых атомов через численные параметры при помощи посылки, представляющей собой равенство для пропорциональной линейной комбинации тех же атомов. Тип приема - "возрастание".

   Составляется список S антецедентов теоремы, пополненный условиями на о.д.з. этих антецедентов и консеквента. Если в списке S встречается равенство $x = t$, где x - переменная, не входящая в t, а терм t не содержит символа "плюс", то оно исключается, а в остальных утверждениях списка S вместо x подставляется t. В списке S находится равенство с нулем в правой части. На самом деле оно имеет для данного типа приемов вид "$pb-aq=0$". Во всех равенствах списка S переменные p, b, a, q заменяются на выражения $2p$, $3a$, $2a$, $3p$ соответственно. Такая же замена выполняется в заменяемом терме T. Все преобразованные термы обрабатываются нормализаторами общей стандартизации.

   Создается задача на преобразование Z, посылками которой служат утверждения S, а условием - преобразованный указанным образом терм T. Цели задачи - "известно", "неопр". Копия задачи Z решается до максимального уровня 7. Если ответ содержит невырожденный числовой атом, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ числовойатом)".

Исключение сложных операций и вычисления

1. Исключение сложной операции.

(a) Непосредственное исключение сложной операции. Тип приема - "стандупорядочение".

Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 7. Если оценка сложности ответа больше оценки сложности заменяемого терма t, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ Оценка(t))".

(b) Исключение сложной операции с помощью кванторного тождества из контекста.

i. Непосредственное исключение сложной операции с помощью кванторного тождества, имеющегося в контексте. Тип приема - "дискретнаяматематика".

Составляется список S антецедентов теоремы. Если в этом списке встречается равенство $x = t$, где x - переменная, не входящая в t, а терм t не содержит символа "плюс", то оно исключается из списка, а в заменяемый терм T вместо x подставляется t. Создается задача на преобразование Z, посылками которой служат утверждения S, а условием - преобразованный указанным образом терм T. Цели задачи - "одз", "прием", "упростить". Копия задачи Z решается до максимального уровня 7. Если ответ содержит заголовок s самого сложного подтерма терма T, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(s))".

2. Переход к сложной операции, имеющей более простеы операнды.

(a) Переход к сложной операции, имеющей более простеы операнды. Тип приема - "описатель".

Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 8. Если оценки сложности ответа и заменяющего терма t равны, причем ответ содержит подтерм максимальной сложности, более длинный, чем любой подтерм такой же сложности в t, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ макслож(t))".

(b) Декомпозиция сложной операции. Тип приема - "сопровождтерм".

Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм. Цели задачи - "одз", "задачи", "упростить". Копия задачи Z решается до максимального уровня 8. Если оценки сложности ответа и заменяющего терма t равны, причем ответ содержит подтерм максимальной сложности, более длинный, чем любой подтерм такой же сложности в t, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ макслож(t))".

(c) Декомпозиция сложной операции для получения повторяющихся вхождений неконстантного терма. Тип приема - "контрольслучаев".

Проверяется, что оценка сложности заменяемого терма t равна оценке сложности заменяющего терма r. Выбирается подтерм s терма r, имеющий максимальную сложность и не входящий в терм t. Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - терм "набор(t, s)". Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 8. Если оценки сложности ответа и заменяющего терма r равны, причем ответ содержит подтерм максимальной сложности, более длинный, чем любой подтерм такой же сложности в r, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ макслож(r))".

(d) Декомпозиция сложной операции, использующая посылку для идентификации подмножества операндов ассоциативно-коммутативной операции. Тип приема - "перечни".

В спецификации приема находится элемент "антецедент(i)", указывающий номер i антецедента, позволяющего идентифицировать подмножество операндов ассоциативно-коммутативной операции, рассматриваемой в заменяемом терме t. Находится данный антецедент и определяется его единственная переменная x. Выбираются новые переменные y, z. В терме t рассматривается ассоциативно - коммутативная операция $f(x, v)$, где v - переменная. Определяется результат t' замены в t этой операции на $f(x, v, y, z)$. Находится список S результатов подстановки в антецеденты теоремы термов $f(x, y)$ и $f(v, z)$ вместо переменных x, v. Этот список пополняется утверждениями, необходимыми для сопровождения по о.д.з. его элементов. Находится результат r подстановки в заменяющий терм теоремы термов $f(x, y)$ и $f(v, z)$ вместо переменных x, v.

Создается задача на преобразование Z, посылками которой служат утверждения списка S, а условием - терм t'. Цели задачи - "дистрибразвертка", "упростить". Копия задачи Z решается до максимального уровня 8. Если оценки сложности ответа и терма r равны, причем ответ содержит подтерм максимальной сложности, более длинный, чем любой подтерм такой же сложности в r, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ макслож(r))".

Координаты

1. Выражение координат объекта через невырожденные числовые атомы. Тип приема - "добавлениеветви".

Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 7. Если ответ не имеет заголовка "набор", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(набор))".

2. Выражение координат объекта через указанные в посылках координаты другого объекта. Тип приема - "Набор".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 7. Если ответ содержит невырожденный числовой атом, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ числовойатом)".

3. Выражение числового атома через координаты.

   (a) Выражение числового атома посылки через координаты. Тип приема - "типы".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, пополненные утверждениями, необходимыми для сопровождения по о.д.з. этих антецедентов и заменяемого терма. Условием служит заменяемый терм. Цели задачи - "одз", "неопр", "упростить". Копия задачи Z решается до максимального уровня 7. Если ответ содержит невырожденный числовой атом, не являющийся названием отдельной координаты (например, "крд"), то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ крд)".

   (b) Выражение числового атома посылки через параметры уравнения для координат множества объектов. Тип приема - "знач".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, пополненные утверждениями, необходимыми для сопровождения по о.д.з. этих антецедентов и заменяемого терма. Условием служит заменяемый терм. Цели задачи - "неопр", "упростить". Копия задачи Z решается до максимального уровня 7. Если ответ содержит невырожденный числовой атом, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ числовойатом)".

16.4 Эквивалентная замена

Общая стандартизация одного утверждения

1. Безусловная общая стандартизация одного утверждения. Тип приема - "нормэкв".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм A. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 5. Если результаты обработки оператором "станд" ответа и терма A совпадают, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ станд(A))".

2. Условная общая стандартизация одного утверждения. Тип приема - "числооперандов".

Создается задача на описание Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемое утверждение A. Цели задачи - "полный", "явное", "прямойответ", "одз", "редакция". Копия задачи Z решается до максимального уровня 7, причем вводится средний ограничитель трудоемкости. Если результаты обработки оператором "станд" ответа и терма A совпадают, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ станд(A))".

3. Элементарная переформулировка с исключением сложного понятия. Тип приема - "обрыв".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм A. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 5. Если оценка сложности ответа не меньше оценки сложности терма A, то рассматривается заменяющий терм B и предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ Оценка(B))".

4. Дизъюнктивно-конъюнктивная декомпозиция элементарного утверждения.

   (a) Конъюнктивная декомпозиция элементарного утверждения.

      i. Конъюнктивная декомпозиция элементарного утверждения. Тип приема - "огрсверху".

         Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм A. Цели задачи - "упростить". Копия задачи Z решается до максимального уровня 7. Если результаты обработки оператором "станд" ответа и терма A совпадают, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ станд(A))".

      ii. Конъюнктивная декомпозиция элементарной посылки. Тип приема - "трансlзамена".

         Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм A. Цели задачи - "смпосылка", "одз", "противоречие". Задаче передается комментарий "(цель равно)".

         Цель "смпосылка" означает, что решение задачи на преобразование сразу же сводится к решению задачи на исследование Z', посылками которой служат все посылки задачи на преобразование и конъюнктивные члены условия задачи на преобразование. Целями задачи на исследование служат все цели задачи на преобразование, отличные от цели "смпосылка". Комментарий (цель равно) означает, что после решение задачи Z' проверяется наличие ее посылки, совпадающей, с точностью до обработки оператором "станд", с исходным условием задачи на преобразование. Если таковая имеется, то выдается ответ 0, иначе - ответ 1.

         Копия задачи Z решается до максимального уровня 7. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(1))".

(b) Дизъюнктивная декомпозиция элементарного утверждения.

i. Дизъюнктивная декомпозиция элементарного утверждения. Тип приема - "сверткавариант".

Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм A. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 7. Если результаты обработки оператором "станд" ответа и терма A совпадают, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ станд(A))".

(c) Дизъюнктивно - конъюнктивная декомпозиция элементарного утверждения.

i. Дизъюнктивно - конъюнктивная декомпозиция элементарного утверждения. Тип приема - "множество".

Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм A. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 7. Если набор атомарных утверждений, из которых ответ образован с помощью логических связок, не является декомпозицией терма A по его переменным, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ разбиение(A))".

5. Константные выражения.

(a) Упрощение утверждения относительно неконстантных выражений. Тип приема - "удалениезамечания".

В спецификации находится элемент "переменные($x_1, \ldots, x_k$)". Пусть S - список антецедентов теоремы, A - заменяемый терм. Рассматривается список $y_1, \ldots, y_m$ всех параметров терма A, отличных от переменных $x_1, \ldots, x_k$. Эти параметры заменяются в утверждениях S и в терме A на константные термы "фикс(1)", ..., "фикс(m)". Затем создается задача на преобразование Z, посылками которой служит измененные утверждения S, а условием - измененный терм A. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 5. Если наибольшая из оценок сложности содержащих переменные $x_1, \ldots, x_k$ подвыражений ответа не меньше аналогичной величины для измененного терма A, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ перемОценка(M переменные($x_1, \ldots, x_k$)))". Здесь M - наибольшая из оценок сложности содержащих переменные $x_1, \ldots, x_k$ подвыражений заменяющей части теоремы.

6. Общая стандартизация с исключением квантора.

(a) Общая стандартизация с исключением квантора. Тип приема - "пример".

Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 7. Если ответ имеет связанные переменные, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ свобоперанд)".

7. Общая стандартизация с исключением описателя. Тип приема - "списокзадач".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 5. Если ответ имеет связанные переменные, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ свобоперанд)".

8. Свертка дизъюнкции. Тип приема - "полный".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 7. Если в ответ входит символ "или", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(или))".

9. Свертка конъюнкции. Тип приема - "блоктеорем".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - терм "фикс(A)", где A - заменяемый терм. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 7. Если в ответ входит символ "и", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(и))".

Общая стандартизация группы утверждений

1. Общая стандартизация группы посылок. Тип приема - "заменатермов".

   Заменяющий терм упрощается при помощи вспомогательной задачи на преобразование, посылками которой служат антецеденты теоремы. Пусть A - результат упрощения. Создается задача на преобразование Z, посылки которой - антецеденты теоремы и конъюнктивные члены заменяемого терма. Условием задачи служит терм A. Цели задачи - "Посылка", "упростить". Копия задачи Z решается до максимального уровня 6. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(1))".

Сокращенная переформулировка

1. Дизъюнктивно-конъюнктивная свертка в условии задачи на свертку. Тип приема - "сборка".

   Создается задача на описание Z, посылками которой служат антецеденты теоремы, а единственным условием - терм "фикс(A)", где A - заменяемый терм теоремы. Цели задачи - "полный", "явное", "прямойответ", "редакция", "соединение". Копия задачи Z решается до максимального уровня 5. Ее ответ будет иметь вид "фикс(R)". Если заменяющий терм B теоремы приема оказался короче терма R, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ короче(фикс(B)))".

2. Сокращенная переформулировка группы условий задачи на свертку. Тип приема - "соединение".

   Проверяется, что заменяемый терм имеет заголовок "и". Затем создается задача на описание Z, посылками которой служат антецеденты теоремы, а условиями - конъюнктивные члены заменяемого терма. Цели задачи - "полный", "явное", "прямойответ", "редакция", "соединение". Копия задачи Z решается до максимального уровня 5. Если заменяющий терм A теоремы приема оказался короче ответа, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ короче(A))".

3. Свертка группы известных условий задачи на описание при редактировании ответа. Тип приема - "комментариииусловия".

   Проверяется, что заменяющее утверждение элементарно. Создается задача на описание Z, посылки которой суть антецеденты теоремы, а условия - конъюнктивные члены заменяемого утверждения. Цели задачи - "полный", "явное", "прямойответ", "редакция". Копия задачи Z решается до максимального уровня 5. Если ответ задачи не является элементарным утверждением, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ элементарно)".

Кванторные свертки и расшифровки

1. Кванторная свертка. Тип приема - "кванторнаясвертка".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - ее заменяемый терм A. Единственная цель - "упростить". Копия задачи Z решается до максимального уровня 7. Если ответ имеет связанные переменные, а заменяющий терм теоремы - не имеет, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ свобоперанд)".

2. Кванторная расшифровка. Тип приема - "теквхожд".

   Составляется список X параметров заменяемого утверждения A. Список антецедентов разбивается на подсписок B_1 утверждений, содержащих переменную списка X, и подсписок B_2 остальных утверждений. Создается задача на описание Z, посылками которой служат утверждения B_2, а условиями - утверждения B_1 и заменяемый терм A. Цели задачи - "полный", "прямойответ", "попыткаспуска", "редуцирование", "одз", "неизвестные X". Копия задачи Z решается до максимального уровня 5. Если ответ не содержит кванторов, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ содержится(длялюбого существует))".

3. Кванторная расшифровка в условии задачи на доказательство. Тип приема - "развертка".

   Создается задача на доказательство Z, посылками которой служат антецеденты теоремы и заменяющий терм, а условием - заменяемый терм. Ей передаются комментарии "кванторнаясвертка", "противоречие", "теортест". Копия задачи

Z решается до максимального уровня 7. Если получен "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(7))".

4. Кванторная расшифровка посылки, приводящая к квантору общности. Тип приема - "задача".

   Если заменяющий терм - квантор существования, то рассматриваются квантор общности K, являющий отрицанием этого квантора существования, а также отрицание A заменяемого терма. Иначе K - заменяющий терм, A - заменяемый. Если консеквент кванторной импликации K - отрицание, причем некоторый антецедент тоже является отрицанием, то выполняется их контрапозиция. Затем создается задача на доказательство Z, посылки которой суть антецеденты теоремы, утверждение A и антецеденты кванторной импликации K. Условие задачи - консеквент импликации K. Копия задачи Z решается до максимального уровня 6. Если получен "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(6))".

5. Кванторная расшифровка посылки, приводящая к квантору существования. Тип приема - "облнорм".

   Если заменяющий терм - квантор общности, то рассматриваются квантор существования K, являющий отрицанием этого квантора общности, а также отрицание A заменяемого терма. Иначе K - заменяющий терм, A - заменяемый. Проверяется, что заголовок K - символ "существует" и что ни один из конъюнктивных членов утверждения под квантором существования не имеет заголовка "значение". Затем создается задача на доказательство Z, посылки которой суть антецеденты теоремы и утверждение A. Условие задачи - утверждение K. Копия задачи Z решается до максимального уровня 7, причем используется средний ограничитель трудоемкости. Если получен "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(7))".

6. Кванторная расшифровка в режиме развертки. Тип приема - "контрольбуфера".

   Создается задача на описание Z, посылками которой служат антецеденты теоремы, а единственным условием - заменяемый терм A. Цели задачи - "редакция", "развертка". Копия задачи Z решается до максимального уровня 5. Если ответ не содержит кванторов, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ содержится(длялюбого существует))".

Применение параметрического описания

1. Попытка использования явного параметрического описания при получении частичного ответа. Тип приема - "параметризация".

   Консеквент теоремы - эквивалентность некоторого утверждения A квантору существования K. Пусть X - связывающая приставка этого квантора, S - список конъюнктивных членов подкванторного утверждения. В списке S находится равенство $y = t$, где y - переменная, не имеющая других вхождений в K.

Решается задача на описание, посылками которой служат антецеденты теоремы, а условиями - отличные от указанного равенства элементы списка S. Цели задачи - "полный", "пример", "неизвестные X". Уровень обращения к задаче равен 6. Проверяется, что ответ отличен от символа "отказ", т.е. пример значений параметров параметрического описания решатель способен найти. Тогда составляется список P, состоящий из всех содержащих y антецедентов теоремы и из утверждения A. Создается задача на описание Z, посылками которой служат все не вошедшие в P антецеденты теоремы, а условиями - утверждения P. Цели задачи - "полный", "пример", "неизвестные y". Копия задачи Z решается до максимального уровня 8. Если ответом служит символ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(8))".

2. Попытка использования неявного параметрического описания при получении частичного ответа. Тип приема - "попыткапараметризации".

   Консеквент теоремы - эквивалентность некоторого утверждения A квантору существования K. Пусть S - список конъюнктивных членов подкванторного утверждения. Решается задача на описание, имеющая своей единственной посылкой константу "истина", а условиями - антецеденты теоремы и утверждения S. Цели задачи - "полный", "пример", "неизвестные X", где X - все параметры условий задачи. Уровень обращения к задаче равен 6. Используется сравнительно сильный ограничитель трудоемкости. Проверяется, что ответ отличен от символа "отказ". Тогда создается задача на описание Z, имеющая своей единственной посылкой константу "истина", а условиями - антецеденты теоремы и утверждение A. Цели задачи - "полный", "пример", "неизвестные Y", где Y - все параметры условий задачи. Копия задачи Z решается до максимального уровня 8, причем используется средний ограничитель трудоемкости. Если ответом служит символ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(8))".

Преобразование утверждений с неизвестными

1. Разрешение относительно неизвестных.

   (a) Разрешение условия либо посылки относительно заданных неизвестных.

      i. Разрешение условия задачи на описание либо посылки задачи на исследование относительно заданных неизвестных. Тип приема - "глуб".

         В спецификации находится элемент "неизвестные($x_1 \ldots x_k$)". Составляется список S всех антецедентов, содержащих хотя бы одну переменную списка $x_1, \ldots, x_k$, и список P остальных антецедентов. Создается задача на описание Z, посылками которой служат утверждения P, а условиями - утверждения S и заменяемый терм теоремы. Цели задачи - "полный", "явное", "прямойответ", "упростить", "одз", "неизвестные $x_1 \ldots x_k$". Копия задачи Z решается до максимального уровня 8. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(8))".

      ii. Разрешение условия задачи на описание относительно заданных неизвестных.

A. Условие задачи на описание преобразуется в явное параметрическое описание неизвестных. Тип приема - "серия".

В спецификации находится элемент "неизвестная($x_1 \dots x_k$)". Составляется список S всех антецедентов, содержащих хотя бы одну переменную списка $x_1, \dots, x_k$, и список P остальных антецедентов. Создается задача на описание Z, посылками которой служат утверждения P, а условиями - утверждения S и заменяемый терм теоремы. Цели задачи - "полный", "явное", "прямойответ", "упростить", "одз", "неизвестные $x_1 \dots x_k$". Копия задачи Z решается до максимального уровня 8. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(8))".

(b) Разрешение группы утверждений относительно заданных неизвестных.

i. Разрешение группы условий задачи на описание относительно заданных неизвестных. Тип приема - "область".

В спецификации находится элемент "неизвестные($x_1 \dots x_k$)". Составляется список S всех антецедентов, содержащих хотя бы одну переменную списка $x_1, \dots, x_k$, и список P остальных антецедентов. Создается задача на описание Z, посылками которой служат утверждения P, а условиями - утверждения S и заменяемый терм теоремы. Цели задачи - "полный", "явное", "прямойответ", "упростить", "одз", "неизвестные $x_1 \dots x_k$". Копия задачи Z решается до максимального уровня 5. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

ii. Группа условий задачи на описание преобразуется в явное параметрическое описание неизвестных. Тип приема - "о".

Действия те же, что в предыдущем пункте. При решении копии задачи Z используется средний ограничитель трудоемкости.

2. Дизъюнктивно-конъюнктивные декомпозиции.

(a) Дизъюнктивно-конъюнктивная декомпозиция условия задачи на описание либо посылки задачи на исследование относительно неизвестных.

i. Дизъюнктивно-конъюнктивная декомпозиция условия задачи на описание относительно неизвестных. Тип приема - "нормнок".

Создается задача на описание Z, имеющая своей единственной посылкой константу "истина", а условиями - антецеденты и заменяемую часть теоремы. Цели задачи - "попыткаспуска", "полный", "прямойответ", "редуцирование", "упростить", "одз", "неизвестные X", где X - все параметры заменяемой части. Копия задачи Z решается до максимального уровня 8. Если набор конъюнктивных членов ответа не представляет собой декомпозицию заменяемой части относительно ее параметров, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ разбиение(X))".

ii. Дизъюнктивно-конъюнктивная декомпозиция условия задачи на описание относительно заданных неизвестных, использующая конечное перечисление относительно константных параметров. Тип приема - "ключ".

В спецификации находится элемент "см($A_1 \dots A_n$)". Составляется список S тех A_i, которые имеют заголовок "целое" либо "натуральное". Пусть X - список всех переменных термов списка S. Процедура "вхконст" перечисляет некоторые наборы K константных термов для значений переменных списка X, в соответствии с условиями на эти значения, содержащимися в антецедентах теоремы и в утверждениях S. Определяется результат P подстановки термов K вместо переменных X в заменяемую часть теоремы. В спецификации находится терм "неизвестные($y_1 \dots y_m$)". Определяется список B результатов подстановки термов K вместо переменных X во все антецеденты теоремы, содержащие параметры терма P. Список B разбивается на подсписок B_1 всех утверждений, содержащих переменные $y_1, \dots, y_m$, и подсписок B_2 остальных утверждений. Создается задача на описание Z, посылки которой суть утверждения B_2, а условия - утверждения B_1 и утверждение P. Цели задачи - "полный", "попыткаспуска", "прямойответ", "редуцирование", "упростить", "или", "одз", "неизвестные $y_1 \dots y_m$". Копия задачи Z решается до максимального уровня 7. Если ответ не содержит символа "или", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ содержится(или))".

3. Свертка нескольких условий задачи на описание в одно условие.

   (a) Свертка группы явно разрешенных относительно неизвестных условий в одно, тоже явно разрешенное относительно неизвестных. Тип приема - "кн".

   В спецификации находится элемент "неизвестные(x)". Список антецедентов теоремы разбивается на подсписок S_1 утверждений, содержащих x, и подсписок S_2 остальных утверждений. Создается задача на описание Z, посылки которой суть утверждения S_2, а условия - утверждения S_1 и конъюнктивные члены заменяемой части теоремы. Цели задачи - "полный", "явное", "прямойответ", "неизвестные x". Копия задачи Z решается до максимального уровня 5. Проверяется, что среди конъюнктивных членов ответа имеется утверждение, глубина вхождения в которое переменной x равна 1, а длина остальных конъюнктивных членов с x равна 2. Если это неверно, причем глубина вхождения x в заменяющий терм равна 1, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ глуб)".

4. Упрощение утверждений относительно неизвестных.

   (a) Расшифровка неизвестного подутверждения условия задачи на описание. Тип приема - "содержится".

   Проверяется, что заменяемый терм имеет единственное подвыражение t максимальной сложности. Создается задача на описание Z, единственная

посылка которой - константа "истина", а условия - конъюнктивные члены заменяемого утверждения. Цели задачи - "полный", "явное", "попыткаспуска", "одз", "неизвестные X", где X - параметры заменяемого утверждения. Копия задачи Z решается до максимального уровня 12. Если ответ содержит заголовок s терма t, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(s))".

(b) Группировка дизъюнктивных членов относительно неизвестной. Тип приема - "имп".

В спецификации находится элемент "неизвестные(x)". Список антецедентов теоремы разбивается на подсписок S_1 всех утверждений, содержащих x, и подсписок S_2 остальных утверждений. Создается задача на описание Z, посылки которой суть утверждения S_2, а условия - утверждения S_1 и конъюнктивные члены заменяемого утверждения. Цели задачи - "полный", "явное", "прямойответ", "неизвестные x". Копия задачи Z решается до максимального уровня 5. Если среди заголовков конъюнктивных членов ответа встречается символ "или", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(или))".

(c) Преобразование условия задачи на описание либо посылки задачи на исследование, исключающее сложное выражение с неизвестными. Тип приема - "прообраз".

В спецификации находится элемент "см(A)". Утверждение A преобразуется к виду д.н.ф, и среди дизъюнктивных членов результата выбирается некоторое утверждение B. Составляется список B' его конъюнктивных членов. Определяется список P всех переменных, выделенных в B' термами "известно(...)". Определяется также список K всех переменных, выделенных в B' термами "натуральное(...)", "целое(...)", "десчисло(...)". Рассматривается теорема T, являющаяся источником текущей теоремы приема. Вместо переменных списка K в нее подставляются последовательные константы $2, 3, \ldots$. Результат подвергается общей стандартизации. Если заменяемая часть теоремы T (после указанных преобразований) имеет два корневыз операнда, один из которых является переменной, то эта переменная добавляется к списку P. Составляется список X всех переменных заменяемой части теоремы T, не вошедших в список P. Проверяется, что наибольшая из оценок сложности содержащих переменные X подтермов заменяющей части теоремы T меньше наибольшей оценки сложности содержащих переменные X подтермов заменяемой части. Список антецедентов теоремы T разбивается на подсписок S_1 утверждений, содержащих переменные X, и подсписок S_2 остальных утверждений. Создается задача на описание Z, посылки которой суть утверждения S_2, а условия - утверждения S_1 и заменяемое утверждение теоремы T. Цели задачи - "полный", "явное", "попыткаспуска", "перемОценка", "неизвестные X". Копия задачи Z решается до максимального уровня 7. Если максимум M оценок сложности содержащих переменные X подтермов заменяющей части теоремы T меньше максимума оценок сложности содержащих переменные X подтермов ответа, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ перемОценка(M набор(X)))".

Упрощение утверждений с описателями

1. Использование эквивалентности для упрощения описателя. Тип приема - "посылка".

   Создается задача на преобразование, посылками которой служат антецеденты теоремы, а условием - заменяемый терм. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 5. Если процедура "упрощописатель" не усматривает, что при переходе от заменяемого терма к ответу задачи описатели стали более простыми (например, отличающимися от исходных отбрасыванием части своих фрагментов), то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ упрощописатель)".

2. Упрощение описателя путем перехода к параметрическому описанию. Тип приема - "разряд".

   Действия те же, что в предыдущем пункте.

3. Разрешение относительно переменной, связанной внешним описателем.

   (a) Параметрическое описание переменной, связанной внешним описателем "класс". Тип приема - "тригповтор".

   Проверяется, что заголовок заменяющей части - квантор существования. Среди конъюнктивных членов его подкванторного утверждения находится равенство $x = t$, где x - параметр заменяемой части теоремы. Список антецедентов теоремы разбивается на подсписок S_1 утверждений, содержащих x, и подсписок S_2 остальных утверждений. Формируется выражение P вида "$\mathrm{set}_x(A(x))$", где $A(x)$ - конъюнкция утверждений списка S_1 и заменяемой части теоремы. Затем создается задача на преобразование Z, посылками которой служат утверждения S_2, а условием - выражение P. Цели задачи - "одз", "упростить". Проверяется, что заменяемый терм задачи имеет единственный подтерм максимальной сложности, и находится заголовок s этого подтерма. Копия задачи Z решается до максимального уровня 6. Если символ s входит в ответ, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(s))".

   (b) Явное разрешение утверждения относительно переменной, связанной внешним описателем. Тип приема - "параллели".

   В спецификации находится элемент "переменная(x)". Список антецедентов теоремы разбивается на подсписок S_1 утверждений, содержащих переменную x, и подсписок S_2 остальных утверждений. Выбирается переменная f, не входящая в теорему, и формируется выражение P вида "$\lambda_x(f(x), A(x))$", где $A(x)$ - конъюнкция утверждений списка S_1 и конъюнктивных членов заменяемой части теоремы. Создается задача на преобразование Z, посылками которой служат утверждения S_2, а условием - выражение P. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 6. В ответе находится подвыражение вида "$\lambda_y(B(y), C(y))$".

Если глубина вхождения переменной y в терм $C(y)$ не меньше глубины r вхождения переменной x в терм $A(x)$, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ варьир(r))".

(c) Группировка элементов описания класса, явно разрешенных относительно переменной связывающей приставки. Тип приема - "десделение".

В спецификации находится элемент "переменная(x)". Список антецедентов теоремы разбивается на подсписок S_1 утверждений, содержащих переменную x, и подсписок S_2 остальных утверждений. Выбирается переменная f, не входящая в теорему, и формируется терм P вида "$\mathrm{set}_x(A(x))$", где $A(x)$ - конъюнкция утверждений списка S_1, конъюнктивных членов заменяемой части теоремы и терма $f(x)$. Создается задача на преобразование Z, посылками которой служат утверждения S_2, а условием - выражение P. Цели задачи - "одз", "известны", "упростить". Копия задачи Z решается до максимального уровня 6. Проверяется, что ответ имеет вид "$\mathrm{set}_y(B(y))$". Если число вхождений в ответ переменной y не меньше числа n вхождений переменной x в терм P, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ связка(n))".

Упрощение кванторов

1. Декомпозиция кванторной импликации в конъюнкцию нескольких кванторных импликаций и элементарных утверждений. Тип приема - "Примечпосылки".

   Создается задача на преобразование Z, посылками которой служат антецеденты, а условием - заменяемый терм. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 5. Если заголовок ответа отличен от символа "и", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(и))".

2. Отбрасывание избыточных обобщенных антецедентов кванторной импликации. Тип приема - "двойнойаргумент".

   Создается задача на преобразование Z, посылками которой служат антецеденты, а условием - заменяемый терм t. Цели задачи - "упростить", "одз". Копия задачи Z решается до максимального уровня 5. Проверяется, что ответ имеет заголовок "длялюбого". Если число антецедентов ответа не меньше числа антецедентов терма t, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ сокращант(t))".

Упрощение проверяемого условия

1. Расшифровка по определению проверяемого условия. Тип приема - "вставкафрагментов".

   Создается задача на доказательство Z, посылками которой служат антецеденты теоремы, конъюнктивные члены заменяющего утверждения и терм "авт". Условием служит заменяемое утверждение. Задача сопровождается комментарием "одз". Копия задачи Z решается до максимального уровня 6. Если не

получается ответ "истина", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(6))".

2. Преобразование условия задачи на доказательство, исключающее сложное выражение. Тип приема - "исключение".

   Создается задача на доказательство Z, посылками которой служат антецеденты теоремы и конъюнктивные члены заменяющего утверждения. Условием служит заменяемое утверждение. Задача сопровождается комментарием "одз". Копия задачи Z решается до максимального уровня 6. Если не получается ответ "истина", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(6))".

Исключение сложных понятий

1. Уменьшение числа сложных понятий. Тип приема - "целые".

   Рассматривается заменяемое утверждение A. Если оно имеет параметры $x_1, \ldots,$ x_k, не встречающиеся в антецедентах, то далее в качестве A берется выражение "$\mathrm{set}_{x_1 \ldots x_k} A$". Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - терм A. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 5. Оператор "уменьшсложн" проверяет, что оценки сложности ответа и терма A равны, причем ответ имеет единственное вхождение самой сложной операции, а терм A - несколько самых сложных операций, декомпозирующих первую по своим переменным. Если это условие не выполняется, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ уменьшсложн)".

Координаты

1. Переформулировка утверждений через координаты.

   (a) Переформулировка утверждений через координаты. Тип приема - "вписана".

   Создается задача на преобразование Z, посылки которой суть антецеденты теоремы, условие - ее заменяемый терм. Единственная цель задачи - "упростить". Копия задачи Z решается до максимального уровня 7. Если параметры ответа не включаются в список параметров заменяющего терма t, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ переменные(t))".

16.5 Усмотрение истинности либо ложности

Непосредственное усмотрение истинности либо ложности

Тип приема - "элементызадачи". Создается задача на доказательство Z, посылки которой суть антецеденты теоремы, а условие - ее консеквент. Задача имеет комментарий "одз". Копия задачи Z решается до максимального уровня 5. Если не получен

ответ "истина", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

Усмотрение истинности либо ложности с помощью проверочного оператора

1. Усмотрение истинности либо ложности с помощью проверочного оператора. Тип приема - "блокпрограммы".

   Создается задача на доказательство Z, посылки которой суть антецеденты теоремы, а условие - ее консеквент. Задача имеет комментарий "одз". Копия задачи Z решается до максимального уровня 5. Если не получен ответ "истина", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

Усмотрение истинности либо ложности из утверждений, содержащихся в контексте

1. Усмотрение истинности либо ложности из утверждений, содержащихся в контексте. Тип приема - "обл".

   Аналогично предыдущему пункту, но максимальный уровень равен 6, а при решении задачи используется средний ограничитель трудоемкости.

16.6 Вывод в посылках

Вывод одного отношения

1. Вывод одноместного отношения.

   (a) Вывод одноместного предиката. Тип приема - "антецедент".

   Создается задача на доказательство Z, посылки которой суть антецеденты теоремы, а условие - ее консеквент. Задача имеет комментарий "одз". Копия задачи Z решается до максимального уровня 5. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

2. Вывод равенства.

   (a) Вывод равенства для числовых атомов.

   i. Равенство двух невырожденных числовых атомов.

   A. Равенство двух невырожденных числовых атомов. Тип приема - "редакторответа".

   Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Создается задача на доказательство Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Задача имеет комментарий "одз". Копия задачи Z решается до максимального уровня 6. Если получен ответ

"отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

ii. Соотношение пропорциональности для двух невырожденных числовых атомов.

A. Соотношение пропорциональности для двух числовых атомов. Тип приема - "оценка".
Действия те же, что в предыдущем пункте. При решении задачи вводится слабый ограничитель трудоемкости.

iii. Вывод явного выражения для значения невырожденного числового атома.

A. Вывод явного выражения для значения числового атома. Тип приема - "частичныйответ".

Определяется список S антецедентов теоремы, пополненный утверждениями, необходимыми для сопровождения этого же списка по о.д.з. Утверждения "разныеточки", "разныепрямые" заменяются в списке S на отрицания равенств соответствующих точек и прямых. Проверяется, что левая часть консеквента теоремы - невырожденный числовой атом A. Находится список $B_1, \ldots, B_n$ всех числовых атомов правой части консеквента. Выбираются переменные $a_1, \ldots, a_n$, не входящие в теорему, и к списку S добавляются равенства $B_1 = a_1, \ldots, B_n = a_n$. Выбирается переменная x, не встречающаяся в теореме и отличная от переменных $a_1, \ldots, a_n$. Создается задача на описание Z, посылками которой служат утверждения S, а условием - утверждения "$x = A$" и "x - число". Цели задачи - "полный", "явное", "прямойответ", "неизвестные x", "известно $a_1 \ldots a_n$". Копия задачи Z решается до максимального уровня 10. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(10))".

B. Выражение текущего числового атома через более простые числовые атомы. Тип приема - "усмцелое".

Определяется список S антецедентов теоремы, пополненный утверждениями, необходимыми для сопровождения этого же списка по о.д.з. Утверждения "разныеточки", "разныепрямые" заменяются в списке S на отрицания равенств соответствующих точек и прямых. Находится список A всех числовых атомов консеквента. Проверяется, что среди них встречается имеющий максимальную сложность подтерм t консеквента теоремы. Проверяется также, что длина любого другого имеющего максимальную сложность подвыражения консеквента (если такой имеется) меньше длины t. Рассматриваются все отличные от t элементы $A_1, \ldots, A_m$ списка A. Выбираются не входящие в теорему переменные $a_1, \ldots, a_m$, и к списку S добавляются равенства $A_1 = a_1, \ldots, A_m = a_m$. Выбирается не входящая в теорему и отличная от $a_1, \ldots, a_m$ переменная x. Создается задача на описание Z, посылками которой служат утверждения S, а условиями - утверждения "число(x)" и "$x = t$". Цели задачи - "полный", "явное", "прямойответ", "неизвестные x", "известно

$a_1 \ldots a_m$". Копия задачи Z решается до максимального уровня 10. Используется средний ограничитель трудоемкости. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(10))".

iv. Общий случай соотношения для невырожденных числовых атомов.

Напомним, что многие типы приемов возникают лишь при доводке. Изначально доводчику передается версия приема со слабыми ограничениями на срабатывание. В процессе прокрутки по задачнику выявляются точки, где его срабатывание нежелательно, и тогда предпринимаются поыптки усилить ограничения за счет изменения типа приема. По этой причине, в данном разделе имеются лишь один тип приема, обладающий сравнительно слабыми ограничениями.

A. Соотношение для старых числовых атомов. Тип приема - "род".

Определяется список S антецедентов теоремы. Утверждения "разныеточки", "разныепрямые" заменяются в списке S на отрицания равенств соответствующих точек и прямых. Находится список A всех числовых атомов консеквента. Проверяется, что все они невырожденные, а их количество не менее двух. Выбирается элемент t списка A. Рассматриваются все отличные от t элементы $A_1, \ldots, A_m$ списка A. Выбираются не входящие в теорему переменные $a_1, \ldots, a_m$, и к списку S добавляются равенства $A_1 = a_1, \ldots, A_m = a_m$. Для каждого A_i, имеющего своим заголовком один из символов "расстояние", "угол", "площадь", "объем", "масса", к списку S добавляется неравенство "$0 < A_i$". Выбирается не входящая в теорему и отличная от $a_1, \ldots, a_m$ переменная x.

Предпринимается попытка пополнить список S. Для этого определяется результат K замены в консеквенте теоремы подтермов $A_1, \ldots, A_m, t$ на переменные $a_1, \ldots, a_m, x$. Создается вспомогательная задача на описание Z', имеющая своей единственной посылкой константу "истина", а условием - равенство K. Ее цели - "полный", "явное", "прямойответ", "или", "упростить", "одз", "неизвестные x". Задача Z' решается до максимального уровня 6 со средним ограничителем трудоемкости. Если ответ отличен от "отказ", то он приводится к виду д.н.ф. и рассматривается список Q конъюнктивных членов какого-либо дизъюнктивного члена этой д.н.ф. Составляется список Q' всех не содержащих x элементов списка Q. Проверяется, что среди них нет равенства. Составляется список P, полученный добавлением к списку Q' равенств $A_1 = a_1, \ldots, A_m = a_m$. Решается задача на исследование с посылками P и целями "противоречие", "одз". Максимальный уровень равен 6. Если в итоговый список посылок этой задачи не входит константа "ложь", то элементы списка Q' добавляются к списку S, и рассмотрение дизъюнктивных членов ответа задачи Z' на этом завершается. Иначе - переход к очередному такому дизъюнктивному члену.

После попытки пополнения списка S создается задача на описание Z, посылками которой служат утверждения S, а условием - равен-

ство $x = t$. Цели задачи - "полный", "явное", "примерка", "прямой-ответ", "упростить", "одз", "неизвестные x", "известно $a_1 \dots a_m$". Для усмотрения планиметрических задач используется анализатор "планиметрия", добавляющий при необходимости посылку "планиметрия" к списку посылок задачи Z.

Копия задачи Z решается до максимального уровня 13. Используется средний ограничитель трудоемкости. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(13))". Если превышен лимит трудоемкости, то тоже предпринимается обращение к процедуре "регприем".

(b) Вывод равенства старых объектов. Тип приема - "склейкаоперандов".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Создается задача на преобразование Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Цели задачи - "одз", "Посылка".

Цель "Посылка" означает, что решение задачи на преобразование сразу сводится к вспомогательной задаче на исследование, цели которой суть все остальные цели задачи на преобразование, а посылки - те же, что у задачи на преобразование. Если задача на исследование изменяет посылки таким образом, что возникает утверждение, для которого результат применения оператора "станд" такой же, как для условия задачи на преобразование, то ответом задачи на преобразование служит 1, иначе - 0.

Копия задачи Z решается до максимального уровня 5. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(1))".

(c) Вывод нечислового равенства.

i. Вывод нечислового равенства. Тип приема - "тела".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Создается задача на доказательство Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Задача имеет комментарий "одз". Копия задачи Z решается до максимального уровня 5. Используется средний ограничитель трудоемкости. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

(d) Вывод равенства старого объекта константному выражению. Тип приема - "последнийоперанд".

Создается задача на преобразование, посылки которой суть антецеденты теоремы, а условие - консеквент. Цели задачи - "одз", "Посылка". Копия задачи Z решается до максимального уровня 5. Если получен ответ 0, то

предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(1))".

3. Вывод двуместного отношения, не являющегося равенством.

   (a) Вывод двуместного отношения в задаче на исследование либо на доказательство. Тип приема - "свертка".

   Определяется список S антецедентов теоремы, пополненный утверждениями, необходимыми для сопровождения этого же списка по о.д.з. Утверждения "разныеточки", "разныепрямые" заменяются в списке S на отрицания равенств соответствующих точек и прямых. Создается задача на доказательство Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Копия задачи Z решается до максимального уровня 5. Используется средний ограничитель трудоемкости. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

   (b) Вывод двуместного отношения в задаче на описание, имеющей цель "пример". Тип приема - "сдвигоператора".

   Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Выбираются параметр a консеквента теоремы, а также переменная x, не входящая в теорему. Определяется результат A замены в консеквенте теоремы переменной a на переменную x. Создается задача на описание Z, имеющая список посылок S и единственное условие A. Цели задачи - "пример", "полный", "прямойответ", "неизвестные x". Копия задачи Z решается до максимального уровня 5. Если получен "отказ" либо ответ содержит хотя бы один из символов "элемент", "внешнийэлемент", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(элемент внешнийэлемент))".

4. Вывод многоместного отношения.

   (a) Вывод многоместного отношения. Тип приема - "унификация".

   Определяется список S антецедентов теоремы, пополненный утверждениями, необходимыми для сопровождения этого же списка по о.д.з. Утверждения "разныеточки", "разныепрямые" заменяются в списке S на отрицания равенств соответствующих точек и прямых. Создается задача на преобразование Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Цели задачи - "одз", "Посылка". Копия задачи Z решается до максимального уровня 5. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(1))". Предварительно проверяется, имеется ли проверочный оператор для обработки консеквента теоремы. Если имеется, причем решаемая до малого уровня задача на доказательство усматривает истинность консеквента из посылок S, то обращение к процедуре "регприем" отменяется.

5. Вывод отрицания равенства. Тип приема - "первыйсимвол".

   Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Создается задача на доказательство Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Задача имеет комментарий "одз". Копия задачи Z решается до максимального уровня 5. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

Контроль противоречивости при разборе случаев

1. Усмотрение противоречия в посылках задачи на исследование, возникшей при разборе случаев. Тип приема - "известны".

   Создается задача на преобразование Z, посылками которой служат антецеденты теоремы, а условием - консеквент (в данном случае - логическая константа "ложь"). Цели задачи - "Посылка", "известно", "контроль". Копия задачи Z решается до максимального уровня 3. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опций "(ответ заголовок(1))", "урвн(3)".

Вывод утверждения существования

1. Вывод утверждения существования. Тип приема - "продолжение".

   Создается задача на доказательство Z, посылками которой служат антецеденты теоремы, а условием - ее консеквент. Копия задачи Z решается до максимального уровня 5. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

Вывод кванторной импликации

1. Вывод кванторной импликации, определяющей значения серии невырожденных числовых атомов. Тип приема - "решить".

   Рассматривается список S антецедентов теоремы. Если в спецификации приема имеются элементы "указатель(занесениепосылки(i A))" и "указатель(новаяпеременная(x))", то находится i-й элемент B списка S, и он заменяется в списке S на "$\forall_x(A \to B)$". Затем создается задача на преобразование Z, посылками которой служат утверждения S, а условием - консеквент теоремы. Задача имеет единственную цель "Посылка".

   Цель "Посылка" означает, что решение задачи на преобразование сразу сводится к вспомогательной задаче на исследование, цели которой суть все остальные цели задачи на преобразование, а посылки - те же, что у задачи на преобразование. Если задача на исследование изменяет посылки таким образом, что возникает утверждение, для которого результат применения оператора "станд" такой же, как для условия задачи на преобразование, то ответом задачи на преобразование служит 1, иначе - 0.

Копия задачи Z решается до максимального уровня 4. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опций "(ответ заголовок(1))", "урвн(4)".

Вывод определения

Тип приема - "нормуглы". Создается задача на доказательство Z, посылки которой суть антецеденты теоремы, а условие - ее консеквент. Копия задачи Z решается до максимального уровня 7. Используется средний ограничитель трудоемкости. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(7))".

Координаты

1. Связь числовых атомов с координатами.

   (a) Связь числовых атомов с координатами. Тип приема - "формоперанды".

   Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Проверяется, что консеквент теоремы - равенство, содержащее единственный невырожденный числовой атом t. Находятся все численные параметры $a_1, \ldots, a_n$ консеквента. Выбирается переменная x, не входящая в теорему, и создается задача на описание Z, посылками которой служат утверждения S, а единственным условием - равенство "$t = x$". Цели задачи - "полный", "явное", "прямойответ", "неизвестные x", "известно $a_1 \ldots a_n$". Копия задачи Z решается до максимального уровня 7. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(7))".

2. Связь текущего числового атома с координатами.

   (a) Связь текущего числового атома с координатами. Тип приема - "Операнды".

   Действия те же, что в предыдущем пункте.

3. Вывод соотношений для координат объектов.

   (a) Вывод соотношений для координат объектов. Тип приема - "больше".

   Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Создается задача на доказательство Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Копия задачи Z решается до максимального уровня 7. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(7))".

4. Выражение координатных наборов через числовые атомы.

(a) Выражение координатных наборов через числовые атомы. Тип приема - "удаление".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Создается задача на преобразование Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Цели задачи - "одз", "Посылка". Копия задачи Z решается до максимального уровня 5. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(1))".

(b) Выражение координатных наборов через численные параметры. Тип приема - "стандравно".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Проверяется, что консеквент теоремы - равенство $t_1 = t_2$, где t_1 - обозначение каких-либо координат (в данном случае - координатного набора). Рассматриваются все параметры $a_1, \ldots, a_n$ терма t_2. Выбирается переменная x, не входящая в теорему, и создается задача на описание Z, посылками которой служат утверждения S, а единственным условием - равенство "$t_1 = x$". Цели задачи - "полный", "явное", "прямой-ответ", "неизвестные x", "известно $a_1 \ldots a_n$". Копия задачи Z решается до максимального уровня 7. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(7))".

5. Выражение текущего координатного набора через числовые атомы.

(a) Выражение текущего координатного набора через числовые атомы. Тип приема - "регистрациячертежа".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. В спецификации приема находится элемент "терм(t)", и к списку S добавляется терм "актив(t)". Создается задача на преобразование Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Цели задачи - "одз", "Посылка". Копия задачи Z решается до максимального уровня 6. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(1))".

(b) Выражение текущего координатного набора через численные параметры. Тип приема - "степени".

Действия те же, что для приведенного чуть выше приема типа "стандравно".

6. Вывод соотношений для отдельных координат и числовых атомов. Тип приема - "текпосылка".

Создается задача на доказательство Z, посылки которой суть антецеденты теоремы, а условие - ее консеквент. Задача имеет комментарий "одз". Копия задачи Z решается до максимального уровня 5. Используется средний ограничитель трудоемкости. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

7. Вывод соотношения, связывающего текущую координату с другими координатами и числовыми атомами. Тип приема - "коррекциязнака".

   Действия те же, что в предыдущем пункте.

8. Вывод ограничений на координаты объектов. Тип приема - "наборслов".

   Действия те же, что в предыдущем пункте.

9. Координаты множества объектов.

   (a) Вывод уравнения для координат множества объектов

      i. Вывод уравнения для координат множества объектов

         A. Вывод уравнения для координат множества объектов. Тип приема - "значениепеременной".

            Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Создается задача на доказательство Z, посылки которой суть утверждения списка S, а условие - консеквент теоремы. Задача имеет комментарий "одз". Копия задачи Z решается до максимального уровня 5. Используется средний ограничитель трудоемкости. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

      ii. Вывод уравнения для текущих координат множества объектов.

         A. Вывод уравнения для текущих координат множества объектов. Тип приема - "семействомножеств".

            Действия те же, что в предыдущем пункте.

         B. Вывод общего вида уравнения для текущих координат множества объектов. Тип приема - "фильтротрезков".

            Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Рассматриваются все конъюнктивные члены консеквента, представляющие собой равенства, у которых в левой части находится обозначение координат, и составляется список $A_1, \ldots, A_n$ левых частей этих равенств. Создается задача на преобразование Z, посылки которой суть утверждения S, а условие - терм A вида "набор($A_1 \ldots A_n$)". Цель задачи - "неопр".

Цель "неопр" сразу переключает решение задачи на вспомогательную задачу на исследование, посылки которой получаются добавлением к посылкам задачи Z терма "неопр(A)". После решения задачи на исследование в ее посылках находится терм "неопр(B)", и B выдается как ответ.

Копия задачи Z решается до максимального уровня 5. Если ответ содержит хотя бы один из заголовков $s_1, \ldots, s_k$ термов $A_1, \ldots, A_n$, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол($s_1 \ldots s_k$))".

iii. Вывод параметрического описания для текущих координат множеств объектов. Тип приема - "Преобр".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Создается задача на преобразование, посылки которой суть утверждения S, а условие - левая часть консеквента теоремы. Цели задачи - "вспомпараметр", "неопр".

Цель "вспомпараметр" передается вспомогательной задаче на исследование, создаваемой согласно цели "неопр". Она указывает на предпочтительное получение параметрического описания.

Копия задачи Z решается до максимального уровня 5. Если ответ не содержит символа "существует", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ содержится(существует))".

iv. Вывод уравнений для координат множеств объектов.

A. Вывод уравнений для координат множеств объектов. Тип приема - "арифмпрогрессия".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Составляется список $a_1 \ldots a_n$ всех обозначений координат, встречающихся в консеквенте теоремы. Создается задача на преобразование Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Цель задачи - "Посылка". Задача снабжается комментарием (цель (равно $a_1 \ldots a_n$)). Этот комментарий означает, что после решения вспомогательной задачи на исследование будет проверяться наличие в ее посылках для каждого терма a_i равенства вида $a_i = t$. Если это выполнено, выдается ответ 1, иначе - 0.

Копия задачи Z решается до максимального уровня 5. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(1))".

v. Дополнительное построение и вывод уравнений для координат множества объектов. Тип приема - "комментарий".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Составляется список A всех обозначений координат, встречающихся в консеквенте теоремы. Находится

список X всех параметров консеквента, не являющихся параметрами антецедентов. Составляется список P всех конъюнктивных членов консеквента, не имеющих связанных переменных. Создается задача на преобразование Z, посылки которой суть утверждения S, а условие - консеквент теоремы. Цель задачи - "Посылка". Задача снабжается комментарием (цель (Равно X P A)). Этот комментарий означает, что после решения вспомогательной задачи на исследование предпринимается попытка найти подстановку Q вместо переменных X в утверждения P, переводящую их в некоторые посылки задачи. Затем будет проверяться, что для каждого выражения a из A результат применения к нему подстановки Q является одной из частей какого-либо равенства в посылках. Если это выполнено, на задачу Z выдается ответ 1, иначе - 0.

Копия задачи Z решается до максимального уровня 5. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(1))".

(b) Вывод соотношений для параметров уравнений координат множеств объектов.

i. Вывод соотношений для параметров уравнений координат множеств объектов. Тип приема - "терминал".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. Создается задача на доказательство Z, посылки которой суть утверждения списка S, а условие - консеквент теоремы. Задача имеет комментарий "одз". Копия задачи Z решается до максимального уровня 5. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(5))".

ii. Вывод соотношений для параметров текущего уравнения координат множества объектов. Тип приема - "фокпараметр".

Действия те же, что в предыдущем пункте.

iii. Связь числовых атомов с параметрами уравнения для координат множества объектов. Тип приема - "функциональныйсимвол".

Действия те же, что в предыдущем пункте.

iv. Связь текущего числового атома с параметрами уравнения для координат множества объектов. тип приема - "монотоннозависит".

Действия те же, что в предыдущем пункте.

v. Вывод ограничения на параметры уравнения для координат множества объектов. Тип приема - "нормнабор".

Действия те же, что в предыдущем пункте.

(c) Выражение координат объекта через параметры уравнений для координат множеств объектов. Тип приема - "равныедлины".

Действия те же, что в предыдущем пункте.

(d) Характеризация множеств объектов, использующая уравнения для их координат.

i. Усмотрение вида множества объектов по уравнению для его координат. Тип приема - "подобны".

Действия те же, что в предыдущем пункте.

ii. Характеризация множеств объектов с помощью уравнений для координат. Тип приема - "ссылканаприем".

Действия те же, что в предыдущем пункте.

iii. Усмотрение специальной связи системы координат с исследуемым множеством объектов, заданным уравнением. Тип приема - "прямыеуглы".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и прямых. В спецификации находится элемент "цель(C)". Создается задача на преобразование Z, посылки которой суть утверждения списка S, а условие - консеквент теоремы. Задача имеет цели "Посылка", "одз", "C". Копия задачи Z решается до максимального уровня 6. Если получен ответ 0, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ заголовок(1))".

16.7 Исключение несущественных неизвестных

Исключение несущественных неизвестных в задаче на описание при невырожденном ограничении

Тип приема - "удалениеусловия".

Проверяется, что левая часть эквивалентности в консеквенте теоремы - квантор существования. Рассматривается его связывающая приставка X и создается задача на описание Z, имеющая единственную посылку "истина". Ее условиями служат антецеденты теоремы и конъюнктивные члены утверждения под квантором существования. Цели задачи - "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные X", "параметры X". Копия задачи Z решается до максимального уровня 3. Если получен ответ "отказ" либо среди параметров ответа содержатся переменные списка X, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ независит(X))".

Исключение несущественных неизвестных в задаче на описание при вырожденном ограничении

Тип приема - "обрывзадачи".

Определяется список S антецедентов теоремы, в котором утверждения "разныеточки", "разныепрямые" заменены на отрицания равенств соответствующих точек и

прямых. Проверяется, что в консеквенте теоремы расположен квантор существования. Рассматривается его связывающая приставка X и создается задача на описание Z, имеющая единственную посылку "истина". Ее условиями служат утверждения списка S и конъюнктивные члены утверждения под квантором существования. Цели задачи - "полный", "явное", "прямойответ", "одз", "упростить", "неизвестные X", "параметры X". Копия задачи Z решается до максимального уровня 3. Если получен ответ "отказ" либо среди параметров ответа содержатся переменные списка X, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опций "(ответ независит(X))", "уровень 1".

16.8 Обратный вывод

Непосредственный подбор примера значений неизвестных, не входящих в невырожденные условия

Тип приема - "подборзначений".

В спецификации приема находится элемент "неизвестная(x)". Определяется список S, получаемый добавлением к антецедентам теоремы утверждений, необходимых для сопровождения консеквента по о.д.з. Создается задача на описание Z, посылками которой служат утверждения списка S, не содержащие переменной x, а единственным условием - консеквент. Цели задачи - "полный", "пример", "одз", "прямойответ", "неизвестные x". Копия задачи Z решается до максимального уровня 6. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(6))".

Подбор параметризованного примера значения неизвестной

Тип приема - "повтор".

В спецификации приема находится элемент "неизвестная(x)". Определяется список S, получаемый добавлением к антецедентам теоремы утверждений, необходимых для сопровождения консеквента по о.д.з.

Рассматривается вспомогательная задача на описание Z', посылками которой служат утверждения списка S, не содержащие переменной x, а условиями - утверждения списка S, содержащие x. Цели задачи - "одз", "пример", "прямойответ", "неизвестные x". Эта задача решается до уровня 6. Если получен ответ, отличный от "отказ", то создается задача на описание Z, посылками которой служат утверждения списка S, не содержащие переменной x, а единственным условием - консеквент. Цели задачи - "пример", "одз", "прямойответ", "неизвестные x". Копия задачи Z решается до максимального уровня 6. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(6))".

Попытка подбора примера значения неизвестной задачи на описание

Тип приема - "смешаннаядробь".

В спецификации приема находится элемент "неизвестная(x)". Среди антецедентов теоремы имеется равенство "$x = t$". Определяется список S, получаемый добавлением к отличным от этого равенства антецедентам теоремы утверждений, необходимых для сопровождения консеквента по о.д.з. Создается задача на описание Z,

посылками которой служат утверждения списка S, не содержащие переменной x, а единственным условием - консеквент. Цели задачи - "пример", "одз", "прямойответ", "неизвестные x". Копия задачи Z решается до максимального уровня 6. Если получен ответ "отказ", то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ отказ(6))".

16.9 Пакетные операторы

Проверочный оператор

Тип приема - "спуск".

В спецификации находится элемент "оператор(P)". Составляется список S антецедентов теоремы. Если в спецификации имеется элемент "указатель(... занесениепосылки (i, A) ...)", причем список Q всех параметров утверждения A, являющихся связанными переменными консеквента теоремы, непуст, то i-й антецедент B заменяется в списке S на кванторную импликацию вида "$\forall_Q(A \to B)$". При помощи справочника "легковидеть" определяется набор $t_1 \ldots t_m$ входных термов проверочного оператора P для проверки консеквента K теоремы приема. Проверяется, что этот оператор пока не усматривает истинность K из посылок S. Тогда создается задача на доказательство Z, посылками которой служат утверждения S и вспомогательный терм "процедура(P набор($t_1 \ldots t_m$))". Условие задачи - консеквент теоремы. Предпринимается обращение к процедуре "регрием" для тестовой задачи Z и опции "(ответ легковидеть)". Такая опция означает, что вместо решения задачи Z оператор "регприем" будет тестировать работу проверочного оператора P на указанных входных данных.

Нормализатор

1. Нормализатор общей стандартизации выражений.

   (a) Прием нормализатора общей стандартизации. Тип приема - "норм".

   В спецификации приема находятся элементы "оператор(P)" и "направл(N)". Рассматриваются заменяемая и заменяющая части t_1, t_2 консеквента теоремы приема. Находится результат R обработки нормализатором P терма t_1 относительно антецедентов теоремы. Проверяется, что терм t_2 короче терма R. Тогда создается задача на преобразование Z, посылки которой суть антецеденты теоремы и вспомогательный терм "процедура(P t_1 t_2)". Условие задачи - терм t_1, единственная цель - "упростить".

   Вспомогательная посылка "процедура(...)" сразу переключает решение задачи Z на обработку терма t_1 нормализатором P; результат обработки выдается в качестве ответа задачи.

   Предпринимается обращение к процедуре "регрием" для тестовой задачи Z и опции "(ответ короче(t_2))".

2. Нормализатор сокращенной перезаписи.

(a) Прием нормализатора сокращенной перезаписи. Тип приема - "нормупростить".

Создается задача на преобразование Z, посылки которой суть антецеденты теоремы, а условие - заменяемый терм. Цели задачи - "одз", "упростить". Копия задачи Z решается до максимального уровня 5. Используется средний ограничитель трудоемкости. Если заменяющий терм t теоремы приема короче ответа, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ короче(t))".

3. Нормализатор приведения к заданным заголовкам.

(a) Непосредственное преобразование терма к нужному заголовку. Тип приема - "нормзаголовок".

В спецификации приема находится элемент "оператор(P)". Предпринимается обращение к оператору P для обработки заменяемого терма теоремы относительно списка ее антецедентов. Проверяется, что результат не имеет заголовка, требуемого для оператора P. Создается задача на преобразование Z, условием которой служит заменяемый терм t, а посылками - антецеденты теоремы, пополненные термом "допустзаголовок(t P)". Единственная цель задачи - "упростить".

Посылка "допустзаголовок(...)" сразу переключает решение задачи на применение оператора P к терму t и проверку того, что получен нужный заголовок. Если это так, выдается ответ 1, иначе - 0.

Предпринимается обращение к процедуре "регрием" для тестовой задачи Z и опции "(ответ заголовок(1))".

4. Нормализатор вычисления.

(a) Непосредственное вычисление. Тип приема - "буква".

В спецификации приема находится элемент "оператор(P)". Создается задача на преобразование Z, условием которой служит заменяемый терм t, а посылками - антецеденты теоремы, пополненные термом "Преобр(t P)". Единственная цель задачи - "упростить".

Посылка "Преобр(...)" сразу сводит решение задачи Z к решению задачи на исследование Z', посылки которой получаются из посылок задачи Z отбрасыванием терма "Преобр(...)". Этой задаче передается комментарий посылок (Преобр t P). Цели задачи - "известно", "Преобр", "неизвестные X", где X - все нечисловые параметры посылок. Вплоть до уровня 3, решение задачи Z' приводит к пополнению списка ее посылок простейшими их следствиями, а на уровне 3 предпринимается обращение к оператору P для обработки терма t относительно текущего спискапосылок. Результат t' регистрируется в комментарии посылок (Преобр t' P) вместо исходного терма t. На этом решение задачи Z' завершается. Далее терм t' извлекается из комментария и выдается как ответ задачи Z.

Копия задачи Z решается до максимального уровня 4. Если ответ содержит логический символ s - заголовок терма t, то предпринимается обращение к процедуре "регприем" для тестовой задачи Z и опции "(ответ логсимвол(s))".

5. Прием пакетного синтезатора. Тип приема - "синтезатор".

   В спецификации находится элемент "оператор(P)". Справочник "синтезатор" определяет по названию оператора P набор (T вход($x_1 \dots x_n$) выход($y_1 \dots y_k$) $A_1 \dots A_m$), задающий формат приемов синтезатора. Здесь T - шаблон реализуемых синтезатором утверждений; $x_1, \dots, x_n$ - входные переменные шаблона; $y_1, \dots, y_m$ - выходные переменные. Определяются термы $t_1, \dots, t_n, r_1, \dots, r_m$, подстановка которых в шаблон T вместо переменных $x_1, \dots, x_n, y_1, \dots, y_m$ дает консеквент Q теоремы приема. Находятся параметры $z_1, \dots, z_k$ термов $r_1, \dots, r_m$, и определяется список S результатов подстановки в антецеденты теоремы приема константных термов "фикс(1)", ..., "фикс(k)" вместо переменных $z_1, \dots, z_k$. Определяются результаты $R_1, \dots, R_n$ такой же подстановки в термы $t_1, \dots, t_n$.

   Проверяется, что оператор P, будучи применен к входным данным $R_1, \dots, R_n$ относительно посылок S, не дает результата. Затем создается задача на описание Z, посылками которой служат утверждения S и вспомогательный терм "процедура(P набор($R_1 \dots R_n$))". Единственное условие задачи - консеквент теоремы приема, цели - "полный", "пример", "прямойответ", "неизвестные $y_1 \dots y_m$.

   Посылка "процедура(...)" сразу переключает решение задачи Z на обращение к синтезатору P для входных данных $R_1, \dots, R_n$. Посылки - те же, что у задачи Z. Список неизвестных задачи Z, по существу, нужен только для определения числа выходных переменных синтезатора. Эти неизвестные, как видно из построения задачи, никак не связаны с ее прочими переменными. Впрочем, если работа синтезатора успешно завершается и выдается результат $u_1, \dots, u_m$, то ответом задачи Z служит конъюнкция равенств $y_1 = u_1, \dots, y_m = u_m$.

   Предпринимается обращение к процедуре "регрием" для тестовой задачи Z и опции "(ответ легковидеть)".

Глава 17

Доводка приемов

Перед описанием процедур, выполняющих завершающую доводку автоматически синтезированных приемов, еще раз кратко перечислим основные этапы цикла работы генератора приемов.

Генератор приемов запускается из просмотра некоторой теоремы в базе теорем (нажатием клавиши "Ctr - ц"). Прежде всего, реализуется цикл вывода следствий из данной теоремы, в процессе которого могут использоваться произвольные другие теоремы, расположенные в базе теорем. Поиск таких дополнительных теорем выполняется либо специальными справочниками поиска теорем, либо путем непосредственного просмотра нужных раздлеов оглавления базы теорем.

В процессе вывода новые теоремы получают некоторые характеристики, используемые для определения того, какие гипотетически полезные приемы могли бы быть созданы по теореме. По завершении цикла вывода процедура спецификатора, рассматривая характеристики новых теорем, выдает списки спецификаций таких приемов.

По спецификации приема запускается компилятор спецификаций, дающий описание приема на ГЕНОЛОГе. Здесь подключается описанная выше процедура тестирования новых приемов. Если прием успешно прошел тестирование, то он компилируется и регистрируется в буфере базы приемов. Одновременно та тестовая задача, которая для него была создана и для решения которой прием оказался необходим, регистрируется в буфере задачника, а та теорема, которая была получена в цикле логического вывода и послужила источником приема, регистрируется в буфере базы теорем. Если теорема уже имелась в базе теорем, она в буфере не регистрируется.

Заметим, что иногда цикл вывода теорем дает сотни новых теорем, а спецификатор по каждой из них предлагает несколько (иногда - десяток) спецификаций приемов. В большинстве случаев эти приемы (и даже породившие их теоремы) совершенно бесполезны. Тестирование оказалось очень сильным фильтром, после которого обычно остается не более двух-трех десятков новых приемов. Впрочем, в отдельных случаях наблюдалось появление свыше сотни вполне разумных приемов, прошедших не только тестирование, но и последующую доводку.

Работа доводчика начинается после того, как цикл тестирования завершен и результаты его зарегистрированы в буферах оглавлений. Целью доводчика является проверка того, не портят ли новые приемы умения системы решать "старые" задачи задачника, и коррекция приемов в тех случаях, когда они это делают. Результаты доводки регистрируются в разделе "Архив" оглавления базы приемов. Они распределе-

ны по нескольким подразделам, в зависимости от того, удалось ли убрать конфликты нового приема со старыми или нет. Окончательная регистраци "удачных" приемов в основных разделах решателя пока выполняется вручную, обычными средствами интерфейса оглавления базы приемов.

17.1 Исходные данные доводчика

Как уже говорилось, перед началом доводки в буферах оглавлений оказываются зарегистрированы откомпилированные приемы, теоремы и задачи. Именно, в разделе "Генератор приемов" буфера оглавления базы приемов ГЕНОЛОГа созданы подразделы, в каждом из которых хранится серия приемов. Число приемов в одной серии не более 50. Аналогичным образом в буферах своих оглавлений хранятся теоремы и задачи.

Из узла каждого нового приема имеются переходы "команда" (к терминалу, хранящему описание приема), "примечание" (к терминалу, хранящему спецификацию приема), "прием" (к терминалу, хранящему логический символ ЛОС- программы приемы и несколько первых ее операторов), "теорема" (к терминалу, хранящему ссылку "теорема(A1 A2)" на теорему, являющуюся источником приема (новую либо старую), "оглавление" (к терминалу, хранящему путь в оглавлении базы приемов к пункту буфера, хранящему ссылку на прием).

Из узла каждой новой теоремы имеются переходы "терм" (к терминлу, хранящему теорему), "комментарии" (к терминалу, хранящему характеристики теоремы), "приемы" (к терминалу, хранящему ссылки "прием(...)" на приемы, источником которых служит теорема, "смвывод" (к терминалу, хранящему путь в оглавлении базы теорем к той ячейке логического вывода, в которой был запущен цикл вывода теорем), "оглавление" (к терминалу, хранящему путь в оглавлении базы теорем к пункту буфера, хранящему ссылку на прием).

Каждая новая задача оформлена в задачнике обычным образом - указаны ее посылки, условия, цели и ответ.

17.2 Действия, предшествующие доводке

Перед началом доводки в файлах сохранена некоторая вспомогательная информация. Чтобы описать ее, проследим действия системы с момента нажатия клавиши "Ctr-ц" при просмотре теоремы.

Организация базы теорем будет рассмотрена лишь в послоедующих разделах. Заметим лишь, что в оглавлении теорем выделены подразделы, называемые ячейками логического вывода. Первый пункт такой ячейки содержит одну либо несколько "стартовых" теорем, из которых все остальные теоремы ячейки должны выводиться системой при помощи ее процедуры логического вывода. Поэтому клавиша "Ctr-ц" должна нажиматься только на одной из стартовых теорем ячейки. Сама процедура вывода теорем представляет собой решатель, насчитывающий очень большое число приемов. Общая его организация и несколько примеров приемов будут приведены ниже. Более полное описание приемов вывода теорем можно найти в 9 томе монографии "Компьютерное моделирование логических процессов".

Размещение описания доводчика в данном раделе объясняется тем, что он завершает цикл действий генератора приемов, выполняемых уже после перехода из базы теорем в базу приемов: компиляция спецификации приема и получение приема ГЕНОЛОГа, компиляция приема ГЕНОЛОГа в ЛОС-программу, тестирование приема и его доводка. Следующие разделы будут посвящены действиям, предшествующим указанному переходу. Работа доводчика почти не требует понимания этих действий, а в тех редких случаях, когда оно понадобится, будут даваться необходимые пояснения.

Выйти на ту точку программы, в которой начинается обработка нажатия "Ctr-ц", можно через пункт "Синтез приемов" - "Программирующий вывод и синтез приемов" - "Исходная точка" оглавления программ.

Прежде всего, счетчик шагов интерпретатора ЛОСа устанавливается на единицу и удаляется ряд ненужных комментариев к посылкам исходной задачи. Кроме того, создается структура данных диалогового блока для отображения числа выведенных теорем и прорисовывается бланк этого блока. Она сохраняется в комментарии к посылкам исходной задачи, так что при получении каждой новой теоремы число в ней увеличивается на единицу и перерисовывается на экране.

Предпринимается обращение к процедуре "прогрвывод", которой сообщается ссылка на тот концевой логический терминал T оглавления базы теорем, который ссылается на стартовые теоремы. Эта процедура выполняет логический вывод и присваивает переменной x6 список троек (новая теорема - список ее характеристик - блок вывода) для полученных результатов. Блок вывода - набор информационных элементов, указывающих способ получения теоремы и ранее предпринимавшиеся шаги вывода следствий из нее. Эти элементы описываются ниже в соответствующем разделе. По завершении вывода теорем происходит удаление комментария к посылкам исходной задачи, содержавшего структуру данных диалогового блока для отображения числа выведенных теорем.

В 6-м информационном блоке (база теорем) от корневого указателя-каталога по меткам "метка", "смвывод" создается переход к логическому терминалу, в который заносится путь в оглавлении базы теорем к текущей ячейке вывода - тому подразделу, первым пунктом которого является указанный выше логический терминал T.

Создается расширенный список x7 троек (теорема - список ее характеристик - блок вывода), получаемый из списка x6 добавлением в начале списка данных о стартовых теоремах. В списке x6 эти теоремы не регистрировались. Для каждой стартовой теоремы в качестве ее блока вывода берется пара (характ K), (начало A). Здесь K - список характеристик теоремы, A - терм "теорема(A_1, A_2)", являющийся ссылкой на теорему в базе теорем. Именно, A_1 - логический символ, к которому относится узел теоремы, A_2 - номер узла. Путь к этому узлу от корневого каталога 6-го информационного блока проходит по метке A_1, и далее - по цифрам номера A_2.

Данные списка x7 будут сохранены в 6-м информационном блоке в логических терминалах, достижимых из его корня по меткам "i", "новый", где i - последовательные логические символы, начиная с 1. Перед сохранением предпринимается удаление старых таких терминалов (просматриваются все номера i, начиная с 1, до того момента, как отсутствует переход по метке "новый"). Затем элементы списка x7 просматриваются в обратном порядке, и начиная с $i = 1$ для них создаются терминалы. Каждый терминал хранит теорему и идущие после нее характеристики. Отбрасываются характеристики, для которых в блоке вывода указано, что по ним теорема была обобщена. Сам блок вывода не регистрируется.

Создание приемов по теоремам, зарегистрированным в терминалах "новый", будет происходить с автоматическими перезапусками системы после обработки каждой теоремы. Режим работы с автоматическими перезапусками был введен, чтобы различные сбои, иногда возникающие в работе системы и пока не поддающиеся отладке из-за своего нерегулярного характера, не нарушали целостности вычислений. При автоматическом перезапуске тот цикл, на котором произошел сбой, полностью повторяется, но уже без сбоя (повторяемость сбоя делает возможной отладку, что обычно и происходило).

Так как весь цикл синтеза приемов достаточно масштабный и использует несколько различных распараллеленных режимов работы, в том числе для прокрутки по задачнику, то иногда, несмотря на множество уже выявленных и исправленных ошибок интерпретатора, сбои все же возникают. Обычно это происходит в режиме распараллеленной между многими ядрами процессора работы программы, где отладка затруднена. Поэтому, в отсутствии средств автоматического возобновления отдельных потоков работа генератора приемов вообще была бы невозможной. Чтобы справиться с этим, пока применяется скрипт fenix, автоматически перезапускающий поток, как только операционная системы его снимает. Тогда весь процесс от начала до конца проходит успешно, и работа генератора приемов выглядит вполне устойчивой.

Чтобы при автоматическом перезапуске работа генератора приемов возобновлялась, инициируются некоторые вспомогательные "регистры", роль которых играют различные логические терминалы в 6-м информационном блоке.

В логический терминал, достижимый из корня 6-го информационного блока по меткам "прием", "теорема", заносится 1. Этот терминал будет хранить символьный номер текущей теоремы из отрезка ссылок "новый", для которой будут создаваться приемы.

В логический терминал, достижимый по меткам "прием", "теоремы", заносится 0. Он будет служить индикатором успешной обработки теоремы. Если в процессе обработки произойдет сбой и система автоматически перезапустится, то наличие нуля будет сигналом о необходимости повторной обработки теоремы. Автоматический перезапуск будет выполняться и при успешном завершении обработки теоремы, но тогда в указанном терминале окажется 1.

В логический терминал, достижимый по меткам "прием", "новыйприем", заносится 0. Здесь будет сохраняться десятичная запись числа созданных приемов.

Затем выполняется оператор "трассировка(0)", выполняющий внутренний перезапуск системы. При этом система продолжит работу "с нуля", за исключением того, что в зоне программ сохранятся считанные туда фрагменты программ ЛОСа.

Дальнейшие действия - шаг обработки очередной теоремы - выполняются программой "вход" после контрольной точки "прием(115)". Выйти на эту точку можно через пункт "Синтез приемов", "Программирующий вывод и синтез приемов", "Начало шага обработки очередной теоремы" оглавления программ.

До этого программа "вход", инициируемая при перезапуске, усмотрела наличие перехода из корня 6-го информационного блока по метке "прием" к некоторому указателю U и отсутствие дальнейшего перехода по метке "число", наличие которого указывало бы на работу доводчика. Этот переход появится позднее, а пока переменной x8 присваивается содержимое логического терминала, достижимого из U по метке "теорема". Он хранит номер x14 текущей обрабатываемой теоремы.

Создается комментарий "доводка" к посылкам исходной задачи. Рассматриваются логические терминалы, достижимые из указателя U по меткам "теоремы" и "новыйприем". Переменной x10 присваивается содержимое первого из них, переменной x13 - число ранее созданных приемов, хранящееся во втором. Если x10 - единица (т.е. на предыдущем цикле не было сбоя), то x14 увеличивается на единицу и сразу перезаписывается в терминал "теорема". При этом в терминал "теоремы" заносится ноль - для контроля следующего цикла. Если x10 - ноль (был сбой), то x14 не изменяется, и действия повторяются.

Из логического терминала "x14" - "новый" считывается набор, образованный теоремой и ее характеристиками. Он присваивается переменной x17. На экране прорисовывается в соответствующем блоке диалога D число x13. Создается комментарий (новприем x13 D) к посылкам исходной задачи, который будет играть роль счетчика числа новых приемов. Набор x17 разбивается на теорему x18 и список ее характеристик x19. Процедура "приемы" генерирует по входным данным x18, x19 список x20 пар (теорема приема - спецификация приема). Эта процедура (она называется спецификатором) описывается в одном из последующих разделов. Отменяются все блокировки приемов, и начинается просмотр элементов x21 списка x20. Переменной x22 присваивается теорема приема, переменной x23 - его спецификация, переменной x25 - тип приема. Затем реализуется обращение к справочнику "задачи" на логическом символе x25. Прочими входными данными обращения служат теорема приема x22, спецификация x23 и пара (x18, x19). Справочник "задачи" был описан в предыдущей главе, посвященной тестированию приемов. Если прием прошел тестирование, то процедура "регприем" обеспечивает его компиляцию в программу ЛОСа, а также регистрацию описания приема на ГЕНОЛОГе в буфере базы приемов. Одновременно в буфере базы теорем регистрируется (если она новая) теорема x18, а в буфере задачника - тестовая задача.

По окончании просмотра списка x20 из комментария (новприем ...) к посылкам исходной задачи извлекается новое значение количества созданных приемов, которое перезаписывается в логический терминал "новыйприем". Кроме того, в терминал "теоремы" записывается единица, свидетельствующая об успешном завершении обработки теоремы. Сразу после этого выполняется внутренний перезапуск системы - переход к обработке очередной теоремы.

По исчерпании списка теорем, о чем свидетельствует отсутствие логического терминала "x12" - "новый", начинается работа доводчика. Прежде всего, предпринимается анализ избыточности созданных приемов для решения тестовых задач. Некоторые из них, прошедшие тестирование как неизбыточные, могут оказаться избыточными после добавления других приемов. Такой анализ осуществляется процедурой "Расчистка". Обращение к процедуре имеет вид "Расчистка(x1)", где x1 - фиктивная входная переменная (фактически - 0). Процедура предпринимает удаление ЛОС-программ тех приемов буфера базы приемов, которые оказываются избыточными для решения задач буфера задачника. Описания приемов на ГЕНОЛОГе сохраняются. Выйти на начало программы "Расчистка" можно через пункт "Синтез приемов" - "Синтез приемов с генерацией задач для тестирования их неизбыточности" - "Процедура "Расчистка" " оглавления программ.

Описание действий процедуры "Расчистка" приводить здесь не будем. Его можно найти в 7 томе монографии "Компьютерное моделирование логических процессов".

17.3 Продолжение предобработки после обращения к процедуре "Расчистка"

После того, как предпринята расчистка, ЛОС-программы избыточных приемов оказываются удалены, а из оставшихся - созданы ссылки через терминалы "задачи" на те тестовые задачи, в которых данный прием срабатывает.

Далее удаляются логические терминалы, достижимые из корневого каталога базы теорем по меткам "прием", "теорема"; "прием", "теоремы" и "прием", "новыйприем". Предпринимается также удаление терминалов, достижимых из корневого каталога по меткам "i", "новый". В них хранились данные о теоремах, по которым создавались приемы. Эти данные уже использованы при регистрации теорем в буфере базы теорем, причем из узлов новых приемов созданы ссылки на теоремы - источники приемов.

Если имелся терминал, достижимый из корневого каталога базы теорем по меткам "прием", "результат", то он удаляется. Такой терминал создается доводчиком (см. ниже), и перед началом доводки должен быть удален. В логический терминал, достижимый из корневого каталога по меткам "метка", "выполнить", заносится логический символ "3". Он указывает номер того действия, которое система должна будет выполнить после внутреннего перезапуска. Наконец, оператор "трассировка(0)" выполняет внутренний перезапуск.

После перезапуска система сразу проверяет наличие терминала "метка", "выполнить" и считывает хранящийся в нем логический символ n. Этот терминал удаляется, и создается комментарий (выполнить n) к посылкам исходной задачи. Далее система выполняет стартовую прорисовку главного меню, обращается к оператору "главноеменю", обеспечивающему стартовый диалог, а внутри этого оператора - к оператору "автоклавиатура". Последний реагирует на комментарий (выполнить n), удаляя его и обращаясь к справочнику "выполнить" на символе n. После этого в комментарии (автоклавиатура ...) к посылкам исходной задачи оказывается создана последовательность нажатий клавиш, обрабатываемых оператором "автоклавиатура" для продолжения действий. Справочник "выполнить" создает также ряд структур данных, уточняющих эти действия.

В нашем случае $n = 3$. Это означает необходимость запуска предварительной прокрутки по задачнику для анализа приемов, созданных в буфере базы приемов. Выйти на начало программы справочника можно через пункт "Общий интерфейс" - "Справочник ВЫПОЛНИТЬ" - "Запуск предварительной прокрутки для анализа приемов, созданных в буфере базы приемов. Предполагается, что в буферах базы теорем и задачника имеются сопровождающие теоремы и задачи".

Прежде всего, из логического терминала "метка" - "смвывод" базы теорем извлекается путь в оглавлении базы теорем, ведущий к той ячейке вывода, в которой предпринимался вывод. По этому пути определяется тот максимальный раздел R, к которому относится ячейка. В оглавлении задачника находится та корневая ветвь, которая соответствует разделу R. Для номера n пункта корневого меню оглавления задачника, с которого начинается данная ветвь, создается комментарий (отрезокзадач (n) (n)) к посылкам исходной задачи. Такой комментарий указывает пути к начальной и конечной точкам прокрутки по задачнику. Далее вводится комментарий (автоклавиатура (подборнеизвестных арктангенс)) к посылкам исходной задачи, и

работа справочника завершается. Логический символ "подборнеизвестных" соответствует клавише "з" и обеспечивает переход из главного меню в оглавление задачника. Логический символ "арктангенс" соответствует клавише "Ctr-д". Он инициирует работу доводчика. Точнее, лишь первый этап этой работы - прокрутку по выбранному разделу задачника.

17.4 Процедура "Примерка"

После того, как оператор "автоклавиатура" перевел систему в оглавление задачника и сымитировал нажатие клавиши "Ctr-д", происходит обращение к процедуре "Примерка". Точка программы, где расположено это обращение, достижима через пункт оглавления программ "Синтез приемов" - "Доводка новых приемов, хранящихся в буфере базы приемов" - "Примерка созданных приемов на выбранных разделах задачника" - "Запуск прокрутки по заданному разделу задачника для оценки поведения доводимых приемов". Так как процедура "Прокрутка" завершает свою работу внутренним перезапуском системы, дальнейшие операторы фрагмента программы, собержащего обращение к ней, несущественны.

Обращение к процедуре имеет вид "Примерка(x1)", причем входная переменная x1 - фиктивная (в нашем случае равна 0). Процедура "Примерка" инициирует распараллеленную прокрутку отрезка задачника, выделенного комментарием (отрезокзадач ...) к посылкам исходной задачи. Эта прокрутка выполняется копиями системы, хранящимися в поддиректориях $EX1, EX2, \ldots EXn$, а также главным экземпляром системы. Число потоков $n+1$ определяется возможностями процессора. Оно устанавливается вручную через интерфейс главного меню. Итоги прокрутки не регистрируются обычным образом в терминалах, связанных с узлами задач, а сохраняются только в разделе "Примерка" буфера оглавления задачника. Сюда заносятся ссылки на задачи, где имело место либо существенное замедление, либо существенное ускорение, либо изменение ответа, либо отказ. Каждый концевой пункт раздела обычным образом ссылается на задачу задачника. Если в процессе примерки возникло более 70 пунктов, то остальные в раздел не заносятся. Кроме ссылки на задачу, концевой пункт оглавления хранит следующие термы:

1. "число(A)" - A есть число шагов работы интерпретатора, затраченных на решение задачи.
2. "ответ(A)" - если имело место изменение ответа, то A - новый ответ. Особо выделяется случай, когда работа программы была оборвана внешним образом (остановлена системой). Тогда A - символ "обрыв".
3. "отказ" - если на задачу был получен отказ.
4. "прием(A_1 A_2 A_3)" - сылки на те новые приемы, которые сработали в задаче при ее решении.

В процессе примерки на решение задачи отводится число шагов работы интерпретатора, не более чем вдвое превышающее соответствующее число до создания новых приемов. Если за это время ответ не получен, регистрируется отказ.

Каждая из копий решателя сохраняет данные об особых случаях в логических терминалах, достижимых из корневого указателя-каталога задачника по путям вида

"s" - "смзадача". Здесь s - последовательные логические символы, начиная с символа "сброс". Сам логический терминал хранит ссылку "задача(A_1 A_2)" на задачу, а также те термы "число(...)", "ответ(...)", "отказ", "прием(...)", которые характеризуют решение задачи, как указано выше. По пути "прием" - "смзадача" из корневого каталога достижим терминал, хранящий первый логический символ s для регистрации очередного особого случая.

Выйти на начальную точку программы "Примерка" можно через пункт оглавления программ "Синтез приемов" - "Доводка новых приемов, хранящихся в буфере базы приемов" - "Примерка созданных приемов на выбранных разделах задачника" - "Процедура "Примерка" " - "Исходная точка. Создание списка ссылок на доводимые приемы в терминале задачника "задача" - "Примерка" ".

Подробное описание программы "Примерка" можно найти в 7 томе монографии "Компьютерное моделирование логических процессов". Здесь его не приводим.

17.5 Запуск доводчика по итогам примерки

В процессе доводки будет происходить коррекция приемов, направленная на восстановление процессов решения "старых" задач, выявленных при прокрутке, с сохранением решения новых задач. Так как коррекция связана с многочисленными примерками (повторными решениями задач), то она является достаточно трудоемкой. Поэтому, по умолчанию, доводка распараллелена: приемы делятся на группы, и в каждой группе доводка происходит раздельно, причем задачи везде одни и те же. По окончании результаты доводки объединяются, а чтобы устранить возможные искажения из-за раздельного рассмотрения приемов, предпринимается доработка их в последовательном режиме. Коррекция приема, в основном, сводится к изменению его уровня срабатывания. Иногда предпринимается вставка дополнительных фильтров либо изменение типа приема - переход к более "модтвированной" его версии согласно имеющимся в логическом ассемблере указаниям на такие версии. Как уже говорилось, доводка является двухэтапной: сначала происходит на одном разделе задачника, затем - по всему задачнику. Оба этапа обслуживаются одними и теми же программами.

Программы доводчика подробно изложены в 7 томе монографии "Компьютерное моделирование логических процессов". Здесь мы их не приводим.

По окончании доводки новые приемы будут размещены в разделе "Архив" корневого меню оглавления программ. В этом разделе заранее созданы следующие подменю:

1. Нейтральные приемы.

   Здесь помещаются те приемы, которые не привели ни к изменению ответов старых задач, ни к существенному их замедлению либо ускорению.

2. Ускорения.

   Здесь помещаются те приемы, которые не привели к изменению ответов старых задач или их существенному замедлению, но при этом заметно ускорили некоторые из них.

3. Равноценные либо лучшие ответы.

Здесь помещаются те приемы, которые не привели к существенному замедлению старых задач, но ответы некоторых из них изменили, не усложнив их вид.

4. Ухудшившиеся ответы.

   Здесь помещаются приемы, которые усложнили вид ответа некоторых старых задач.

5. Отказы.

   Здесь помещаются приемы, которые привели к получению отказов на некоторые старые задачи.

6. Большие замедления.

   Здесь помещаются приемы, существенно замедлившие решения некоторых старых задач.

7. Конфликты с отказами.

   Здесь помещаются приемы, попытки коррекции которых для преодоления отказов на старые задачи привели к отказам для новых задач.

8. Конфликты с замедлениями.

   Здесь помещаются приемы, попытки коррекции которых для преодоления больших замедлений в решении стархы задач привели к отказам для новых задач.

9. Зацикливания.

   Здесь помещаются приемы, из-за которых решение некоторых старых задач было прервано до получения ответа либо отказа.

После второго этапа доводки происходит переразмещение приемов, к этому моменту уже размещенных на первом этапе.

В корневом меню оглавлений задачника и базы теорем тоже имеется подменю "Архив", куда будут заноситься тестовые задачи и новые теоремы.

Все указанные подменю следует расчищать перед началом очередного цикла работы генератора приемов. Для этого служит клавиша "Ctr-я", нажимаемая из оглавления базы приемов.

Глава 18

Организация базы теорем

Логический ассемблер является следующим шагом в направлении к выявлению источников приемов после ГЕНОЛОГа. Он задает прием в виде теоремы, сопровожденной спецификацией. Однако, эта теорема в действительности не совсем теорема. Она является частью алгоритмического языка и поэтому допускает определенные искажения исходной "настоящей" теоремы. У нее могут быть отброшены антецеденты, проверка которых считается излишней ввиду стандартного контекста применения приема, добавлены какие-то чисто технические вставки, удобные для компиляции, и т.п. В ряде случаев теорема приема выглядит как что-то не только неверное, но даже бессмысленное - если не знать язык ГЕНОЛОГ. Чтобы понять, как теорема приема возникла, необходимо иметь в логической системе и базу "настоящих" теорем. Для каждого приема ГЕНОЛОГа в базе теорем должен быть указан его источник - теорема, из которой прием был получен.

Здесь мы переходим черту, отделяющую теоремы от приемов (или, иными словами, от алгоритмов). Выше логического ассемблера уже не будет никакого другого языка программирования - будут только база теорем и процедуры, обеспечивающие ее развитие и алгоритмизацию.

База теорем логической системы вначале возникала не путем логической формализации информации, извлеченной из учебников, а "обратным ходом" - от приемов к теоремам, являющимся их источниками. Попытки прямой формализации учебников приводили к теоремам, совершенно оторванным от приемов решателя. Лишь впоследствии, после того, как были изучены механизмы преобразования теорем в приемы, появилась возможность обучать решатель путем регистрации в его базе теорем фактов из учебников. Пока эта возможность может рассматриваться лишь как полуавтоматическое средство обучения, существенно помогающее учителю, но не отменяющее необходимости ручного синтеза и ручной коррекции приемов. Впрочем, начатая работа позволяет надеяться, что в итоге будет достигнута и полная автоматизация. Разумеется, при этом придется использовать эмпирическую адаптацию решателя к предметной области. Немного об этом будет рассказано ниже, хотя механизмы такой адаптации еще предстоит развивать.

18.1 Оглавление базы теорем

Вход в оглавление базы теорем возможен из главного меню системы по нажатию клавиши "б". Это оглавление напоминает оглавление базы приемов ГЕНОЛОГа и разбито на корневые разделы, соответствующие предметным областям, для которых

выполнялось обучение системы. Впрочем, некоторые из разделов в базе теорем пока не представлены. По мере углубления в подразделы, соответствие оглавления базы теорем оглавлению базы приемов нарушается. Теоремы сгруппированы по несколько другому принципу. Оказалось выгодным размещать в общем подразделе теоремы, которые получаются друг из друга простыми переходами, даже если они содержат понятия из совершенно различных областей.

Вначале размещение теорем по разделам было сравнительно произвольным, в целом придерживающимся традиционного разбиения на разделы и подразделы. Однако, при обучении системы логическому выводу, ориентированному на получение полезных для решателя теорем, оказалось целесообразным постепенно переупорядочить материал - разбить его на так называемые "ячейки логического вывода".

Ячейка логического вывода представляет собой подраздел оглавления, все пункты которого - концевые, причем в первом пункте представлены "исходные" теоремы данной ячейки, а в остальных пунктах располагаются теоремы, которые система должна самостоятельно "открывать", зная только исходные. Фактически, это задачник, используемый при обучении системы программирующему и исследовательскому логическому выводу. Первый из них преобразует базисные теоремы предметной области в теоремы приемов, сопровождая их необходимыми обобщающими параметрами или комбинируя с другими теоремами. Второй - просто находит новые важные факты в предметной области.

Чтобы определить, является ли подраздел ячейкой логического вывода, уже вовлеченной в процесс обучения системы, нужно выделить первый его пункт и нажать "Ctr-и". Пункт оглавления пропадет, а вместо него в верхней части экрана появится некоторая техническая информация о логическом выводе. Подраздел является ячейкой вывода, если эта информация непустая. Постепенно, по мере обучения системы, вся база теорем должна быть разбита на ячейки вывода.

В концевом пункте базы теорем зарегистрированы одна или несколько теорем. К их просмотру можно перейти, нажав в данном пункте клавишу "курсор вправо". Переход от одной теоремы пункта к другой выполняется клавишами "курсор вверх" - "курсор вниз". Для возвращения в оглавление используется "курсор влево".

Редактирование оглавления базы теорем выполняется стандартными средствами, описанными еще в первом томе монографии.

18.2 Связь теорем с приемами

Теорема представляется двум окнами, отделенными друг от друга горизонтальной чертой. В верхнем окне прорисовывается сама теорема - формульным либо текстовым редактором. Для смены режима прорисовки служат клавиши "ф" (формульный режим) и "т" (текстовый режим). В нижнем окне приводится список характеристик теоремы - логических символов либо термов, указывающих на различные свойства теоремы либо способа ее получения. Подробнее типы таких характеристик будут описаны в главе, посвященной характеризатору. Пока лишь заметим, что создание приемов по теореме будет происходить в процессе сканирования ее списка характеристик.

Имеется оглавление типов характеристик, для перехода к которому следует из просмотра теоремы нажать "Ctr-к"(кир.). Как и для любого справочного оглавления,

нажатие в нем "a" (кир.) приводит к просмотру всех логических символов, использованных в качестве заголовков характеристик, а нажатие "Ctr-a" - к просмотру всех неиспользованных символов. При переходе из концевого пункта оглавления типов характеристик по "курсор вправо" возникает экран, на котором в верхней части голубым цветом перерисован текст концевого пункта, а под ним текстовым редактором прорисована соответствующая характеристика. Ее можно редактировать: нажатие "Enter" переводит в текстовый редактор. Для сохранения новой (либо исходной) версии достаточно снова нажать "Enter".

Чтобы проследить происхождение приема ГЕНОЛОГа, он связывается с теоремой из базы теорем, называемой его источником. Для создания источника есть две возможности. Первая из них заключается в переходе от просмотра приема в базу теорем нажатием "Ctr-F9", выборе подходящего концевого пункта базы теорем либо создания нового концевого пункта, и копировании теоремы приема в этом пункте. Для этого нужно зайти в пункт и нажать "Й". Теорема приема окажется скопирована в базе теорем и зафиксирована как источник приема. При необходимости ее можно редактировать, сохраняя связь с приемом. Вторая возможность заключается в том, чтобы после перехода в базу теорем по "Ctr-F9" выбрать уже имеющуюся теорему и объявить ее источником приема. Для этого из просмотра теоремы нужно нажать "И". Вторая возможность обеспечивает и смену ранее выбранного источника. Теорема может быть источником нескольких приемов. Для просмотра их списка нажимается "п".

Если источник приема уже создан, для перехода к его просмотру достаточно нажать "Ctr-F9".

При автоматическом создании приемов по теоремам сразу регистрируется связь их с источниками.

Приведем несколько примеров, иллюстрирующих отличие теоремы приема от теоремы, находящейся в базе теорем. Теорема приема, преобразующего логарифм произведения в сумму логарифмов, имеет следующий вид:

$\forall_{abc}(0 \le a \ \& \ 0 \le b \to \log_c(ab) = \log_c a + \log_c b)$

Источником приема служит следующая теорема:

$\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ 0 < a \ \& \ \neg(a - 1 = 0) \ \& \ 0 < b \ \& \ 0 < c \to \log_c(ab) = \log_c a + \log_c b)$

Как видим, здесь добавлены все необходимые условия по о.д.з. для логарифмов. В теореме приема строгие неравенства заменены на нестрогие просто потому, что их априори проще проверять. Однако, так как прием применяетя в логически корректном контексте, эти нестрогие неравенства будут влечь строгие.

В некоторых случаях теорема приема получается из своего источника менее тривиальными преобразованиями, чем отбрасывание условий на о.д.з. Эти преобразования выполняются процедурой спецификатора, создающей спецификации приемов в процессе анализа характеристик теоремы. Могут вводиться переменные для функций, появляться описатели "класс" и т.п. Например, следующая теорема приема, исключающего описатель "класс":

$\forall_{ab}(\text{set}_c(a(c)) \subseteq b \leftrightarrow \forall_c(a(c) \to c \in b))$

- имеет своим источником определение включения множеств:

$\forall_{ab}(a - \text{set} \ \& \ b - \text{set} \to a \subseteq b \leftrightarrow \forall_c(c \in a \to c \in b))$

Помимо "настоящих" теорем, в базе теорем имеются термы другого рода. Они являются техническими характеристиками предметной области и решений, принятых относительно способов ее алгоритмизации и развития. Такие термы делятся на протоколы и квазипротоколы. По сути дела, отличие лишь в том, что протоколы набираются текстовым редактором и обычно не очень велики по размерам. Однако, если определяемая протоколом техническая информация слишком громоздка, ее чтение становится затруднительным. Тогда эта информация разделяется на две части. Одна из них вводится формульным редактором и внешне может быть похожа на теорему. Другая часть переносится в характеристики этой псевдотеоремы. Такая конструкция называется квазипротоколом.

В качестве простейшего примера можно привести протокол "нормализация(дробь нормдробь)", фиксирующий, что для логического символа "дробь" введен нормализатор общей стандартизации "нормдробь". Протокол инициализирует создание нормализатора.

Квазипротокол с теоремой

$(a + b)^n c/d + e$

и характеристиками "теоремаприема", "стандхаракт", "оператор(стандплюс)", "указатель(единица(1 x3 x4 x14)единица(0 x5))", "см(натуральное(x14) или(не(заголовок(x3 1)) не(заголовок(x4 1)) не(заголовок(x14 1))))" задает шаблон выражений, допускающих невырожденное преобразование к стандартной форме "стандплюс" (т.е. раскрывание скобок).

Иногда протокол играет роль установки на специальный цикл логического вывода. В этом случае он помещается в первый пункт ячейки логического вывода. Например, протокол "стандформа(станддн отр(1)кн(2)дн(2))" с характеристикой "число(20)" представляет собой установку на вывод первых 20 тождеств для упрощения дизъюнктивных нормальных форм. Цифры в скобках после логических операций указывают на их стоимость, относительно которой происходит упрощение.

Протоколы и квазипротоколы могут являться источниками теорем приемов. Например, протокол "развертка(суммавсех плюс)", означающий, что для двуместной коммутативно-ассоциативной операции "плюс" введено ее обобщение "суммавсех" на любое конечное число операндов, является источником целой серии теорем приемов, порождаемых спецификатором. Например, источником следующего приема, заносящего внешний член под знак операции над семейством:

$\forall_{abmn}(b = a(m-1) \to b + \sum_{i=m}^{n} a(i) = \sum_{i=m-1}^{n} a(i))$.

18.3 Интерфейс просмотра и редактирования теоремы

Приведем полный список возможностей интерфейса, связанного с теоремой, хотя многие из них прояснятся лишь в последующем изложении. Прежде всего, отметим действия мышью и курсорами. Левой кнопкой мыши можно выделять подтермы теоремы, при необходимости корректируя выделение клавишами курсора. Если после выделения подтерма нажать правую кнопку мыши, то под теоремой будет прорисовано пояснение к выделенному логическому символу. Если выделить левой кнопкой

мыши характеристику теоремы, то снизу появится текст, поясняющий смысл этой характеристики.

Заметим, что в одном концевом пункте оглавления базы теорем может быть "спрятано" несколько теорем. Переходы между ними осуществляются клавишами "курсор вверх" - "курсор вниз".

Если нужно перенести теорему в другой пункт оглавления, то на ней нажимается "Insert", затем ищется тот концевой пункт оглавления, куда эту теорему нужно перенести, и на нем (без перехода к просмотру его теорем) снова нажимается "Insert". Эти же действия выполняются, чтобы перенести теорему в начало списка того же самого пункта.

Заметим, что цвет линии под характеристиками теоремы указывает на способность системы выводить ее из исходных теорем ячейки вывода. Черный цвет означает, что пока система не была обучена выводу этой теоремы. Зеленый цвет - что она выводит теорему. Красный цвет - что вывод имелся, но утерян. Синий цвет - что он утерян только что.

Контекстное меню просмотра теоремы вызывается нажатием клавиши F1. В нем имеются следующие пункты:

1. Переход к справочнику по системе (повторное нажатие F1).
2. Ввод либо изменение теоремы формульным редактором (Ctr-ф).
3. Ввод либо изменение теоремы текстовым редактором (Ctr-т).
4. Ввод либо редактирование чертежа, сопровождающего теорему (ч).
5. Редактирование характеристик теоремы (к).
6. Автоматическое добавление антецедентов, необходимых для сопровождения по о.д.з. (Ctr-о).

   Данный пункт полезен при перенесении теоремы из базы приемов. Такое перенесение выполняется в два этапа. Сначала из просмотра приема нажимается "Ctr-F9", которое переводит в базу теорем. В базе теорем создается новый либо выбирается старый концевой пункт. После захода в этот пункт нажимается "Й", и теорема приема копируется в базе теорем. Однако, в ней отброшены антецеденты, обеспечивающие сопровождаение по о.д.з. Нажатие "Ctr-о" восстанавливает их. Иногда появляются избыточные антецеденты, которые далее исключаются вручную.

7. Создание копии теоремы (Ctr-д).

   При копировании теоремы в список ее характеристик добавляется ссылка "копия(s, N)" на исходную теорему. Здесь s - логический символ, N - номер узла этого символа, в котором хранится теорема (см. ниже структуры данных базы теорем).

   Копирование теоремы обычно служит для того, чтобы какое-то следствие одной ячейки логического вывода сделать исходной теоремой другой ячейки. Ограничители системы вывода часто вводят блокировку действий в зависимости от цепочки предыдущих выведенных теорем, а иногда и просто блокировку слишком длинных выводов. Создание новоя ячейки для получения следствий теоремы позволяет снять эти ограничения.

8. Формульный режим просмотра теоремы (ф).

9. Текстовый режим просмотра теоремы (т).

10. Вход в просмотр подтермов теоремы (Ctr-курсор вправо).

    Команды, связанные с просмотром подтермов, приводятся ниже.

11. Занесение теоремы в буфер для перенесения ее в другой раздел оглавления (Insert).

    Чтобы фактически ее перенести, нужно выбрать концевой пункт оглавления и нажать на нем "Insert".

12. Удаление теоремы (Ctr-Del).

13. Удаление чертежа теоремы (Ctr-ч).

14. Регистрация текущего чертежа в буфере (б).

15. Извлечение чертежа из буфера (Ч).

    Заметим, что при перенесении теоремы приема в базу теорем можно перенести и чертеж. Для этого в просмотре приема достаточно нажать "ч", сохранив его чертеж в буфере, а после извлечь из буфера данной командой.

16. Сброс буфера чертежа (Ctr-б).

17. Приемы, связанные с теоремой. Имеются следующие команды:

    (a) Просмотр списка приемов, источником которых служит теорема (п).

    Для смены приема в списке используются клавиши "курсор вверх - курсор вниз". Для возвращения к просмотру теоремы нажимается "курсор влево". Чтобы перейти от приема списка к его размещению в базе приемов, нажимается "Enter".

    (b) Выбор теоремы в качестве источника приема (И).

    Переход в базу теорем должен был состояться по "Ctr-F9" из просмотра приема.

    (c) Регистрация теоремы приема в базе теорем (Й).

    Аналогично предыдущему.

    (d) Создание спецификаций приемов по теореме (г).

    Если хотя бы одна спецификация создана, то она появляется на экране. Переходы между спецификациями - "курсор вверх - курсор вниз". Если для текущей спецификации нажать "б", то в буфере базы приемов по ней будет создан и откомпилирован прием ГЕНОЛОГа. При желании его можно будет перенести в любой раздел базы приемов. Если этого не сделать, то при расчистке буфера прием пропадет.

    (e) Трассировка создания спецификаций с выбором точки прерывания по оглавлению программ (Ctr-г).

    Используется при отладке и развитии спецификатора. После нажатия клавиши на экране появляется оглавление программ, в котором нужно найти подходящий концевой пункт и нажать "курсор вправо".

(f) Трассировка моментов создания спецификаций (Г).

При каждом обращении к оператору "ктс" (контрольная точка спецификатора) будет происходить выход в отладчик ЛОСа.

(g) Синтез новых приемов по теореме и регистрация их в буфере базы приемов (с).

Сначала работает спецификатор, затем по всем новым спецификациям создаются приемы.

18. Логический вывод, связанный с теоремой.

(a) Характеризация теоремы - исходная либо повторная (Х - кир.).

Если теорема снабжена одной из характеристик "пв", "теоремаприема", "блок", "протокол", "тожд", "станд", то данная команда блокируется. Это делается в тех случаях, когда характеризатор не компетентен изменять характеристику. Например, если теорема была выведена программирующим выводом для специальной цели и им же охарактеризована. Собственно, характеристика "пв" (сокращение от "программирующий вывод"), сопровождающая все теоремы ячеек вывода, кроме исходных, и указывает на такую ситуацию.

При характеризации все старые характеристики удаляются, а вместо них вводятся новые.

(b) Тестирование характеризации по шагам вывода характеристик (Ctr-x; кир.).

После нажатия "Ctr-x" происходит выход в отладчик ЛОСа при получении первой характеристики. Она оказывается значением переменной x1 процедуры "характ", регистрирующей очередную характеристику в накопителе характеристик. К процедуре "характ" обращается внешняя процедура "характеризатор", и можно через отладчик ЛОСа посмотреть, как возникла текущая характеристика. Чтобы перейти к просмотру следующей характеристики, нажимаются "0" и "Enter". Для обрыва просмотра можно нажать Esc.

(c) Запуск программирующего логического вывода (л).

Команда работает только на теоремах, расположенных в первом пункте какой-либо ячейки логического вывода. Процесс вывода на экране не отображается. По завершении его выдается список полученных теорем. Чтобы просмотреть подробности вывода какой-либо теоремы, нужно на ней нажать "курсор вправо". Далее можно либо просмотреть всю цепочку вывода этой теоремы из исходных теорем ячейки вывода, либо перейти в отладчик ЛОСа на момент получения теоремы. Последнее достигается путем повторного запуска процедуры вывода (внутри интерфейса просмотра результатов вывода) и занимает некоторое время.

(d) Запуск программирующего вывода с регистрацией в архиве хода получения теорем данной ячейки вывода (Л).

Аналогично предыдущему, но в архиве базы теорем сохраняется полная информация о дереве вывода каждой из теорем ячейки вывода, для которых система смогла получить вывод. Просмотр этой информации - см. следующий пункт.

(e) Просмотр вывода теоремы по данным архива (д).

Клавиша "д"нажимается из просмотра теоремы. Появляется вершина дерева вывода. Для просмотра других вершин используются клавиши курсора. Возможен повторный запуск вывода текущей теоремы с выходом в отладчик ЛОСа либо на начальном этапе работы приема вывода, либо на завершающем этапе. Фактически снова запускается полная процедура вывода в ячейке, но посторонние выводы отсекаются, и данный вывод осуществляется быстро.

(f) Запуск сокращенного программирующего логического вывода по текущей теореме (Ctr-л).

Используется для проверки того, что ранее зарегистрированные в архиве выводы теорем сохранились. Блокируются действия, связанные с получением других теорем.

(g) Просмотр хэша текущей теоремы (Ctr-F3).

Для быстрой проверки того, что вновь выведенная теорема уже имелась, все теоремы базы теорем хэшируются. В качестве хэша выступает некоторый логический символ. При нажатии на клавишу происходит выход в отладчик ЛОСа, где переменной x8 присвоен хэш теоремы.

(h) Запуск цикла вывода теорем, создания приемов и их доводки (Ctr-ц).

Фактически, это кнопка запуска генератора приемов по теореме. Запускается на теоремах из первого пункта ячеек вывода. Сначала предпринимается цикл программирующего вывода. На экране отображается число выведенных теорем. Затем предпринимается создание приемов по этим теоремам. Далее для каждого приема предпринимается попытка создания тестового примера, где этот прием мог бы оказаться необходимым. Отбираются только те приемы, без которых тестовый пример решателем не делается. Эти приемы компилируются. Выполняется их расчистка: если новые тестовые примеры делаются другими новыми приемами, то данный новый прием удаляется. Выбирается раздел задачника, к которому естественно отнести созданные приемы, и происходит распараллеленная прокрутка решателя по этому разделу. В процессе прокрутки отбираются те задачи, которые замедлились, либо сильно ускорились, либо перестали решаться, либо изменили ответ. По этим задачам предпринимается доводка приемов: варьируются их уровни срабатывания, корректируются фильтры, меняются типы. Доводка распараллелена: приемы разбиваются на примерно равные группы, и каждая группа доводится на всем списке отобранных задач независимо от других групп. По окончании прокрутки результаты объединяются, выполняется небольшая дополнительная доводка (уже в последовательном режиме), и выдается результат. Он распределен по разделам "Архив" корневых меню оглавлений базы приемов, теорем и задачника. В базе приемов раздел "Архив" имеет подразделы "Нейтральные приемы", "Ускорения", "Отказы", "Изменившиеся ответы", "Большие замедления", "Конфликты с отказами", "Конфликты с замедлениями", "зацикливания". Приемы из первых двух разделов оказываются откомпилированы, из других разделов - зарегистрированы, но не откомпилированы. Раздел "Архив" базы теорем сохраняет новые теоремы, на которых основаны созданные приемы, раздел "Архив" задачника - тестовые примеры, обосновывающие пользу от новых приемов.

(i) Установка режима доводчика (Ctr-p).
Для целей отладки можно отказаться от распараллеливания доводки. Иначе отследить действия одного из параллельных процессов сложно.

19. Просмотр ссылки на теорему в базе теорем (Ctr-й).
Такой ссылкой служит терм "теорема(s, N)", где s - логический символ, N - номер узла этого символа, являющегося узлом теоремы.

20. Переход от копии теоремы к оригиналу (о - кир.).

21. Просмотр оглавления типов характеристик теорем (Ctr - к; кир.).

22. Просмотр оглавления логического языка системы (Ctr-я).

23. Просмотр оглавления типов протоколов базы теорем (Ctr-а).

Знания интеллектуальной системы о реальном мире в значительной степени состоят из примеров логики устройства реальных объектов. Такого рода пример удобно представлять в виде кванторной импликации, у которой антецеденты характеризуют тип объекта, пример которого приводится, а консеквент имеет вид квантора существования, вводящего вспомогательные параметры, через которые выражается устройство объекта. Обычно эти кванторные импликации чрезвычайно громоздки, и для удобства работы с ними пришлось создасть специальный интерфейс. В качестве примера можно указать описание устройства прямоугольного стола с четырьмя ножками, которое расположено в разделе "Словарь" - "С" - "Стол" - "Схемы" - "Четырехугольный стол с четырьмя ножками". Здесь антецеденты располагаются после строки "если", а конъюнктивные члены консеквента - после строки "то". Квантор существования отброшен, так как его переменные - все новые переменные утверждений консеквента. Характеристики теоремы помещаются сверху от вертикальной черты над строкой "если". В нашем случае это характеристика "прим(x1)", указывающая, что приведен пример устройства стола x1. Подробнее о данном интерфейсе предполагается рассказать в последующих томах монографии.

18.4 Структуры данных базы теорем

База теорем находится в 6-м информационном блоке. Корневой указатель этого блока - каталог. Ссылка из него по логическому символу s осуществляется на указатель - список, являющийся корнем статьи символа s. Как и в базе приемов, используется дерево номеров узлов статьи логического символа. Концевые вершины этого дерева суть узлы теорем. Ссылка на узел теоремы - терм "теорема(s, N)", где N - номер узла.

Узел теоремы представляет собой указатель-список, из которого имеют место переходы по следующим меткам:

1. "терм" - переход к логическому терминалу, хранящему теорему.

2. "оглавление" - переход к логическому терминалу, хранящему путь в оглавлении базы теорем к концевому терминалу, хранящему основную ссылку на теорему. Допускается (хотя практически не используется) наличие дополнительных ссылок на ту же самую теорему из других разделов оглавления базы теорем.

3. "примечание" - переход к текстовому терминалу, хранящему примечания к теореме. Заполняется при необходимости вручную. Практически не используется.

4. "комментарии" - переход к логическому терминалу, хранящему характеристики теоремы.

5. "приемы" - переход к логическому терминалу, хранящему ссылки "прием(A_1, A_2, A_3)" на те приемы, для которых данная теорема является источником. Из узла каждого такого приема по метке "теорема" - переход к логическому терминалу, хранящему обратную ссылку "теорема(B_1, B_2)" на узел теоремы.

6. "геомредактор" - переход к логическому терминалу, хранящему описание сопровождающего теорему чертежа.

7. "хэштеоремы" - переход к логическому терминалу, хранящему символьный хэш A данной теоремы. По меткам A, "хэштеоремы" от корневого каталога 6-го информационного блока - переход к логическому терминалу, хранящему ссылки "теорема(B_1, B_2)" на все теоремы, имеющие хэш A.

8. "пв" - переход к логическому терминалу, хранящему ссылку "теорема(A_1, A_2)" на ту исходную теорему ячейки логического вывода, из которой данная была выведена программирующим логическим выводом. В этом же терминале сохраняется терм "приемы($B_1 \dots B_m$)", где $B_1, \dots, B_m$ - термы "прием(C_1, C_2)", являющиеся ссылками на последовательно применявшиеся приемы вывода. Если на последнем цикл тестирования вывод был утерян, то в терминал добавляется символ "минус". Если вывод теоремы был утерян ранее, чем на последнем цикле, то вместо символа "минус" используется символ "Минус".

Из корневого каталога базы теорем имеется также ряд переходов к информационным логическим терминалам, используемым процедурами логического вывода и синтеза приемов.

Оглавление базы теорем имеет своим именем символ "теорема". В концевом логическом терминале этого оглавления хранятся термы "теорема(B_1, B_2)", ссылающиеся на отнесенные к нему теоремы. Заголовок "теорема" одного из таких термов может быть заменен на символ "нормуравн" для указания на текущую теорему пункта. Просмотр теорем этого пункта при возвращении к нему будет начинаться с данной теоремы. Терм "замечание($C_1, \dots, C_n$)" в концевом терминале хранит набор информационных элементов $C_1, \dots, C_n$, характеризующих принцип группировки в нем теорем. По метке "замечание" из указателя оглавления - переход к логическому терминалу, хранящему элементы таких же типов и характеризующих данную ветвь оглавления. Эти замечания позволяют системе искать нужные теоремы через оглавление.

Используются следующие типы элементов C_i:

1. "раздел(A)" - раздел с названием A. Названия разделов - те же, что в базе приемов.

2. "логсимвол(A)" - подраздел, соответствующий логическому символу A.

Имеется архив базы теорем, сохраняющий информацию о выводе теорем. Для каждой исходной теоремы ячейки логического вывода в нем сохраняется дерево следствий теоремы, указывающее траектории вывода прочих теорем данной ячейки. Фактически, это означает сохранение доказательств таких теорем.

Архив был создан для ускоренного тестирования процедуры программирующего логического вывода. Он позволяет быстро убедиться в том, что выводы не были утеряны в процессе обучения системы. Ускорение достигается за счет того, что отбрасываются любые попытки применения приемов вывода, не указанные в дереве следствий.

Хотя организация программирующего вывода будет подробно изложена лишь в последущих главах, устройство архива базы теорем можно описать уже сейчас.

Архив базы теорем хранится в 15-м информационном блоке. Если узел стартовой (расположенной в первом пункте ячейки логического вывода) теоремы T относится к логическому символу S и имеет номер $N = N_1 \dots N_k$, где N_i - цифры, то в 15-м информационном блоке из корня статьи символа S по цепочке ребер, помеченных цифрами $N_1, \dots, N_k$, вдет путь к указателю - списку U, называемому узлом вывода данной теоремы.

От узла вывода по метке "равно" - переход к корню поддерева D, ребра которого помечены символьными числами (начиная с 1). Вершины этого поддерева соответствуют теоремам, выведенным из T процедурой "прогрвывод" - тем, которые зарегистрированы в ячейке логического вывода либо промежуточным теоремам, использованным при выводе последних. Дерево D называется деревом следствий теоремы T. Каждая вершина дерева D представляет собой указатель - список.

Помимо ребер, связывающих между собой вершины дерева следствий, из каждой вершины этого дерева выходят ребра со следующими отметками:

1. "вход" - ведет к логическому терминалу, хранящему теорему. Стартовая теорема тоже продублирована в таком терминале.

2. "прием" - ведет к логическому терминалу, указывающему прием программирующего вывода, примененный при получении теоремы. В терминале хранится четверка A_1, A_2, A_3, A_4, где A_1 - логический символ, к программе которого относится прием (фактически - инициировавшая его срабатывание характеристика исходной теоремы); A_2 - уровень срабатывания приема (символьное число); A_3 - номер контрольной точки, с которой начинается собственно программа приема. A_4 - заголовок набора B либо логический символ B, извлеченные из элемента (источник B) блока вывода. Такие символы B закрепляются за приемами программирующего вывода в качестве их локальных (относительно символа A_1) имен.

3. "комментарии" - ведет к логическому терминалу, хранящему список характеристик теоремы, созданных процедурой вывода.

4. "второйтерм" - ребро имеется, если прием вывода использовал дополнительные теоремы, и ведет к логическому терминалу, содержащему эти теоремы.

5. "корень" - ребро имеется, если вершина дерева следствий соответствует не промежуточному результату, а теореме из ячейки логического вывода. Оно ведет к логическому терминалу, хранящему ссылку "теорема(...)" на узел теоремы в базе теорем.

6. "указатель" - переход к логическому терминалу, фиксирующему успех либо неуспех ускоренного тестирования вывода данной теоремы. При успешном выводе содержит символ "плюс", иначе - символ "минус".

7. "параметры" - используется только для корня дерева следствий. Ведет к логическому терминалу, хранящему термы "число(A_1)" и "число(A_2)". A_1 - трудоемкость последнего запуска ускоренного тестирования, A_2 - величина приращения трудоемкости по сравнению с предыдущим запуском. Если ячейка вывода имеет несколько стартовых теорем, то содержимое данного терминала продублировано во всех корнях их деревьев следствий.

Глава 19

Характеристики теорем

Характеристики теорем нужны для их алгоритмизации. Одни из них констатируют самые общие свойства теоремы, другие - указывают на возможность создания по теореме приемов различных типов, третьи - фиксируют особенности этих приемов. Генератор приемов сканирует характеристики теоремы и запускает по текущей характеристике процедуру спецификатора, предлагающую различные версии спецификаций. Используются характеристики также при просмотре базы теорем процедурой логического вывода для отбора нужных утверждений.

Источники характеристик различны. Часть их типов создается процедурой характеризатора, которой сообщается только теорема, быть может, дополненная ранее имевшимися ее характеристиками. Другие характеристики вводятся процедурой логического вывода, которая знает, для чего была получена теорема, и регистрирует эту ее целевую направленность в виде характеристики. Наконец, многие типы характеристик на текущий момент вводятся вручную. Они были созданы лишь для того, чтобы спецификатор мог как-то связать источник приема с уже имеющимся приемом и представляют собой задания на дальнейшее обучение системы.

Перечислим некоторые наиболее часто встречающиеся типы характеристик. Более полное их перечисление можно найти в 8 томе монографии "Компьютерное моделирование логических процессов". Перейти к оглавлению типов характеристик теорем можно, войдя в просмотр любой теоремы из базы теорем и нажав клавишу "Ctr-к".

19.1 Общие характеристики

Определения

1. определение(t). Теорема представляет собой определение терма t. Пример:

   $\forall_n(n - \text{натуральное} \leftrightarrow n - \text{целое} \ \& \ 1 \leq n)$

   Характеристика - "определение(натуральное(n))".

   $\forall_a(a - \text{число} \ \& \ \neg(a = 0) \rightarrow \text{sg}(a) = (1 \text{ при } 0 < a, \text{ иначе } -1))$

   Характеристика - "определение(сигнум(a))".

   $\forall_a(a - \text{set} \ \& \ \text{огрснизу}(a) \ \& \ \neg(a = \emptyset) \ \& \ a \subseteq \mathbf{R} \rightarrow \text{наибольший}(\inf(a), \text{set}_b(\text{нижняягрань}(b, a))))$

   Характеристика - "определение(инф(a))".

2. опр(x1). Теорема представляет собой определение функции, обозначенной символом x1. Пример:

   обрфункция($\lambda_x(\sin x, x \in [-\pi/2, \pi/2])) = \lambda_x(\arcsin x, x \in [-1, 1])$

   Характеристика - "опр(арксинус)".

3. принадлежит(x1). Эквивалентность может рассматриваться как определение принадлежности множеству. x1 - направление замены. Пример:

   $\forall_{abf}(f - $функция $\& \; b - \text{set} \; \& \; b \subseteq \text{Dom}(f) \rightarrow a \in$ образ$(f, b) \leftrightarrow \exists_x(a = f(x) \; \& \; x \in b))$

   Характеристика - "принадлежит(второйтерм)".

Информация для создания приемов

1. см(x1). Синтезируемым приемам передаются фильтры списка x1. Характеристика определяется процедурой логического вывода, если она заранее знает, для какого приема создана теорема.

2. указатель(x1). Синтезируемым приемам передаются указатели списка x1. Характеристика определяется процедурой логического вывода, если она заранее знает, для какого приема создана теорема.

3. внутрпреобр(x1). Теорема предназначена для приема анализатора x1, выполняющего тождественное либо эквивалентное преобразование. Характеристика определяется процедурой логического вывода.

4. сокращгрупп(x1). Тождество, выведенное для создания приема справочника поиска теорем "Сокращение". x1 - направление замены. Пример:

   $\forall_{abc}(a -$ число $\& \; b -$ число $\& \; c -$ число $\& \; 0 \leq a \; \& \; 0 < b \; \& \; 0 < c \; \& \; b + c = 1 \rightarrow a^b a^c = a)$

   Теорема самостоятельной ценности не имеет, так как для перемножения степеней с одинаковым основанием есть другая теорема. Однако, она используется программирующим выводом в качестве подсказки на возможность исключения степенного выражения.

19.2 Тождества или эквивалентности

В данном подразделе собраны характеристики, которые могут относиться к заменам любого типа - тождественным или эквивалентным.

1. группировки. Теорема представляет собой тождество либо эквивалентность, обе части которых - элементарные бесповторные термы, имеющие одно и то же множество переменных. Пример:

   $\forall_{ab}(-a < -b \leftrightarrow b < a)$

2. вычпрог(x1 x2 x3). Теорема представляет собой тождество либо эквивалентность, которые в случае применения в направлении x1 обеспечивают упрощение относительно констант, использующее вычисления ГЕНОЛОГа. x2 - конъюнкция фильтров, уточняющих контекст стандартизации (в нее включаются

указания на типы константных значений переменных), x3 - список подвыражений заменяющего терма, обрабатываемых путем непосредственных вычислений. Пример:

$\forall_{abcdef}(a -$ число $\&$ $b -$ число $\&$ $c -$ число $\&$ $d -$ число $\&$ $e -$ число $\&$ $f -$ число $\& \neg(c = 0) \& neg(d = 0) \& \neg(f = 0) \to ab/(cd)+ae/(cf) = a(bf+de)/(cdf))$

Характеристика - "вычпрог(второйтерм и(десчисло(d)десчисло(e)десчисло(b) десчисло(f)), $bf + de$, df).

19.3 Тождества

Упрощающие тождества

1. нормализация(x1). Теорема представляет собой тождество, обеспечивающее общую стандартизацию. x1 - направление замены. Пример:

   $\forall_a(a -$ число $\to a + 0 = a)$

   Характеристика - "нормализация(второйтерм)".

2. Уменьшение оценки сложности.

   (a) упрощение(x1). Тождество, обеспечивающее переход к более простым в смысле справочника "оценка" термам. x1 - направление замены. Пример:

   $\forall_a(a -$ число $\to \cos(4 \operatorname{arctg} a) = (a^4 - 6a^2 + 1)/(1 + a^2)^2)$.

   (b) упрощкн(x1 x2). Тождество, обеспечивающее переход к более простым в смысле справочника "оценка"термам, при условии, что заданные выражения не зависят от заданных параметров. x1 - список термов "конст(A1 A2)", где A1 - переменная, обозначающая выражение, которое не должно зависеть от переменных списка A2. x2 - направление замены. Пример:

   $\forall_{ABijnp}($верпространство(B) $\&$ $A -$ слово $\&$ Val$(A) \subseteq$ события(B) $\&$ $l(A) = n$ $\&$ незавсобытия(A, B) $\& j \in \{0, \ldots, n\} \& \forall_k(k \in \{1, \ldots, n\} \to$ вероятность$(A(k), B) = p) \to$ вероятность(слойсемейства$(A,$ элемсобытия$(B), j), B) = C_n^j p^j (1 - p)^{n-j})$

   Характеристика - "упрощкн(конст(p k) второйтерм)".

   (c) варианты(x1). Тождество преобразует корневую сложную операцию в условное выражение с более простыми операциями. x1 - направление замены. Пример:

   $\forall_{ab}(a -$ число $\&$ $b -$ число $\to \min(a, b) = (a$ при $a \leq b,$ иначе $b))$

   Характеристика - "варианты(второйтерм)".

3. Уменьшение числа подтермов наибольшей сложности.

   (a) группмножитель(x1). Тождество выполняет группировку сложных операций. x1 - направление замены. Пример:

$\forall_{adf}(a - \text{set}\ \&\ d - \text{set}\ \&\ f - \text{функция} \to \text{прообраз}(f, a) \cup \text{прообраз}(f, d) = \text{прообраз}(f, a \cup d))$

Характеристика - "группмножитель(второйтерм)".

(b) единствсущ(x1). Тождество преобразует терм с несколькими вхождениями операции максимальной сложности, хотя бы одно из которых содержит все переменные заменяемой части, в выражение с единственным вхождением этой операции максимальной сложности. x1 - направление замены. Пример:

$\forall_{abcdeg}(a - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{число}\ \&\ e - \text{число}\ \&\ b - \text{комплексное}\ \&\ g - \text{комплексное} \to ab/(cg) + db/(eg) = (ae + cd)/(ce) \cdot b/g)$

Здесь уменьшается число комплексных дробей.

Характеристика - "единствсущ(второйтерм)".

(c) уменьшсложн(x1). Теорема одновременно уменьшает число выражений максимальной сложности и приводит к более простым (по числу переменных) таким выражениям. x1 - направление замены. Пример:

$\forall_{ade}(d - \text{число}\ \&\ e - \text{число}\ \&\ 0 \le e\ \&\ a - \text{число} \to a\sqrt{2\sqrt{e} + 2\sqrt{e + d^2}} = a\sqrt{d + \sqrt{e + d^2}} + a\sqrt{-d + \sqrt{e + d^2}})$

Характеристика - "уменьшсложн(первыйтерм)".

(d) сокращ(x1). Тождество упрощает выражение под корневой сложной операцией и не вводит новых операций большей либо равной сложности. x1 - направление замены. Пример:

$\forall_{afd}(a - \text{set}\ \&\ d - \text{set}\ \&\ f - \text{функция} \to \text{прообраз}(f, a) \cup \text{прообраз}(f, d) = \text{прообраз}(f, a \cup d))$

Характеристика: сокращ(первыйтерм).

4. Тождества с описателями.

(a) описатель(x1). Теорема представляет собой тождество, которое используется для перехода к более простым описателям либо для исключения описателей. x1 - направление замены. Пример:

$\forall_{Aa}(A - \text{set}\ \&\ a - \text{число}\ \&\ \text{конечное}(A) \to \sum_{x, x \in A} a = a\text{card}A)$

Характеристика - "описатель(второйтерм)".

(b) развязка(x1). Тождество выносит наружу операцию над семейством из-под сложного понятия. x1 - направление замены. Пример:

$\forall_c(c - \text{функция}\ \&\ \text{семействомножеств}(c)\ \&\ \text{конечное}(\text{Dom}(c))\ \&\ \text{разделимы}(c)\ \&\ \text{конечные}(\text{Val}(c)) \to \text{card} \bigcup(c) = \sum_{a, a \in \text{Dom}(c)} \text{card}c(a))$

Характеристика - "развязка(второйтерм)". Сложным понятием считается "мощность".

(c) парамописание(x1). Теорема представляет собой тождество, преобразующее параметрическое описание класса в его явное задание. x1 - направление замены. Пример:

$\forall_{abcd}(a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{число}\ \&\ \neg((a,c) = (0,0)) \to \text{set}_{xy}(\exists_t(x = at + b\ \&\ y\ ct + d\ \&\ t - \text{число})) = \text{set}_{xy}(cx - ay + ad - bc\ \&\ x - \text{число}\ \&\ y - \text{число}))$

Характеристика - "парамописание(второйтерм)".

5. внешзнак(x1). Тождество сохраняет сложность самого сложного подвыражения, но уменьшает сложность его надвыражений.

   Пример:

   $\forall_{abf}(\neg(a - 1 = 0)\ \&\ \neg(b - 1 = 0)\ \&\ 0 < a\ \&\ 9 < b\ \&\ a - \text{число}\ \&\ b - \text{число}\ \&\ f - \text{число} \to f \log_b a = f / \log_a b)$

   Характеристика - "внешзнак(первыйтерм)".

Декомпозирующие тождества

1. декомпозиция(x1). Тождество для декомпозиции выражения со сложным заголовком. x1 - направление замены. Пример:

   $\forall_{ab}(a - \text{число}\ \&\ b - \text{число} \to \cos(a + b) = \cos a \cos b - \sin a \sin b)$

   Характеристика - "декомпозиция(второйтерм)".

2. частичнпредст(x1). Тождество для декомпозиции сложного выражения, использующее равенство в посылках для значения самого сложного подвыражения одного из декомпозирующих термов. x1 - направление замены. Пример:

   $\forall_{abcfp}(c = a \setminus b\ \&\ \text{card}(\text{roots}(f,b)) = p\ \&\ b \subseteq a \to \text{card}(\text{roots}(f,a)) = p + \text{card}(\text{roots}(f,c)))$

   Прием используется в задачах на определение числа корней. Схема решения их такова, что выводятся посылки, определяющие число корней на отдельных промежутках, а данный прием постепенно сужает область, для которой число корней еще не найдено. Имеется другая версия приема, у которой в посылках находится равенство не для $\text{card}(\text{roots}(f,b))$, а для $\text{roots}(f,b))$.

   Характеристика - "частичнпредст(второйтерм)".

Константные подвыражения

Теоремы с характеристиками, перечисляемыми в данном разделе, обычно создаются ради приемов заранее известного типа. По существу, это скорее схемы вычислений, чем теоремы. Их можно считать квазипротоколами.

1. константы(x1 x2 x3). Тождество обеспечивает стандартизацию константных подтермов неизвестных выражений. x1 - направление замены, x2 - набор термов

"типданных($A\ x_1 \ldots\ x_n$)", уточняющих тип A константных значений переменных $x_1, \ldots, x_n$, x3 - терм "неизвестные($y_1\ \ldots\ y_m$)", перечисляющий переменные для неизвестных выражений. Пример:

$\forall_{abcde}(b = ac \to a^d b^e = a^{d+e} c^e)$

Характеристика - "константы(второйтерм типданных(натуральное a b) неизвестные(d e))".

2. нормконст(x1 x2). Тождество обеспечивает группировку константных подвыражений для упрощения с помощью нормализаторов общей стандартизации. x1 - направление замены, x2 - конъюнкция фильтров. Пример:

$\forall_{abcdefgmn}(a - \text{число}\ \&\ b - \text{число}\ \&c - \text{число}\ \&\ d - \text{число}\ \&\ e - \text{число}\ \&\ f - \text{число}\ \&\ g - \text{число}\ \&\ m - \text{число}\ \&\ n - \text{число}\ \&\ 0 \le a\ \&\ 0 < b\ \&\ \neg(d = 0) \to ea^{cn+f}/(db^{cm+g}) = e(a^n/b^m)^c \cdot a^f/(db^g))$

Характеристика - "нормконст(второйтерм и(натуральное(x1) натуральное(x2) не(равно(нод(x1 x2)1)) целое(x13)целое(x14)не(константа(x3))константа(x6)константа(x7)))". Группировка позволяет сократить a и b.

Преобразование описателей

1. Операции над семействами.

   (a) упрощплс. Тождество, позволяющее перейти к более простой операции над семейством. x1 - направление замены. Пример:

   $\forall_{Paf}(P - \text{set}\ \&\ a - \text{число}\ \&\ f - \text{функция}\ \&\ \forall_i(i \in P \to 0 < f(i))\ \&\ P \subseteq \mathrm{Dom}(f)\ \&\ \neg(a - 1 = 0)\ \&\ 0 < a \to \log_a \prod_{i,i\in P} f(i) = \sum_{i,i\in P} \log_a f(i))$.

   (b) развернуть(x1). Тождество исключает операцию над семейством при развертке описателей в заменяющей части. x1 - направление замены. Пример:

   $\forall_{abm}(b - \text{целое}\ \&\ k - \text{целое}\ \&\ m - \text{целое}\ \&\ 0 \le b\ \&\ k \le m \to \sum_{n=b}^{k} C_m^n = (2^m - \sum_{n=0}^{b-1} C_m^n - \sum_{n=k+1}^{m} C_m^n \text{ при } 0 \le k - b, \text{ иначе } 0))$

   Характеристика - "развернуть(второйтерм)".

2. Определение характеристики класса.

   (a) упрощэкв(x1). Упрощающий переход к характеристикам других классов. Пример:

   $\forall_{APQ}(\mathrm{card}(\mathrm{set}_{xyz}(x - \text{set}\ \&\ x \subseteq A\ \&\ y - \text{set}\ \&\ y \subseteq A\ \&\ P(x \triangle y))) = \mathrm{card}(\mathrm{set}_x(x - \text{set}\ \&\ x \subseteq A\ \&\ P(x))) \cdot \mathrm{card}(\mathrm{set}_{yz}(y - \text{set}\ \&\ y \subseteq A\ \&\ Q(y, z))))$

   Характеристика - "упрощэкв(второйтерм)".

   (b) исключотр(x1). Исключение внутреннего квантора существования в описании класса при определении его мощности. x1 - направление замены. Пример:

   $\forall_{Bgh}(\mathrm{card}(\mathrm{set}_{fx}(\exists_A(g(f, A, x)\ \&\ \text{Отображение}(f, A, B(x)))\ \&\ h(f, x))) = \mathrm{card}(\mathrm{set}_{fxA}(g(f, A, x)\ \&\ \text{отображение}(f, A, B(x))\ \&\ h(f, x))))$

   Характеристика - "исключотр(второйтерм)".

Тождества с условными выражениями

Напоним, что в разделе "Упрощающие тождества" уже рассматривалась связанная с условными выражениями характеристика "варианты".

1. вариант. Обе части тождества не содержат связанных переменных, причем в одну из них входит "вариант", а в другую - не входит. Пример:

   $\forall_{ab}(a-$число $\&\ b-$число $\&\ 0 \le b+\pi\ \&\ 0 \le \pi-b\ \&\ 0 < a \to \arg(a\sin b-(a\cos b)i) = (b-\pi/2$ при $0 < 2b+\pi,$ иначе $b+3\pi/2))$

2. упрощвариант(x1). Тождество, имеющее в заменяющей части условное выражение и позволяющее в каждом из подслучаев перейти к более простому в смысле справочника "оценка" терму. x1 - направление замены. Годится предыдущий пример, у которого характеристика - "упрощвариант(второйтерм)".

Тождества свертки

1. свертка(x1). Теорема представляет собой тождество, обеспечивающее переход к более компактной записи. x1 - направление замены. Пример:

   $\forall_{abc}(a -$ число $\&\ b -$ число $\&\ c -$ число $\to ab + ac = a(b + c))$

   Характеристика - "свертка(второйтерм)".

2. склейка(x1 x2). Тождество, позволяющее перейти от выражения с несколькими вхождениями переменной x1 к выражению с одним вхождением этой переменной. x2 - направление замены. Пример:

   $\forall_{ab}(a -$ число $\&\ b -$ число $\to \sqrt{3}a\sin b + a\cos b = 2a\sin(b + \pi/6))$

   Характеристика - "склейка(b второйтерм)".

3. свертки(x1). Свертка, позволяющая сгруппировать новый терм со старым. x1 - направление замены. Пример:

   $\forall_{abcdepq}(a -$ число $\&\ b -$ натуральное $\&\ c -$ натуральное $\&\ d -$ число $\&\ e -$ число $\&\ q-$натуральное $\&\ \neg(e = 0)\ \&\ p = \min(b, c) \to d(\sin a)^b(\operatorname{tg} a)^q/(\cos a)^c e = (\operatorname{tg} a)^{p+q}(\sin a)^{b-p}d/(\cos a)^{c-p}e)$

   Характеристика - "свертки(второйтерм)".

4. дистрибразвертка(x1). Тождество вида $f(g(x, y), g(x, z)) = g(x, h(y, z))$, где f, h - ассоциативны и коммутативны. x1 - направление замены от заголовка g к заголовку f. Пример:

   $\forall_{abc}(a -$ число $\&\ b -$ число $\&\ c -$ число $\to ab + ac = a(b + c))$

   Характеристика - "дистрибразвертка(первыйтерм)".

Изменение заголовка

1. заголовок(x1 x2). Тождество позволяет переходить от выражений с заголовком x1 к выражениям с заголовком x2 и не имеет существенных антецедентов. Пример:

   $\forall_a(a - \text{rational} \ \& \ \neg(a = 0) \to \text{числитель}(1/a) = \text{знаменатель}(a))/$

   Характеристика - "заголовок(числитель знаменатель)".

2. коммутативно(x1). Тождество преобразует некоммутативную двуместную операцию в операцию с коммутативно-ассоциативным заголовком и хотя бы двумя неконстантными операндами. x1 - направление замены. Пример:

   $\forall_{bcf}(b - \text{set} \ \& \ c - \text{set} \ \& \ f - \text{set} \to b \setminus c \cup b \setminus f = b \setminus (c \cap f))$

   Характеристика - "коммутативно(первыйтерм)".

Выражение с неизвестными

1. нормглуб(x1 x2). Тождество уменьшает глубину вхождения переменной x1. x2 - направление замены. Пример:

   $\forall_{cdefi}(c - \text{число} \ \& \ d - \text{число} \ \& \ e - \text{число} \ \& \ f - \text{число} \ \& \ i - \text{число} \ \& \ \neg(f = 0) \ \& \ \neg(c = 0) \ \& \ i - \text{rational} \ \& \ \neg(\text{знаменатель}(i) - \text{even}) \ \& \ d - \text{rational} \ \& \ \neg(\text{знаменатель}(d) - \text{even}) \to e/(fc^{di}) = e(1/c^d)^i/f)$

   Характеристика - "нормглуб(i второйтерм)".

2. глубина(x1 x2). Тождество перестановочного типа уменьшает глубину переменной x1 до единицы. x2 - направление замены. Пример:

   $\forall_{ab}(a - \text{число} \ \& \ 0 < a \ \& \ b - \text{число} \to (1/a)^b = 1/a^b)$

   Характеристика - "глубина(b первыйтерм)".

3. нормнеизв(x1 x2). Теорема представляет собой тождество, которое можно использовать для уменьшения числа вхождений неизвестной x1. x2 - направление замены. Пример:

   $\forall_{acegh}(\neg(c = 0) \ \& \ \neg(\text{знаменатель}(e) - \text{even}) \ \& \ 0 < g \ \& \ a - \text{число} \ \& \ a - a - \text{число} \ \& \ c - \text{число} \ \& \ e - \text{число} \ \& \ g - \text{число} \ \& \ h - \text{число} \ \& \ e - \text{rational} \to hg^{e-a}(-1)^e/c = h(-g)^e/(cg^a))$

   Характеристики - "нормнеизв(e второйтерм)", "нормнеизв(g первыйтерм)".

4. упростить(x1 x2). Тождество, исключающее сложную операцию над неизвестным выражением с помощью равенства из посылок. x1 - направление замены, x2 - результирующий подтерм с неизвестными. Пример:

   $\forall_{abcdepq}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ d - \text{число} \ \& \ m - \text{число} \ \& \ p - \text{число} \ \& \ 0 \le a + b \ \& \ \neg(c = 0) \ \& \ e - d + bc = mp^2 \ \& \ ac + d = e \to \sqrt{a + b} = \sqrt{m/c}|p|)$

   Характеристика - "упростить(второйтерм p)". В приеме, созданном по этой теореме, последний антецедент идентифицируется с равенством из посылок,

предпоследний - выделен указателем "идентификатор", а его левая часть обрабатывается нормализатором разложения на множители. Теорема выведена специально для создания такого приема.

5. исключчнеизв(x1). Тождество, позволяющее заменить неизвестные подтермы на известные с помощью равенств из посылок. x1 - список номеров антецедентов, идентифицируемых с посылками. Пример:

   $\forall_{abmnxy}(a - \text{число } \& \ b - \text{число } \& \ m - \text{число } \& \ n - \text{число } \& \ x - \text{число } \& \ y - \text{число } \& \ \neg(y = 0) \ \& \ \neg(n = 0) \ \& \ \neg(a = 0) \ \& \ ax + by = 0 \rightarrow mx/(ny) = bm/(an))$

   Предполагается, что прием будет проверять наличие неизвестных в x, y и отсутствие их в a, b. Отличие от предыдущего пункта состоит в том, что здесь содержащих неизвестные исключаемых подвыражений несколкьо. Характеристика - "исключчнеизв(десять)".

6. смнеизв(x1 x2 x3). Тождество, упрощающее относительно неизвестных заданный корневой операнд преобразуемого выражения. x1 - направление замены, x2 - номер корневого операнда, x3 - конъюнкция фильтров, уточняющих контекст. Пример:

   $\forall_{abcd}(\neg(c = 0) \ \& \ a - \text{число } \& \ c - \ \& \ d - \text{число } \& \ 0 \leq a \ \& \ b - \text{число } \& \ 0 \leq b \rightarrow (ba^{d/c})^c = b^c a^d)$

   Тождества данного типа используются при выводе теорем. В данном случае - с целью получения эквивалентности возведения обеих частей уравнения в степень для исключения дробных степеней. Характеристика - "смнеизв(второйтерм 1 и(тип(описать) условие известно(c) не(известно(a))натуральное(d) или(натуральное(c)известно(b))))".

7. значениепеременной. Тождество, у которого в одной части находится переменная, не входящая в противоположную часть. Пример:

   $\forall_{an}(a - \text{число } \& \ n - \text{целое } \& \ n \leq a \ \& \ a < n + 1 \rightarrow [a] = n)$

Перегруппировочные тождества

1. перестановка. Теорема представляет собой тождество перестановочного типа. Пример:

   $\forall_{cde}(\neg(c - 1 = 0) \ \& \ 0 < c \ \& \ 0 < d \ \& \ c - \text{число } \& \ d - \text{число } \& \ e - \text{число } \rightarrow e \log_c d = \log_c(d^e))$

   Заметим, что это же самое тождество одновременно рассматривается и как тождество общей стандартизации в направлении справа налево.

2. ассоциативно(x1). Теорема представляет собой тождество, не изменяющее сложности терма, но обеспечивающее переход к ассоциативно-коммутативному заголовку. x1 - направление замены. Пример:

   $\forall_{abf}(\neg(a - 1 = 0) \ \& \ \neg(b - 1 = 0) \ \& \ 0 < a \ \& \ 0 < b \ \& \ a - \text{число } \& \ b - \text{число } \& \ f - \text{число } \rightarrow f \log_b a = f / \log_a b)$

   Характеристика - "ассоциативно(первыйтерм)".

3. замещение(x1). Теорема представляет собой тождество стандартизирующей разгруппировки для ассоциативно-коммутативных операций. x1 - направление замены. Пример:

   $\forall_{ab}(a - \text{комплексное} \ \& \ b - \text{комплексное} \rightarrow \mathrm{Im}(a+b) = \mathrm{Im}(a) + \mathrm{Im}(b))$

   Характеристика - "замещение(второйтерм)".

4. группировки. Тождество с бесповторными частями, имеющими одинаковые множества переменных. Пример:

   $\forall_{ab}(a - \text{число} \ \& \ b - \text{число} \rightarrow -ab = (-a)b)$

5. варьир(x1 x2 x3). Тождество для варьирования сложного терма. x1 - направление замены, x2 - исходная версия терма, x3 - новая версия. Пример:

   $\forall_a(a - \text{число} \ \& \ 0 \leq a \ \& \ 0 \leq \pi - a \rightarrow \sin a = \sqrt{1 - (\cos a)^2})$

   Характеристика - "варьир(второйтерм синус(a) косинус(a))".

6. вхождперем(x1). Тождество для перенесения сложного выражения на более удобное место. x1 - направление замены. Пример:

   $\forall_{abdeghi}(\neg(a = 0) \ \& \ \neg(e + hg^{i/2} = 0) \ \& \ \neg(hg^{i/2} - e = 0) \ \& \ \neg(\text{знаменатель}(b) - \text{even}) \ \& \ a - \text{число} \ \& \ d - \text{число} \ \& \ e - \text{число} \ \& \ g - \text{число} \ \& \ h - \text{число} \ \& \ i - \text{число} \ \& \ b - \text{rational} \ \& \ 0 \leq g \rightarrow d(hg^{i/2} - e)^b/(a(g^i h^2 - e^2)^b) = d/(a(e + hg^{i/2})^b))$

   Характеристика - "вхождперем(первыйтерм)". Радикал из знаменателя переносится в числитель. Числитель предпочтительнее знаменателя, так как при сложении дробных выражений подтермы знаменателей увеличивают число вхождений, а числителей - не увеличивают.

Тождества с невырожденными числовыми атомами

Невырожденные числовые атомы - простейшие численные характеристики нечисловых объектов - $l(AB)$, $\angle(ABC)$, "масса(a)", и т.п. Для алгоритмизации теорем с такими атомами понадобились свои типы приемов.

1. числовойатом. Тождество для невырожденных числовых атомов. Пример:

   $\forall_{ABC}(A - \text{точка} \ \& \ B - \text{точка} \ \& \ C - \text{точка} \ \& \ \neg(A = C) \ \& \ \neg(B = C) \ \& \ \neg(A = B) \rightarrow \angle(ABC) + \angle(BCA) + \angle(BAC) = \pi)$

2. пропорция(x1 x2). Соотношение пропорциональности для числовых атомов x1,x2. Пример:

   $\forall_{ABCDE}(A - \text{точка} \ \& \ B - \text{точка} \ \& \ C - \text{точка} \ \& \ D - \text{точка} \ \& \ E - \text{точка} \ \& \ \neg(A = E) \ \& \ \neg(B = C) \ \& \ \neg(D = E) \ \& \ \neg(A = D) \ \& \ B \in \text{прямая}(AD) \ \& \ \text{прямая}(DE) \parallel \text{прямая}(BC) \ \& \ c \in \text{прямая}(AE) \ \& \ \neg(\text{прямая}(AE) = \text{прямая}(BC)) \rightarrow l(AD)l(AC) = l(AE)l(AB))$

   Характеристики - "пропорция($lAD \ l(AE)$)", "пропорция($lAD \ l(AB)$)", "пропорция($lAC \ l(AE)$)", "пропорция($lAC \ l(AB)$)".

3. Упрощение либо исключение невырожденных числовых атомов.

   (a) числатом. Тождество выражает сложный числовой атом через более простые. Пример:

   $\forall_{ab}(a-\text{set}\ \&\ b-\text{set}\ \&\ \text{конечное}(a)\ \&\ \text{конечное}(b)\ \&\ \text{непересек}(a,b) \rightarrow \text{card}(a \cup b) = \text{card}(a) + \text{card}(b))$

   (b) числатомы(x1). Тождество, выражающее терм с невырожденными числовыми атомами через численные параметры. x1 - направление замены. Пример:

   $\forall_{ABCKabcdef}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ \text{прямкоорд}(K)\ \&\ \neg(A = C)\ \&\ \neg(A = B)\ \&\ \text{коорд}(\text{вектор}(AB), K) = (a,b,c)\ \&\ \text{коорд}(\text{вектор}(AC), K) = (d,e,f) \rightarrow \cos(\angle(BAC)) = (ad+be+cf)/(\sqrt{a^2+b^2+c^2} \cdot \sqrt{d^2+e^2+f^2}))$

   Характеристика - "числатомы(второйтерм)". Заметим, что данную характеристику имеют также многие "чисто геометрические" теоремы, у которых численные параметры появляются либо за счет вспомогательных вычислений в антецедентах, либо вследствие непосредственной идентификации с посылками, имеющими численные параметры (например, с соотношениями пропорциональности).

   (c) числпарам(x1). Тождество, использующее равенство из контекста для выражения числового атома через численные параметры. x1 - номер антецедента, идентифицируемого с равенством, x2 - направление замены. Пример:

   $\forall_{Kav}(\text{вправо}(v,K)\ \&\ \text{длина}(v) = a \rightarrow \text{крд}(v,K,1) = a)$

   Характеристика - "числпарам(2 второйтерм)".

   (d) извлечпарам(x1). Тождество, выражающее невырожденный числовой атом с помощью явно указанных в антецедентах значений других числовых атомов. x1 - направление замены. Пример:

   $\forall_{ABCDab}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ D - \text{точка}\ \&\ \neg(A = D)\ \&\ \neg(A = C)\ \&\ \neg(A = B)\ \&\ \angle(BAC) = a\ \&\ \angle(CAD) = b\ \&\ \text{однасторона}(B, D, \text{прямая}(AC)) \rightarrow \angle(BAD) = |a - b|)$

   Характеристика - "извлечпарам(второйтерм)".

4. Выражение одного числового атома через другие.

   (a) числзнач. Тождество, в одной из частей которого расположен невырожденный числовой атом, не встречающийся в противоположной части, не являющейся числовым атомом, но содержащей невырожденные числовые атомы. Пример:

   $\forall_{ABCDE}(B-\text{точка}\ \&\ C-\text{точка}\ \&\ D-\text{точка}\ \&\ E-\text{точка}\ \&\ \text{центр}(A, \text{фигура}(BCDE))\ \&\ \text{квадрат}(BCDE) \rightarrow l(BC) = \sqrt{2} l(AB))$

5. Числзнач. Тождество, в одной части которого находится нечисловая переменная, а в другой - выражение ее через невырожденные числовые атомы. Пример:

   $\forall_z(z - \text{комплексное} \rightarrow z = |z|(\cos(\arg(z)) + i \sin(\arg(z))))$

Нечисловые тождества

1. Равно. Нечисловое равенство. Пример:

   $\forall_{cef}(c - \text{set}\ \&\ e - \text{set}\ \&\ f - \text{set} \to (c \cup f) \cap (e \cup f) = f \cup (c \cap e))$

Тождества для атомарных выражений

Напомним, что выражение считается атомарным, если оно либо однобуквенное, либо тип его значения отличается от типов значений операндов.

1. равны. Равенство двух атомарных выражений. Пример:

   $\forall_{ABCDEF}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ D - \text{точка}\ \&\ E - \text{точка}\ \&\ F - \text{точка}\ \&\ \neg(D = E)\ \&\ \neg(A = B)\ \&\ C \in \text{прямая}(AB)\ \&\ C \in \text{прямая}(DE)\ \&\ F \in \text{прямая}(AB)\ \&\ F \in \text{прямая}(DE)\ \&\ \neg(C = F) \to \text{прямая}(AB) = \text{прямая}(DE))$

2. Атомарное(x1). Тождество для атомарных выражений типа x1. Пример:

   $\forall_{ABCa}(a - \text{число}\ \&\ A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ \neg(A = B)\ \&\ C \in \text{прямая}(AB)\ \&\ 0 \le a\ \&\ l(AC) = al(AB)\ \&\ \text{точкалуча}(A, B, C) \to a \cdot \text{вектор}(AB) = \text{вектор}(AC))$

   Характеристика - "Атомарное(Вектор)".

3. равно. Равенство двух переменных либо конъюнкция таких равенств. Пример:

   $\forall_{ABE}(\text{эллипсоид}(E)\ \&\ \text{центр}(A, E)\ \&\ \text{центр}(B, E) \to A = B)$

Тождества с координатами

1. Координаты отдельных объектов.

   (a) систкоорд. Тождество для определения координат объекта. Пример:

   $\forall_{AEKabcd}(a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{число}\ \&\ \text{прямкоорд}(K)\ \&\ \text{парабола}(E)\ \&\ \text{коорд}(E, K) = \text{set}_{xy}(ay^2 + by + cx + d = 0\ \&\ x - \text{число}\ \&\ y - \text{число})\ \&\ \text{вершина}(A, E) \to \text{коорд}(A, K) = ((b^2 - 4ad)/(4ac), -b/(2a)))$

   (b) коорд. Тождество для координат объектов. Сюда относятся тождества, в которых присутствуют переменные для каких-либо координат отдельных объектов, идентифицированные в антецедентах. В частности, допускаются тождества, выражающие координаты какого-либо объекта через координаты других объектов. Пример:

   $\forall_{ABCDKabcd}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ D - \text{точка}\ \&\ \text{прямкоорд}(K)\ \&\ \neg(C = D)\ \&\ \neg(A = B)\ \&\ \text{коорд}(\text{вектор}(AB), K) = (a, b)\ \&\ \text{коорд}(\text{вектор}(CD), K) = (c, d)\ \&\ \text{прямая}(AB) \perp \text{прямая}(CD) \to ac + bd = 0)$

   (c) крд(x1). Тождество, в заменяемой части которого расположено выражение для отдельной координаты объекта. x1 - направление замены. Пример:

   $\forall_{ABK}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ \text{прямкоорд}(K)\ \&\ \text{Трехмерн}(K)\ \&\ \text{вперед}(\text{вектор}(AB), K) \to \text{крд}(B, K, 2) = \text{крд}(A, K, 2) + l(AB))$

   Характеристика - "крд(второйтерм)".

(d) нормкрд. Равенство либо конъюнкция равенств, содержащих отдельные координаты. Пример:

$\forall_{AKabc}$(Трехмерн(K) & $A-$точка & коорд(A,K) = (a,b,c) → крд($A,K,1$) = a & крд($A,K,2$) = b & крд($A,K,3$) = c)

(e) новкадр. Тождество, выражающее координаты объекта в одной системе координат через его координаты в другой системе. Пример:

$\forall_{ABCDKQabcdefxy}$($A-$точка & $B-$точка & $C-$точка & $D-$точка & систкоорд(K) & систкоорд(Q) & $K = (A,B,C)$ & коорд(A,Q) = (a,b) & коорд(вектор(AB), Q) = (c,d) & коорд(вектор(AC), Q) = (e,f) & коорд(D,K) = (x,y) → коорд(D,Q) = $(a+xc+ye, b+xd+yf)$)

2. Связь невырожденных числовых атомов с координатами.

(a) значпарам(x1). Тождество, выражающее невырожденный числовой атом через параметры координат. x1 - направление замены. Пример:

$\forall_{Kabd}$(прямкоорд(K) & Вектор(d) & коорд(d,K) = (a,b) → длина(d) = $\sqrt{a^2+b^2}$)

Характеристика - "значпарам(второйтерм)".

(b) числкоэфф. Тождество, связывающее невырожденные числовые атомы с параметрами координат. Пример:

$\forall_{ABCDK}$(систкоорд(K) & $K = (A,B,C)$ & $D \in$ прямая(AB) & точкалуча(A,B,D) & коорд(D,K) = $(a,b) \to l(AD) = al(AB)$)

3. Координаты множеств объектов.

(a) уравнмножество. Тождество либо дизъюнкция тождеств, определяющих координаты заданного множества объектов.

$\forall_{ABEKabcp}$($c-$число & прямкоорд(K) & парабола(E) & вершина(A,E) & коорд(A,K) = (a,b) & фокпараметр(E) = p & направлпараболы(E,B) & коорд(B,K) = $(c,0)$ & $\neg(c=0)$ → коорд(E,K) = $\text{set}_{xy}((y-b)^2-2p\text{sg}(c)(x-a)=0$ & $x-$число & $y-$число))

(b) числсвязь. Тождество без невырожденных числовых атомов, содержащее параметры уравнения для координат множества точек. Пример:

$\forall_{ABCDKabcdef}$($a-$число & $b-$число & $c-$число & $d-$число & $e-$число & $f-$число & $A-$точка & $B-$точка & $C-$точка & $D-$точка & прямкоорд(K) & $\neg(B=C)$ & $\neg(A=D)$ & коорд(прямая(AD), K) = $\text{set}_{xy}(ax+by+c=0$ & $x-$число & $y-$число) & прямая(AD) $\perp$ прямая(BC) & коорд(прямая(BC), K) = $\text{set}_{uv}(du+ev+f=0$ & $u-$число & $v-$число) → $ad+be=0$)

(c) новконтекст. Тождество, выражающее координаты множества точек в одной системе координат через его координаты в другой системе. Пример:

$\forall_{ABCKMTabcdefp}$($A-$точка & $B-$точка & $C-$точка & систкоорд(K) & систкоорд(T) & $T = (A,B,C)$ & коорд(A,K) = (a,b) & коорд(вектор(AB), K) = (c,d) & коорд(вектор(AC), K) = (e,f) & коорд(M,K) = $\text{set}_{xy}(x-$число & $y-$число & $p(x,y)$) → коорд(M,T) = $\text{set}_{xy}(x-$число & $y-$число & $p(a+cx+ey, b+dx+fy)$))

Тождества рекурсивного характера

1. префикснаярекурсия(x1). Тождество позволяет перейти от сложной операции с подтермом "префикс(A B)" к такой же операции с подтермом B. x1 - направление замены. Пример:

   $\forall_{ab}(b - \text{слово } \& \neg(a \in \{;b\}) \to \text{card}\{(a;b)\} = \text{card}\{;b\} + 1)$

   Напомним, что $\{a;b\}$ означает "перечень(префикс(a,b))", $\{;b\}$ - "перечень(b)".

   Характеристика - "префикснаярекурсия(второйтерм)".

2. натуральное(x1 x2). Тождество позволяет выразить сложную операцию с натуральным параметром x1 через такую же операцию, в которой переменная x1 заменена на выражение, имеющее натуральное значение, меньшее x1. x2 - направление замены. Пример:

   $\forall_{Amn}(A - \text{функция } \& \text{ Val}(A) \subseteq \mathbb{R} \ \& \ n - \text{натуральное } \& \ m - \text{натуральное } \&$
   $\text{Dom}(A) = \{1, \ldots, m\} \times \{1, \ldots, m\} \ \& \ 0 \leq n - 2 \to A^n = A \cdot A^{n-1})$

   Характеристика - "натуральное(n второйтерм)".

3. длинанабора(x1 x2). Тождество позволяет выразить сложную операцию с параметром x1, значением которого служит набор, через такую же операцию, в которой переменная x1 заменена на выражение, имеющее своим значением набор меньшей длины. x2 - направление замены. Пример:

   $\forall_{ABafgin}(a - \text{слово } \& \ n - \text{натуральное } \& \ l(a) = n \ \& \ \forall_x(x \in \text{Val}(a) \to x -$
   $\text{функция}) \ \& \ i \in \{1, \ldots, n-1\} \ \& \ f = a(i) \ \& \ g = a(i+1) \to \text{произведение}(a) =$
   $\text{произведение}(\lambda_j((a(j) \text{ при } j < i, \text{ иначе } (\lambda_y(f(g(y)), y \in \text{Dom}(g) \ \&$
   $g(y) \in \text{Dom}(f)) \text{ при } j = i, \text{ иначе } a(j+1))), j \in \{1, \ldots, n-1\})))$

   Характеристика - "длинанабора(x1 второйтерм)".

Использование тождества для вычислений

1. вычислить. Тождество для сведения сложного вычисления к цепочке более простых вычислений. Пример:

   $\forall_{abmn}(m = \text{card}(a) \ \& \ n = \text{card}(b) \ \& \ m - \text{число } \& \ n - \text{число} \to \text{card}(a \times b) = mn)$

2. Функвых. Тождество, дающее явное выражение для функциональной характеристики объекта. Пример:

   $\forall_{PXa}(\text{веропространство}(P) \ \& \ \text{случвеличина}(X, P) \ \& \ a - \text{число } \&$
   $\text{Пуассон}(X, P, a) \to \text{рядраспред}(X, P) = \lambda_i(a^i/i! \exp(-a), i \in \mathbb{N}+))$

19.4 Эквивалентности

Упрощающие эквивалентности

1. Эквивалентности общей стандартизации.

(a) общнорм(x1). Общая стандартизация утверждения, не имеющего вида дизъюнкции либо конъюнкции. Заменяющее утверждение - элементарное. x1 - направление замены. Пример:

$\forall_{abc}(\neg(b = 0)\ \&\ a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число} \to ab = bc \leftrightarrow a = c)$

Характеристика - "общнорм(второйтерм)".

(b) Сокращение(x1). Теорема представляет собой эквивалентность без существенных посылок, переменные заменяющего утверждения которой образуют собственное подмножество переменных заменяемого. x1 - направление замены. Пример:

$\forall_{abc}(a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ 0 < c\ \&\ \neg(a - 1 = 0)\ \&\ 0 < b\ \&\ 0 < a \to \log_a b - \log_a c \leftrightarrow b - c = 0)$

Характеристика - "Сокращение(второйтерм)".

(c) вычеркивание(x1). Теорема представляет собой эквивалентность, заменяемое утверждение которой содержит все переменные теоремы, а заменяющее - лишь собственное подмножество этих переменных. x1 - направление замены. Пример:

$\forall_{abc}(a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ \neg(b = 0)\ \&\ 0 < b \to a/b < c/b \leftrightarrow 0 < c - a)$

Характеристика - "вычеркивание(второйтерм)".

2. конечное(x1). Эквивалентность имеет в одной части одноместный предикат $P(x)$, а в другой - дизъюнкцию равенств вида $x = t$, где t - константное выражение. x1 - направление перехода к дизъюнкции. Пример:

$\forall_a(a - \text{boolean} \leftrightarrow a = 0\ \vee\ a = 1)$

Характеристика - "конечное(второйтерм)".

3. Эквивалентности с описателями.

(a) нормсвязок(x1). Заменяемая часть эквивалентности содержит описатели, а заменяющая - не имеет связанных переменных. Ее сложность не превосходит сложности заменяемой части. x1 - направление замены. Пример:

$\forall_{abc}(a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число} \to \text{card}(\text{set}_x(x - \text{число}\ \&\ ax^2 + bx + c = 0)) = 1 \leftrightarrow \neg(a = 0)\ \&\ b^2 - 4ac = 0\ \vee\ a = 0\ \&\ \neg(b = 0))$

Характеристика - "нормсвязок(второйтерм)".

4. усиление(x1). Усиление элементарного утверждения. Пример:

$\forall_{AB}(A - \text{set}\ \&\ B - \text{set}\ \&\ \text{конечное}(B)\ \&\ 0 \le \text{card}(A) - \text{card}(B) \to A \subseteq B \leftrightarrow A = B)$

Характеристика - "усиление(второйтерм)".

5. Уменьшение оценки сложности.

(a) уменьшение(x1). Эквивалентность исключает символы с наибольшей оценкой сложности. x1 - направление замены. Пример:

$\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ \neg(b = 0) \ \& \ c - \text{число} \to c < a/b \leftrightarrow 0 < b \ \& \ bc < a \ \vee \ b < 0 \ \& \ a < bc)$

Характеристика - "уменьшение(второйтерм)".

6. сборка(x1). Теорема представляет собой эквивалентность, используемую для сокращенной переформулировки утверждений. x1 - направление замены. Пример:

$\forall_{ab}(a - \text{число} \ \& \ b - \text{число} \to 0 < ab \leftrightarrow a < 0 \ \& \ b < 0 \ \vee \ 0 < a \ \& \ 0 < b)$

Характеристика - "сборка(первыйтерм)".

7. коммутатор. Теорема преобразует бесповторное утверждение так, чтобы выявилась симметрия по переменным. Пример:

$\forall_{ab}(a - \text{число} \ \& \ b - \text{число} \ \& \ \neg(b = 0) \to a/b \leq 0 \leftrightarrow ab \leq 0)$

8. констнабор(x1 x2). Эквивалентность упрощает утверждение относительно неконстантных подвыражений. x1 - список переменных, идентифицируемых с константными выражениями, x2 - направление замены. Пример:

$\forall_{abcd}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ d - \text{число} \ \& \ 0 \leq d \ \& \ 0 \leq b \ \& \ 0 < a \ \& \ 0 < c \to a\sqrt{b} - c\sqrt{d} \leftrightarrow a^2b - c^2d = 0)$

Характеристика - "констнабор(a c первыйтерм)". Неконстантные радикалы исключаются, а возникающие новые подтермы a^2, c^2 константные.

Декомпозирующие эквивалентности

1. и(x1). Декомпозиция элементарного утверждения в конъюнкцию бескванторных утверждений. x1 - направление замены. Пример:

$\forall_{abc}(b - \text{число} \ \& \ c - \text{число} \to a \in (b, c) \leftrightarrow a - \text{число} \ \& \ b < a \ \& \ a < c)$

Характеристика - "и(второйтерм)".

2. или(x1). Декомпозиция элементарного утверждения в дизънкцию бескванторных утверждений. x1 - направление замены. Пример:

$\forall ab(a - \text{число} \ \& \ b - \text{число} \to ab = 0 \leftrightarrow a = 0 \ \vee \ b = 0)$

Характеристика - "или(второйтерм)".

Перегруппировочные эквивалентности

1. группировка(x1). Теорема представляет собой эквивалентность, позволяющую группировать в одной части бинарного отношения две переменные, ранее расположенные в разных частях. x1 - направление замены. Пример:

$\forall_{cde}(c - \text{число} \ \& \ d - \text{число} \ \& \ e - \text{число} \to e = c + d \leftrightarrow c = e - d)$

Характеристики - "группировка(первыйтерм)" и "группировка(второйтерм)".

2. разделение(x1). Теорема представляет собой эквивалентность, позволяющую переносить в разные части двуместного отношения две переменные, ранее расположенные в одной части. x1 - направление замены. Пример:

   $\forall_{ac}(a - \text{число} \ \& \ c - \text{число} \to c = -a \leftrightarrow a + c = 0)$

   Характеристика - "разделение(первыйтерм)".

3. раздпарам(x1). Теорема представляет собой эквивалентность, в заменяющей части которой расположено такое равенство $A = B$, что некоторые два переменные имеются в A, но отсутствуют в B, и обратно, некоторые две переменные имеются в B, но отсутствуют в A. Заменяемая часть имеет вид равенства, содержащего все указанные переменные, причем обе части равенства неоднобуквенные. x1 - направление замены. Если в направлении x1 эквивалентность осуществляет общую стандартизацию, то характеристка "раздпарам(x1)" не используется. Это не мешает вводить ее для противоположного направления. Пример:

   $\forall_{abdf}(\neg(d = 0) \ \& \ \neg(f = 0) \ \& \ a - \text{число} \ \& \ b - \text{число} \ \& \ d - \text{число} \ \& \ f - \text{число} \to af = bd \leftrightarrow a/d = b/f)$

   Характеристика - "раздпарам(второйтерм)".

4. усмгруппа(x1 x2). Теорема представляет собой эквивалентность, преобразующую одно утверждение с помощью другого таким образом, что новое утверждение содержит параметры обоих исходных утверждений. x1 - номер антецедента, идентифицируемого с другим утверждением, x2 - направление замены. Пример:

   $\forall_{abcd}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ d - \text{число} \ \& \ a = b \to c = d \leftrightarrow a + c = b + d)$

   Характеристика - "усмгруппа(5 второйтерм)".

Кванторные конструкции в эквивалентности

1. развертка(x1 x2). Теорема представляет собой эквивалентность для кванторной расшифровки. x1 - тип возникающего при расшифровке квантора ("длялюбого", "существует"). x2 - направление замены. Пример:

   $\forall_{ab}(a - \text{set} \ \& \ a \subseteq \mathbb{R} \to \text{Нижняягрань}(b, a) \leftrightarrow b - \text{число} \ \& \ \forall_c(c \in a \to b < c))$

   Характеристика - "развертка(длялюбого второйтерм)".

2. Параметрические описания.

   (a) параметризация. Теорема представляет собой эквивалентность с квантором существования в одной из своих частей, которую можно неизбыточным образом использовать для получения явного параметрического описания. Пример:

   $\forall_{ab}(b - \text{set} \ \& \ a - \text{set} \to b \subseteq a \leftrightarrow \exists_c(a = b \cup c \ \& \ c - \text{set}))$

(b) попыткапараметризации. Теорема представляет собой эквивалентность с квантором существования в одной из своих частей, которую можно неизбыточным образом использовать для получения неявного параметрического описания. Пример:

$\forall_{ab}(a - \text{set} \ \& \ b - \text{set} \to \exists_c(c \in a \ \& \ c \in b) \leftrightarrow \neg(\text{непересек}(a, b)))$

(c) связка. Теорема представляет собой эквивалентность с квантором существования в левой части, которую можно использовать для исключения несущественных неизвестных. Пример:

$\forall_{ab}(a - \text{число} \ \& \ b - \text{число} \to \exists_c(c < b \ \& \ a < c \ \& \ c - \text{число}) \leftrightarrow a < b)$

(d) обознач. Теорема представляет собой параметрическое описание объектов заданного типа,задающее стандартный вид обозначения таких объектов. Пример:

$\forall_A(\text{Прямая}(A) \leftrightarrow \exists_{BC}(B - \text{точка} \ \& \ C - \text{точка} \ \& \ \neg(B = C) \ \&$
$A = \text{прямая}(BC)))$

(e) биекция. Теорема представляет собой такое явное параметрическое описание, у которого параметры однозначно определяются по параметризуемому объекту. Пример:

$\forall_{ab}(b - \text{set} \ \& \ a - \text{set} \to b \subseteq a \leftrightarrow \exists_c(a = b \cup c \ \& \ \text{непересек}(b, c) \ \& \ c - \text{set}))$

3. Исключение квантора.

(a) кванторнаясвертка(x1). Теорема представляет собой эквивалентность для кванторной свертки. x1 - направление замены. Пример:

$\forall_{ab}(a - \text{set} \ \& \ b - \text{set} \to a \subseteq b \leftrightarrow \forall_{ac}(c \in a \to c \in b))$

Характеристика - "кванторнаясвертка(первыйтерм)".

(b) Существует(x1). Заменяемая часть эквивалентности представляет собой квантор, а заменяющая - бескванторная. x1 - направление замены. Пример:

$\forall_{amnx}(m{-}\text{целое} \ \& \ n{-}\text{целое} \ \& \ a{-}\text{boolean} \ \& \ 0 \le n \ \& \ 0 \le m \to \text{двнабор}(x, n) \ \&$
$\text{колич}(x, a) = m \leftrightarrow \exists_b(b \subseteq \{1, \ldots, n\} \ \& \ \text{card}(b) = m \ \& \ x = \lambda_i((a \text{ при } i \in$
$b, \text{ иначе } \neg a), i \in \{1, \ldots, n\})))$

Характеристика - "Существует(первыйтерм)".

(c) нормили(x1). Теорема имеет квантор существования в заменяемой части и дизъюнкцию элементарных утверждений - в заменяющей. x1 - направление замены. Пример:

$\forall_f(\exists_x(x - \text{boolean} \ \& \ f(x)) \leftrightarrow f(0) \ \vee \ f(1))$

Характеристика - "нормили(второйтерм)".

(d) связпарам(x1). Заменяемая часть имеет связанные переменные, а заменяющая - нет. Сложность заменяющей части не более сложности заменяемой. x1 - направление замены. Пример:

$\forall_{mn}(m - \text{целое} \ \& \ n - \text{целое} \to \forall_k(k - \text{целое} \ \& \ m|k \to n|k) \leftrightarrow n|m)$

Характеристика - "связпарам(второйтерм)".

4. Упрощение кванторов.

   (a) связприставка(x1). Заменяемая часть эквивалентности представляет собой квантор. Кванторы в заменяющей части имеют более короткие связывающие приставки, причем сложность символов заменяющей части не более чем сложность заменяемой. x1 - направление замены. Пример:

   $\forall_{Pax}(\forall_z(P(z) \to a(z) - \text{rational} \ \& \ \neg(a(z) = 0)) \to \exists_{yz}(0 < y \ \& \ y - \text{rational} \ \& \ x = a(z)y \ \& \ P(z)) \leftrightarrow \exists_z(P(z) \ \& \ x - \text{rational} \ \& \ 0 < a(z)x))$

   Характеристика - "связприставка(второйтерм)".

Дизъюнктивно-конъюнктивные члены эквивалентности

1. соединение. Теорема представляет собой эквивалентность, заменяющую конъюнкцию бесповторных элементарных утверждений на одно бесповторное элементарное утверждение, причем все указанные утверждения имеют одни и те же переменные. Пример:

   $\forall_{ab}(a - \text{set} \ \& \ b - \text{set} \to a = b \leftrightarrow a \subseteq b \ \& \ b \subseteq a)$

2. упрощобъединение(x1 x2). Эквивалентность для дизъюнктивно - конъюнктивной свертки, упрощающей относительно переменной x1. x2 - направление замены. Пример:

   $\forall_{Aafx}(f(a) \ \& \ a - \text{целое} \ \& \ A \to x = a \ \& \ A \vee a < x \ \& \ x - \text{целое} \ \& \ f(x) \leftrightarrow -[-a] \leq x \ \& \ x - \text{целое} \ \& \ f(x))$

   Характеристика - "упрощобъединение(x второйтерм)".

Эквивалентности для разрешения относительно переменной

1. Явное разрешение относительно неизвестной.

   (a) Уменьшение глубины вхождений неизвестной до единицы.

      i. глуб(x1 x2). Теорема представляет собой эквивалентность, заменяемая часть которой - элементарное утверждение, имеющее вхождения переменной x1, глубина которых (с отбрасывание внешнего отрицания) более 1, а заменяющая часть построена при помощи логических связок из утверждений, содержащих каждое не более одного вхождения переменной x1, и притом глубины 1. x2 - указатель направления замены. Проверяется неизбыточность эквивалентности при решении уравнений с неизвестной x1. Все обычные формулы для решения уравнений (линейное уравнение, квадратное уравнение, показательное уравнение и т.п.) имеют такую характеристику. Исключение составляют формулы, определяющие серии решений. Пример:

      $\forall_{bcx}(b - \text{число} \ \& \ c - \text{число} \ \& \ x - \text{число} \ \& \ \neg(c = 0) \to bx = c \leftrightarrow x = c/b \ \& \ \neg(b = 0))$

      Характеристика - "глуб(x второйтерм)".

ii. заменаусловия(x1 x2). Теорема представляет собой эквивалентность, заменяемая часть которой - конъюнкция элементарных утверждений, глубина вхождения в которые переменной x1 (с отбрасыванием внешнего отрицания) равна 1; заменяющая часть - элементарное утверждение с единственным вхождением переменной x1, глубина которого равна 1. x2 - указатель направления замены. Проверяется неизбыточность эквивалентности при решении уравнений с неизвестной x1. Характеристика указывает на возможность использования теоремы для группировки уже разрешенных относительно неизвестной условий. Пример:

$\forall_{abc}(a - \text{число } \& \ b - \text{число } \& \ c - \text{число} \to c \le a \ \& \ c \le b \leftrightarrow c \le \min(a, b))$

Характеристика - "заменаусловия(c второйтерм)".

iii. сокращнеизв(x1 x2). Теорема представляет собой эквивалентность, заменяемая часть которой есть дизъюнктивно - конюнктивная конструкция из элементарных утверждений, глубина вхождения в которые переменной x1 (с отбрасыванием внешнего отрицания) равна 1; заменяющая часть - элементарное утверждение с единственным вхождением переменной x1, глубина которого равна 1. x2 - указатель направления замены. Пример:

$\forall_{abc}(b - \text{set } \& \ c - \text{set} \to a \in b \cup c \leftrightarrow a \in b \ \vee \ a \in c)$

Характеристика - "сокращнеизв(a первыйтерм)".

iv. нормотр(x1 x2). Теорема представляет собой эквивалентность, заменяемая часть которой есть элементарное утверждение с заголовком "не", заменяющая - элементарное утверждение без отрицания, причем глубина вхождения переменной x1 в заменяемое и заменяющее утверждения (без учета отрицания) равна 1, а число ее вхождений в каждое из этих утверждений равно 1. x2 - указатель направления замены. Пример:

$\forall_a(a - \text{число } \& \ a \le 0 \to \neg(a = 0) \leftrightarrow a < 0)$

Характеристика - "нормотр(a второйтерм).

v. неизвперем(x1 x2). Теорема представляет собой эквивалентность, заменяемая часть которой - конъюнкция элементарных утверждений, глубина вхождения в которые переменной x1 (с отбрасыванием внешнего отрицания) равна 1; заменяющая часть - дизъюнкция конъюнкций, каждая из которых имеет единственный содержащий переменную x1 член, причем глубина вхождения x1 в этот член равна 1. x2 - указатель направления замены. Пример:

$\forall_{abc}(a - \text{число } \& \ b - \text{число } \& \ c - \text{число} \to a \le c \ \& \ b \le c \leftrightarrow 0 < a - b \ \& \ a \le c \ \vee \ \neg(0 < a - b) \ \& \ b < c)$

Характеристика - "неизвперем(c второйтерм)".

(b) Получение параметрического описания неизвестных.

i. неизвпарам(x1 x2). Теорема представляет собой эквивалентность, дающую явное параметрическое описание значений неизвестной x1. x2 - направление замены. Пример:

$\forall_{ax}(a - \text{число } \& \ x - \text{число} \to \cos x = a \leftrightarrow |a| \le 1 \ \& \ \exists_n(n - \text{целое } \& \ (x = 2\pi n + \arccos a \ \vee \ x = 2\pi n - \arccos a)))$

Характеристика - "неизвпарам(x второйтерм)".

(c) Кванторная импликация.

i. функразр(x1). Теорема представляет собой эквивалентность, явно разрешающую кванторную импликацию относительно функциональной переменной. x1 - направление замены. Пример:

$\forall_{afk}(k - \text{целое} \to \forall_n(n - \text{натуральное} \ \& \ k \le n \to f(n) = af(n-1)) \leftrightarrow \forall_n(n - \text{натуральное} \ \& \ k - 1 \le n \to f(n) = f(k-1)a^{n-k+1}))$

Характеристика - "функразр(второйтерм)".

(d) Неизвестные(x1 x2). Теорема представляет собой эквивалентность, полученную для разрешения группы утверждений относительно неизвестных списка x1. x2 - направление замены. Пример:

$\forall_{abcdxy}(c = a^2 - 4b \ \& \ d = \sqrt{c} \to x + y = a \ \& \ xy = b \leftrightarrow 0 \le c \ \& \ (x = (a-d)/2 \ \& \ y = (a+d)/2 \ \vee \ x = (a+d)/2 \ \& \ y = (a-d)/2))$

Характеристика - "Неизвестные(x y второйтерм)".

2. Выражение одной неизвестной через другие.

(a) пропорцнеизв(x1 x2 x3). Теорема позволяет выразить неизвестную x1 через неизвестную x2. x3 - направление замены. Пример:

$\forall_{abxy}(a - \text{число} \ \& \ b - \text{число} \ \& \ x - \text{число} \ \& \ y - \text{число} \ \& \ \neg(a = 0) \to ax = ay + b \leftrightarrow x = y + b/a)$

Характеристика - "пропорцнеизв(x y второйтерм)".

(b) неизвестные(x1 x2 x3). Теорема представляет собой эквивалентность, полученную для выражения неизвестной x1 через неизвестные, входящие в выражения, идентифицируемые с переменными списка x2. x3 - направление замены. Пример:

$\forall_{abcxy}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ x - \text{число} \ \& \ y - \text{число} \to ax^2 + bxy + cy^2 = 0 \leftrightarrow \neg(a = 0) \ \& \ 0 \le b^2 - 4ac \ \& \ (x = y(\sqrt{b^2 - 4ac} - b)/(2a) \ \vee \ x = -y(\sqrt{b^2 - 4ac} + b)/(2a)) \ \vee \ a = 0 \ \& \ (y = 0 \ \vee \ bx + cy = 0) \ \vee \ b^2 - 4ac < 0 \ \& \ x = 0 \ \& \ y = 0)$

Характеристика - "неизвестные(x y второйтерм)".

3. Упрощение выражений с неизвестными.

(a) неизвоценка(x1 x2). Теорема представляет собой эквивалентность, применение которой в направлении x1 позволяет получить более простые выражения с неизвестными. x2 - фильтр, уточняющий контекст. Пример:

$\forall_{bdegh}(b - \text{число} \ \& \ d - \text{число} \ \& \ e - \text{число} \ \& \ g - \text{натуральное} \ \& \ 0 \le d \to 2bh + d^g e^2 - b^2 = h^2 \ \& \ h - \text{число} \ \& \ 0 \le e(h - b) \leftrightarrow h = b + e \cdot d^{g/2})$

Характеристика - "неизвоценка(первыйтерм и(тип(описать) условие не(известно(d)) натуральное(g) известно(h) смнеизв(b)))". Фильтр "смнеизв(b)" указывает, что если b ненулевое, то содержит неизвестные.

4. Подготовка возможности явного разрешения.

   (a) неизвтермы(x1 x2 x3 x4). Теорема представляет собой эквивалентность, подготавливающую возможность разрешения относительно заданного подвыражения x1 с неизвестными при помощи нормализатора уравнений x2. x3 - список переменных, идентифицируемых с содержащими неизвестные подтермами, x4 - направление замены. Если заменяющий терм устойчив к общей стандартизации, то ссылка на нормализатор x2 может отсутствовать. Пример:

   $\forall_{abcdefghijk}(0 < a \ \& \ b - 2c = f \ \& \ d = ae \ \& \ g - 2c = h \to ie^b + jd^c + ka^g = 0 \leftrightarrow ie^f((e/a)^c)^2 + j(e/a)^c = -ka^h)$

   Характеристика - "неизвтермы($(e/a)^c$ квадруравн b c g второйтерм)". Теорема представляет собой квазипротокол, преобразующий показательное уравнение к виду квадратного уравнения и разрешающие последнее нормализатором "квадруравн".

   (b) Неизвестная(x1 x2). Теорема представляет собой эквивалентность, подготавливающую возможность разрешения относительно неизвестной x1. x2 - направление замены. Пример:

   $\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \to a\cos c - a\sin c \le b \leftrightarrow a\sqrt{2}\cos(c + \pi/4) \le b)$

   Характеристика - "Неизвестная(c второйтерм)".

   (c) дизъюнктблок(x1 x2). Свертка дизъюнктивного условия, приводящая к разрешимому относительно неизвестной x1 утверждению. x2 - направление замены. Пример:

   $\forall_a(a - \text{число} \to \sin a = 0 \ \vee \ \cos a = 0 \leftrightarrow \sin(2a) = 0)$

   Характеристика - "дизъюнктблок(a второйтерм)".

5. Декомпозиция утверждений с неизвестными.

   (a) уравнвариант(x1 x2). Теорема преобразует уравнение к виду, допускающему декомпозицию. x1 - конъюнкция фильтров, x2 - направление замены. Пример:

   $\forall_{abcde}(a{-}\text{целое} \ \& \ b{-}\text{целое} \ \& \ c{-}\text{целое} \ \& \ d{-}\text{целое} \ \& \ e{-}\text{целое} \to ab{+}ac{+}bd = e \leftrightarrow (a + d)(b + c) = e + cd)$

   Характеристика - "уравнвариант(и(не(известно(a))не(известно(b)) целое(c) целое(d) целое(e) не(заголовок(терм($e + cd$)0)))) второйтерм)".

6. Упрощение одного утверждения с неизвестными при помощи другого.

   (a) исклповтор(x1 x2 x3 x4). Теорема представляет собой эквивалентность, преобразующую некоторое утверждение задачи с помощью утверждения из контекста. x1 - номер антецедента, идентифицируемого со вторым утверждением, x2 - направление замены, x3 - общая переменная двух утверждений, исключаемая при замене и идентифицируемая с неизвестным подтермом, x4 - терм "известно(...)", перечисляющий все переменные, идентифицируемые с термами без неизвестных. Пример:

$\forall_{ab}(a - \text{число } \&\ \cos(2a) = b \to \neg(\sin a = 0) \leftrightarrow \neg(1 - b = 0))$

Характеристика - "исклповтор(2 второйтерм a известно(b))".

(b) сокращПлюс(x1). Эквивалентность преобразует одно уравнение задачи на исследование с помощью другого, исключая неизвестное подвыражение. x1 - список всех переменных заменяющей части, которые могут содержать неизвестные. Пример:

$\forall_{abcxy}(ax = b \ \&\ \neg(c = 0) \to ac = y \leftrightarrow xy = bc)$

Характеристика - "сокращПлюс(x y)". Исключено было содержащее неизвестные подвыражение a.

7. альтзначения(x1). Теорема преобразует утверждение к альтернативным сложным подвыражениям с неизвестными. x1 - направление замены. Пример:

$\forall_{abcd}(a - \text{число } \&\ b - \text{число } \&\ c - \text{число } \&\ d - \text{число } \&\ 0 < a \to ba^c + d = 0 \leftrightarrow 0 < b \ \&\ d < 0 \ \&\ \log_2 a \cdot c - \log_2(-d) = -\log_2 b \ \vee\ b < 0 \ \&\ 0 < d \ \&\ \log_2 a \cdot c - \log_2 d = -\log_2(-b) \ \vee\ b = 0 \ \&\ d = 0)$

Характеристика - "альтзначения(второйтерм)". Теорема позволяет перейти от показательных выражений к логарифмическим.

8. сложнреш. Эквивалентность для решения уравнения приводит к чрезмерно громоздким выражениям без неизвестных. Пример:

$\forall_{abcdpqrx}(\neg(a = 0) \ \&\ p = (3ac - b^2)/(3a^2) \ \&\ q = (2b^3 - 9abc - 27a^2d)/(27a^3) \ \&\ r = q^2/4 + p^3/27 \ \&\ 0 < r \to ax^3 + bx^2 + cx = d \leftrightarrow x = \sqrt[3]{-q/2 + \sqrt{r}} + \sqrt[3]{-q/2 - \sqrt{r}} - b/(3a))$

Характеристика указывает, что прием для формулы Кардано должен применяться лишь в крайних случаях (например, если все коэффициенты - константы).

9. независит(x1). Теорема указывает способ подбора корней, не зависящих от заданных параметров. x1 - список исключаемых переменных, идентифицируемых с выражениями, имеющими запрещенные параметры. Пример:

$\forall_{abdef}(a - b = de + f \ \&\ e = 0 \ \&\ f = 0 \to a = b)$

Характеристика - "независит(d)". Теорема представляет собой квазипротокол, преобразуемый в следующую теорему приема:

$\forall_{abcdef}(a - b = c \ \&\ c = de + f \ \&\ a - \text{число} \to a = b \leftrightarrow e = 0 \ \&= f = 0)$

После упрощения разности частей уравнения в результирующей сумме c усматривается слагаемое, имеющее сомножители с исключаемыми параметрами. Чтобы избавиться от этих сомножителей, остаток произведения приравнивается к нулю. Нулю полагается равным и остаток слагаемых выражения c.

Эквивалентности с описателями

1. эквменьше(x1). Эквивалентность упрощает описатель. x1 - направление замены. Пример:

   $\forall_{abcdef}(a - \text{set}\ \&\ f - \text{функция}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{boolean}\ \&\ e - \text{boolean}\ \&\ a \subseteq \mathbb{R} \to \text{образ}(\lambda_x(-f(x), x - \text{число}), a) \subseteq [b, c] \leftrightarrow \text{образ}(\lambda_x(f(x), x - \text{число}), a) \subseteq [-c, -b])$

   Характеристика - "эквменьше(второйтерм)". Двоичные параметры b, d уточняют статус концов промежутка.

Числовые атомы

1. Разрешение уравнений относительно невырожденных числовых атомов.

   Хотя приемы данного раздела относятся скорее к элементарной алгебре, нежели к тем разделам, в которых встречаются невырожденные числовые атомы, они имеют существенную специфику. Разрешение уравнения с невырожденными числовыми атомами даже в простейших случаях, не приводящих к сложным выражениям, может разрушить ход решения задачи, заставив решатель делать множество ненужных вещей. Поэтому обычно разрешающие приемы имеют достаточно большой уровень срабатывания и проверяют различные дополнительные условия. При их создании приходится учитывать, какие уравнения и группы уравнений типичны для рассматриваемых числовых атомов.

   Следует учитывать, что вывод соотношений для числовых атомов обычно нужен лишь для усмотрения цепочки связей между старыми числовыми атомами через вспомогательные числовые атомы. Отвлекаться на обработку этих соотношений, даже простейшую, целесообразно лишь в особых случаях.

   (a) числ(x1). Эквивалентность, разрешающая уравнение относительно невырожденного числового атома. x1 - направление замены. Пример:

   $\forall_{ABabc}(\neg(a = 0) \to al(AB)/c = b \leftrightarrow l(AB) = bc/a)$

   Характеристика - "числ(второйтерм)".

   (b) выражение(x1). Эквивалентность, позволяющая выразить один невырожденный числовой атом через другие. x1 - направление замены. Пример:

   $\forall_{ABCDab}(\neg(a = 0) \to al(AB) = bl(CD) \leftrightarrow l(AB) = bl(CD)/a)$

   Характеристика - "выражение(второйтерм)".

   (c) числтабл(x1). Эквивалентность для разрешения системы уравнений относительно невырожденных числовых атомов. x1 - направление замены. Пример:

   $\forall_{ABCDapq}(0 < p + q \to l(AB) + l(CD) = a\ \&\ pl(AB) = ql(CD) \leftrightarrow l(AB) = aq/(p + q)\ \&\ l(CD) = ap/(p + q))$

   Характеристика - "числтабл(второйтерм)".

(d) исключтеор(x1). Эквивалентность, использующая соотношение пропорциональности двух числовых атомов для исключения одного из них. x1 - направление замены. Пример:

$\forall_{ABCDabcpq}(pl(AB) = ql(CD)\ \&\ \neg(q = 0) \to al(AB) + c = bl(CD) \leftrightarrow (aq - bp)l(AB) + cq = 0)$

Характеристика - "исключтеор(второйтерм)".

2. упрощУмножение(x1). Эквивалентность для упрощения равенства относительно невырожденных числовых атомов. x1 - направление замены. Пример:

$\forall_{ABabcd}(0 \le a\ \&\ 0 \le b\ \&\ 0 \le d\ \&\ c = d^2 \to al(AB)^2 = bc \leftrightarrow \sqrt{a}l(AB) = \sqrt{b}d)$

Характеристика - "упрощУмножение(второйтерм)".

3. эквуглы. Эквивалентность двух равенств невырожденных числовых атомов. Пример:

$\forall_{ABC}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ \triangle(ABC) \to l(AB) = l(BC) \leftrightarrow \angle(BAC) = \angle(BCA))$

4. неравенства. Эквивалентность для двух неравенств с невырожденными числовыми атомами. Пример:

$\forall_{ABC}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ \neg(A = B)\ \&\ \neg(B = C)\ \&\ \neg(A = C) \to l(AB) < l(BC) \leftrightarrow \angle(ACB) < \angle(BAC))$

5. числвыраз(x1). Переформулировка нечислового отношения через отношение для числовых атомов. x1 - направление замены. Пример:

$\forall_{ABCD}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ D - \text{точка}\ \&\ \neg(\text{прямая}(AB) = \text{прямая}(AC))\ \&\ \neg(A = C)\ \&\ \neg(A = B)\ \&\ D \in \text{прямая}(BC) \to \text{биссектриса}(BACD) \leftrightarrow \angle(CAD) = \angle(DAB))$

Характеристика - "числвыраз(второйтерм)".

Координаты

1. числопред(x1). Эквивалентность выражает утверждение через параметры координат. x1 - направление замены. Пример:

$\forall_{ABCKabcdxy}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ \text{систкоорд}(K)\ \&\ \neg(A = B)\ \&\ \text{коорд}(A, K) = (a, b)\ \&\ \text{коорд}(B, K) = (c, d)\ \&\ \text{коорд}(C, K) = (x, y) \to C \in \text{прямая}(AB) \leftrightarrow (c - a)(y - b) = (d - b)(x - a))$

Характеристика - "числопред(второйтерм)".

2. числитель(x1). Эквивалентность переформулирует утверждение через отдельные координаты объектов. x1 - направление замены. Пример:

$\forall_{Kpqr}(\text{Вектор}(p)\ \&\ \text{Вектор}(q)\ \&\ \text{одномерный}(p, K)\ \&\ \text{одномерный}(q, K)\ \&\ \text{одномерный}(r, K)\ \&\ \text{Трехмерн}(K) \to p + q = r \leftrightarrow \text{крд}(p, K, 1) + \text{крд}(q, K, 1) = \text{крд}(r, K, 1))$

Характеристика - "числитель(второйтерм)".

3. уравнкрив. Эквивалентность определяет общий вид уравнения для координат точек множества заданного типа. Пример:

   $\forall_{ABCKP}$(систкоорд(K) & $K = (A, B, C)$ & $P \subseteq$ плоскость(ABC) $\rightarrow$ Прямая(P) $\leftrightarrow$ $\exists_{abc}(a -$ число & $b -$ число & $c -$ число & $\neg(a^2 + b^2 = 0)$ & коорд(P, K) = $\text{set}_{xy}(x -$ число & $y -$ число & $ax + by + c = 0)))$

4. вспомописание(x1). Эквивалентность исключает вспомогательные параметры в задаче на преобразование, имеющей цель "класс". Пример:

   $\forall_{ABCDEKabc}(A -$ точка & $D -$ точка & $E -$ точка & прямкоорд(K) & $\neg(A = E)$ & $K = (A, B, C)$ & коорд(D, K) = (a, b) & коорд(E, K) = $(c, 0)$ & $0 < c$ & $\neg(A = D) \rightarrow a < 0 \leftrightarrow 0 < \angle(DAE) - \pi/2)$

   Характеристика - "вспомописание(второйтерм)".

19.5 Кванторные импликации

Вид консеквента

1. Заголовок консеквента - символ отношения.

   (a) элементарно. Теорема представляет собой кванторную импликацию без существенных посылок, имеющую элементарный консеквент. Примеры:

   $\forall_a(\neg(a \in \emptyset))$

   $\forall_{bc}(b -$ set & $c -$ set $\rightarrow b \subseteq b \cup c)$

   (b) свойство. Теорема представляет собой простую импликацию, консеквентом которой служит одноместное отношение от переменной. Пример:

   $\forall_{Af}(A -$ set & конечное(A) & перестановка(f, A) $\rightarrow$ взаимнооднозначно(f))

   (c) пример. Теорема представляет собой кванторную импликацию без существенных посылок, консеквент которой - элементарное утверждение $f(A_1 \ldots A_n)$ либо отрицание такого утверждения. Все A_i, кроме одного, суть попарно различные переменные. Пример:

   $\forall_{bc}(b -$ set & $c -$ set $\rightarrow$ непересек($c, b \setminus c$))

   (d) опредзначение(x1). Консеквент импликации имеет вид $P(t_1 \ldots t_n)$, причем значение t_i, если вообще существует, однозначно определено значениями $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$. x1 - номер i. Пример:

   $\forall_a(a -$ set & $\neg(a = \emptyset)$ & $a \subseteq \mathbb{R}$ & огрсверху(a) $\rightarrow$ наименьший(sup(a), set_b(верхняягрань(b, a))))

   Характеристика - "опредзначение(1)". Теорема является определением точной верхней грани.

   (e) свойства(x1). Простая импликация, выражающая свойства "сложного" объекта x1. Пример:

   $\forall_a(a -$ число $\rightarrow \cos a \leq 1)$

   Характеристика - "свойства(косинус(a))".

(f) числоценка. Простая импликация, дающая неравенство для невырожденных числовых атомов. Пример:

$\forall_{ABCD}(A-$точка $\&\ B-$точка $\&\ C-$точка $\&\ D-$точка $\&$ трапеция$(ABCD) \to 0 \le \angle(ABC) - \pi/2)$

(g) принадл. Кванторная импликация, консеквент которой имеет вид "принадлежит$(x\ t)$", где x - переменная, не встречающаяся в выражении t. Пример:

$\forall_{ABCDEF}(A-$точка $\&\ B-$точка $\&\ C-$точка $\&\ D-$точка $\&\ E-$точка $\&\ F-$точка $\&\ \neg(E = F)\ \&$ трапеция$(ABCD)\ \&\ E \in$ фигура$(ABCD)\ \&\ F \in$ прямая$(AD)\ \&$ прямая$(EF) \perp$ прямая$(AD) \to F \in$ отрезок$(AD))$

(h) реализация. Кванторная импликация, корневое отношение консеквента которой имеет наибольшую сложность среди всех ее подтермов. Пример:

$\forall_{ABCf}(A - \text{set}\ \&\ B - \text{set}\ \&$ Отображение$(f, A, B)\ \&\ C - \text{set}\ \&\ C \subseteq A \to$ разбиение$(C, \text{set}_x(\exists_y(y \in B\ \&\ x =$ слой$(\text{сужение}(f, C), y)))))$

2. Квантор существования в консеквенте.

(a) допконтекст. Теорема представляет собой кванторную импликацию, консеквент которой - квантор существования от конъюнкции элементарных утверждений. Пример:

$\forall_z(z - \text{комплексное} \to \exists_{ab}(z = a \cdot \cos(b) + (a \cdot \sin(b))i\ \&\ a - \text{число}\ \&\ b - \text{число}\ \&\ 0 \le a\ \&\ -\pi < b\ \&\ b \le \pi))$

(b) нормсуществует. Теорема представляет собой кванторную импликацию (возможно, вырожденную) без существенных посылок с квантором существования в консеквенте. Пример:

$\forall_a(a - \text{число} \to \exists_b(b - \text{число}\ \&\ b < a))$

3. Дизъюнкция в консеквенте.

(a) дизъюнкция. Консеквент имеет вид дизъюнкции. Пример:

$\forall_{ab}(a - \text{число}\ \&\ b - \text{число}\ \&\ 0 < ab \to a < 0\ \vee\ 0 < b)$

(b) альтернатива. Консеквент имеет вид дизъюнкции попарно несовместных утверждений. Пример:

$\forall_{ab}(a - \text{число}\ \&\ b - \text{число} \to a < b\ \vee\ b < a\ \vee\ a = b)$

4. Квантор общности в консеквенте.

(a) квантимплик. Кванторная импликация с кванторной импликацией в консеквенте, представляющей собой расшифровку антецедента. Пример:

$\forall_a(\text{убывмножества}(a) \to \forall_{ij}(i \in \mathbb{N}\ \&\ j \in \mathbb{N}\ \&\ i < j \to a(j) \subseteq a(i)))$

5. Конъюнкция в консеквенте.

Это редкий случай в базе теорем, причем приводимые ниже характеристики большей частью одноразовые - относятся к единственной теореме.

(a) конъюнкция. Теорема представляет собой кванторную импликацию с элементарными антецедентами и консеквентом, имеющим вид конъюнкции элементарных утверждений. Пример:

$\forall_{abcdefghjrK}(\text{коорд}(a,K)=(h,j,r)\ \&\ g-\text{число}\ \&\ \text{систкоорд}(K)\ \&\ \text{Вектор}(a)\ \&\ b=ga\ \&\ \text{коорд}(b,K)=(c,d,e)\to c=gh\ \&\ d=gj\ \&\ e=gr)$

(b) равнчисл. Теорема представляет собой кванторную импликацию с элементарными антецедентами и консеквентом, имеющим вид конъюнкции равенств для невырожденных числовых атомов. Пример:

$\forall_{ABC}(A-\text{точка}\ \&\ B-\text{точка}\ \&\ C-\text{точка}\ \&\ \neg(B=C)\ \&\ \neg(A=C)\ \&\ \neg(A=B)\ \&\ l(AB)=l(AC)\ \&\ \text{прямая}(AB)\perp\text{прямая}(AC)\to\angle(ABC)=\pi/4\ \&\ \angle(ACB)=\pi/4)$

6. Описатели в консеквенте.

(a) отображение. Кванторная импликация имеет элементарные антецеденты и бескванторный консеквент с описателем "отображение". Пример:

$\forall_{abc}(a-\text{число}\ \&\ b-\text{натуральное}\ \&\ d-\text{целое}\ \&\ c-\text{целое}\ \&\ |a|<1\ \&\ 0\le bc+d\to\sum_{i=c}^{\infty}a^{bi+d}=a^d/(1-a^b)-\sum_{i=0}^{c-1}a^{bi+d})$

(b) функперех(x1). Теорема представляет собой частный случай определения функциональной характеристики. x1 - номер операнда консеквента, на котором расположена характеризуемая функция. Пример:

$\text{первообразная}(\lambda_x(1/(1+x^2),x-\text{число}),\lambda_y(\operatorname{arctg}y,y-\text{число}))$

Характеристика - "функперех(1)".

(c) функ. Импликация, полученная из определения отношения путем явного выражения функции через прочие параметры. Пример:

$\forall_g(\forall_x(x\in\text{Dom}(g)\to\text{дифференцируема}(g,x))\ \&\ \text{Dom}(g)\subseteq\mathbb{R}\ \&\ \text{Val}(g)\subseteq\mathbb{R}\ \&\ g-\text{функция}\to\text{первообразная}(\lambda_x(\text{производная}(g,x),x\in\text{Dom}(g)),g))$

Вид антецедентов

1. антецедент. Теорема представляет собой простую импликацию, некоторый существенный антецедент которой содержит все переменные связывающей приставки. Пример:

$\forall_{ab}(a-\text{число}\ \&\ b-\text{число}\ \&\ 0<ab\to 0<a/b)$

Связь консеквента с антецедентами

1. вывод. Теорема представляет собой простую импликацию, у которой каждое выражение консеквента встречается в антецедентах и которая может представлять интерес для создания приема вывода. Пример:

$\forall_{abcd}(b-\text{set}\ \&\ c-\text{set}\ \&\ d-\text{set}\ \&\ a\in d\ \&\ d\subseteq b\cup c\ \&\ \neg(a\in b)\to a\in c)$

2. исключ. Теорема представляет собой простую импликацию с существенными посылками, у которой консеквент содержит не все параметры антецедентов, но каждый параметр консеквента встречается в антецедентах. Пример:

   $\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ a < b \ \& \ b + c \leq 0 \to a + c < 0)$

3. обобщение(x1 x2). Консеквент импликации получается из x1-го антецедента подстановкой вместо переменной x2 выражения $f(\text{x2}, y)$, имеющего единицу по переменной y. Пример:

   $\forall_{bce}(b - \text{set} \ \& \ c - \text{set} \ \& \ e - \text{set} \ \& \ b \subseteq e \ \& \ c \subseteq e \to b \cup c \subseteq e)$

   Характеристики - "обобщение(4 b)" и "обобщение(5 c)".

4. опрзнач(x1). Кванторная импликация однозначно определяет (быть может, в сочетании с рядом антецедентов) своим консеквентом значение переменной x1, которая независимо от этого определяется некоторой другой группой антецедентов. Пример:

   $\forall_{ABCD}(A - \text{точка} \ \& \ B - \text{точка} \ \& \ C - \text{точка} \ \& \ D - \text{точка} \ \& \ \neg(C = D) \ \& \ l(AB) = l(CB) \ \& \ l(AD) = l(DB) \ \& \ D \in \text{прямая}(AB) \ \& \ \neg(A = B) \to \text{прямая}(CD) \perp \text{прямая}(AB))$

   Характеристика - "опрзнач(D)".

5. Специальные типы импликаций.

   (a) транзитивно. Теорема представляет собой свойство транзитивности некоторого двуместного отношения. Пример:

   $\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ a < b \ \& \ b < c \to a < c)$

   (b) транзитоперанд. Теорема имеет вид обобщенной транзитивности: два бесповторных существенных антецедента и бесповторный консеквент, каждый не более чем с 2 переменными; переменные консеквента - симметрическая разность множеств переменных существенных антецедентов. Пример:

   $\forall_{abc}(a - \text{set} \ \& \ b - \text{set} \ \& \ a \subseteq b \ \& \ c \in a \to c \in b)$

   (c) транзитпереход. Теорема представляет собой обобщенную транзитивность относительно части переменных: имеются два существенных антецедента A_1, A_2 и консеквент A_3. Вне непустого пересечения параметров всех этих трех утверждений остаются ровно три переменные x, y, z. При этом x, y входят в A_1; y, z - в A_2; x, z - в A_3. Пример:

   $\forall_{ABCDE}(A{-}\text{точка} \ \& \ B{-}\text{точка} \ \& \ C{-}\text{точка} \ \& \ D{-}\text{точка} \ \& \ E{-}\text{точка} \ \& \ \neg(D = E) \ \& \ \text{непересек}(\text{отрезок}(AB), \text{прямая}(DE)) \ \& \ \text{непересек}(\text{отрезок}(BC), \text{прямая}(DE)) \to \text{непересек}(\text{отрезок}(AC), \text{прямая}(DE)))$

   (d) монотонно. Теорема имеет единственный существенный антецедент $P(x, y)$ и консеквент вида $P(f(x, z), f(y, z))$, где x, y, z - переменные. Пример:

   $\forall_{bcd}(b - \text{set} \ \& \ c - \text{set} \ \& \ d - \text{set} \ \& \ b \subseteq d \to b \cap c \subseteq d \cap c)$

Прямой вывод

1. Вывод безотносительно к неизвестным.

   (a) текобъект(x1). Вывод одноместного предиката, характеризующего текущий объект x1. Пример:

   $\forall_{ABC}(A-\text{точка}\ \&\ B-\text{точка}\ \&\ C-\text{точка}\ \&\ \neg(A=B)\ \&\ \neg(C\in\text{прямая}(AB))\ \to\ \text{эллипс}(\text{Окружность}(ABC)))$

   Характеристика - "текобъект(Окружность(ABC))".

   (b) отношение. Теорема выводит отличный от равенства нечисловой двуместный предикат без новых объектов. Пример:

   $\forall_{ABC}(A-\text{точка}\ \&\ B-\text{точка}\ \&\ C-\text{точка}\ \&\ \neg(A=B)\ \&\ A\in\text{отрезок}(BC)\ \to\ C\in\text{прямая}(AB))$

   (c) констнорм. Теорема выводит равенство переменной константному выражению. Пример:

   $\forall_{ABCDKab}(\text{систкоорд}(K)\ \&\ K=(A,B,C)\ \&\ D\in\text{прямая}(AC)\ \&\ \text{коорд}(D,K)=(a,b)\ \to\ a=0)$

2. Вывод с учетом неизвестных.

   (a) исклтангенс. Вывод следствия условия задачи на описание, в котором исключено сложное понятие. Пример:

   $\forall_{abc}(\arcsin a+b=c\ \to\ a-\sin(c-b)=0)$

   (b) слож. Вывод условия, подготавливающий возможность исключения сложного выражения с неизвестными. Пример:

   $\forall_{abcdef}(a\sqrt[3]{d}+b\sqrt[3]{e}+c\sqrt[3]{f}=0\ \to\ a^3d+b^3e+c^3f-3abc\sqrt[3]{d}\sqrt[3]{e}\sqrt[3]{f}=0)$

   Здесь подготавливается возможность исключения кубических радикалов.

   (c) конечнпересек(x1). Вывод условия, ограничивающего значения неизвестных списка x1 конечным множеством. Пример:

   $\forall_{abcx}(x-\text{натуральное}\ \&\ ax+b=0\ \&\ -b/x\le c\ \&\ a-\text{натуральное}\ \to\ a\le c)$

   Характеристика - "конечнпересек(a)".

   (d) бинарноеотношение(x1). Вывод двуместного отношения, связывающего неизвестный объект x1 с известными. Пример:

   $\forall_{Afmnpyz}(\text{Max}(f,\{m,\ldots,n\},y,z)\ \&\ f=\lambda_x(p(x),A(x))\ \&\ \forall_i(i\in\{m,\ldots,n\}\ \to\ A(i))\ \to\ y\subseteq\text{set}_i(i\in\{m,\ldots,n\}\ \&\ (i=m\ \vee\ 0\le p(i)-p(i-1))\ \&\ (i=n\ \vee\ 0\le p(i)-p(i+1))))$

   Характеристика - "бинарноеотношение(y)". Теорема ограничивает поиск максимума теми элементами конечного отрезка целых чисел, которые либо являются его концами, либо дают значения, не меньше двух соседних.

3. альтоперанды(x1). Вывод дизъюнкции для разбора случаев при усмотрении подвыражения x1. Пример:

   $\forall_n(n-\text{целое}\ \to\ n-\text{even}\ \vee\ \neg(n-\text{even}))$

   Характеристика - "альтоперанды($(-1)^n$)".

Обратный вывод

1. неизвестная(x1). Теорема представляет собой кванторную импликацию, которая может быть использована для подбора значения неизвестной x1. Пример:

   $\forall_{abcdefhi}(0 \leq 4d^3 + 27a^2 \ \& \ \neg(3bh - i^2 = 0) \ \& \ \neg(h = 0) \ \& \ h -$ число $\& \ i -$ число $\& \ b -$ число $\& \ e -$ число $\& \ f = \sqrt{4d^3 + 27a^2} \ \& \ c = -i/(3h) + (\sqrt[3]{f + 3\sqrt{3}a} + \sqrt[3]{-f + 3\sqrt{3}a})/(\sqrt[3]{2}\sqrt{3}) \ \& \ d = b/h - i^2/(3h^2) \ \& \ a = e/h + i(9bh - 2i^2)/(27h^3) \rightarrow hc^3 + ic^2 + bc = e)$

   Характеристика - "неизвестная(c)". Теорема представляет собой формулу Кардано для случая единственного вещественного корня.

2. попыткаспуска. Теорема представляет собой простую импликацию с бесповторными антецедентами и консеквентом, у которой каждая переменая антецедентов встречается в консеквенте. Пример:

   $\forall_{ab}(0 < a \ \& \ 0 < b \ \& \ a -$ число $\& \ b -$ число $\rightarrow 0 < ab)$

3. подбор(x1 x2). Теорема представляет собой кванторную импликацию, обеспечивающую попытку обратного вывода с переходом к более простым понятиям. x1 - неизвестная либо терм, перечисляющий неизвестные; x2 - набор номеров заменяющих антецедентов. Пример:

   $\forall_f($последовательность$(f, \mathbb{R}) \ \& \ \exists_a(a -$ число $\& \ f = \lambda_n(a, n -$ натуральное$)) \rightarrow$ сходится$(f))$

   Характеристика - "подбор(f 2)".

4. разделить(x1). Обратный вывод для декомпозиции условия. x1 - набор номеров заменяющих антецедентов. Пример:

   $\forall_{ab}(a -$ число $\& \ b -$ число $\& \ 0 < a \ \& \ 0 < b \rightarrow \neg(a + b = 0))$

   Характеристика - "разделить(3 4)".

Проверочные операторы

1. спуск(x1). Теорема представляет собой простую импликацию, которую можно неизбыточным образом использовать в проверочном операторе с заголовком x1, причем все ее антецеденты также обрабатываются проверочными операторами. Пример:

   $\forall_{bef}(b -$ set $\& \ e -$ set $\& \ f -$ set $\& \ $непересек$(b, e) \ \& \ $непересек$(b, f) \rightarrow$ непересек$(b, e \cup f))$

   Характеристика - "спуск(усмнепересек)".

2. легковидеть(x1 x2). Теорема представляет собой простую импликацию, которую можно неизбыточным образом использовать в проверочном операторе с заголовком x1, причем ее x2-й антецедент - непосредственно идентифицируемый, а остальные антецеденты обрабатываются проверочными операторами. Пример:

$\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ a < b \ \& \ b + c \leq 0 \rightarrow a + c < 0)$

Характеристики - "легковидеть(усмменьше 4)" и "легковидеть(усмменьше 5)".

3. Спуск. Теорема представляет собой такую простую импликацию, что конъюнкция ее существенных посылок эквивалентна консеквенту, причем замена консеквента на данную конъюнкцию является преобразованием общей стандартизации либо конъюнктивной декомпозицией. Пример:

   $\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ 0 < a - b \ \& \ 0 < c - a \rightarrow a \in (b, c))$

4. Спуск(x1). Теорема представляет собой такую простую импликацию, что конъюнкция ее существенных посылок, за вычетом существенных посылок с номерами из списка x1, эквивалентна консеквенту, причем замена консеквента на данную конъюнкцию является преобразованием общей стандартизации либо конъюнктивной декомпозицией. Пример:

   $\forall_{abc}(a - \text{set} \ \& \ b - \text{set} \ \& \ c - \text{set} \ \& \ c \subseteq a \ \& \ \neg(\text{непересек}(b, c)) \rightarrow \neg(\text{непересек}(c, a \cap b)))$

   Характеристика - "Спуск(4)".

5. блокпроверок(x1). Теорема представляет собой простую импликацию, которую можно неизбыточным образом использовать в проверочном операторе с заголовком x1, причем более одного ее антецедента будут непосредственно идентифицируемыми. Пример:

   $\forall_{ABCDKa}(D - \text{точка} \ \& \ \text{систкоорд}(K) \ \& \ K = (A, B, C) \ \& \ \text{коорд}(D, K) = (0, a) \rightarrow D \in \text{прямая}(AC))$

   Характеристика - "блокпроверок(усмпринадлежит)".

Усмотрение истинности либо ложности с помощью кванторной имплика-ции

1. стандменьше(x1). Кванторная импликация используется для усмотрения истинности утверждения путем идентификации одного антецедента с посылкой и обработки другого проверочным оператором. x1 - номер антецедента, обрабатываемого проверочным оператором. Пример:

   $\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ 0 < b - c \ \& \ b < a \rightarrow c < a)$

   Характеристика - "стандменьше(4)".

2. квантор. Усмотрение истинности либо ложности кванторной импликации. Пример:

   $\forall_a(\text{убывмножества}(a) \rightarrow \forall_{ij}(i \in \mathbb{N} \ \& \ j \in \mathbb{N} \ \& \ i < j \rightarrow a(j) \subseteq a(i)))$

3. контроль. Кванторная импликация с консеквентом "ложь" используется для усмотрения нереализуемых подслучаев. Пример:

   $\forall_{ab}(a - \text{число} \ \& \ b - \text{число} \ \& \ 0 < b \ \& \ a^2 = -b \rightarrow \text{ложь})$

Координаты

1. видобъекта. Кванторная импликация определяет вид объекта по уравнению для координат его точек. Пример:

   $\forall_{EKabcdefgm}(a-$число $\&\ b-$число $\&\ c-$число $\&\ d-$число $\&\ e-$число $\&\ f-$число $\&\ g-$число $\&$ прямкоорд$(K)\ \&$ коорд$(E,K) = \text{set}_{xyz}(ax^2+by^2+cz^2+dx+ey+fz+g=0\ \&\ x-$число $\&\ y-$число $\&\ z-$число$)\ \&\ m=bcd^2+ace^2+abf^2-4abcg\ \&\ 0<ab\ \&\ 0<ac\ \&\ 0<m \to$ эллипсоид$(E))$

2. характ. Кванторная импликация определяет связи объекта, заданного уравнением для координат его точек, с другими объектами. Пример:

   $\forall_{AEKabcdemn}(a-$число $\&\ b-$число $\&\ c-$число $\&\ d-$число $\&\ e-$число $\&\ A-$точка $\&$ гипербола$(E)\ \&$ коорд$(E,K) = \text{set}_{xy}(ax^2+bx+cy^2+dy+e=0\ \&\ x-$число $\&\ y-$число$)\ \&\ A \in E\ \&$ коорд$(A,K) = (m,n)\ \&\ (2am+b=0 \vee 2cn+d=0) \to$ вершина$(A,E))$

3. Крд. Кванторная импликация, консеквент которой - конъюнкция неравенств либо отрицаний равенств для параметров координат. Пример:

   $\forall_{ABCDKabcdefghp}(A-$точка $\&\ B-$точка $\&$ систкоорд$(K)\ \&\ \neg(A=B)\ \&$ коорд$(A,K) = (a,b)\ \&$ коорд$(B,K) = (c,d)\ \&$ коорд$(C,K) = (e,f)\ \&$ коорд$(D,K) = (g,h)\ \&\ p = ad-bc\ \&$ однасторона$(C,D,$прямая$(AB)) \to 0 \le ((b-d)e+(c-a)f+p)((b-d)g+(c-a)h+p))$

4. координаты. Теорема представляет собой кванторную импликацию с квантором существования в консеквенте, дающим общий вид уравнения для координат множества объектов. Пример:

   $\forall_{EKabcdef}($прямкоорд$(K)\ \&$ парабола$(E) \to \exists_{abcdef}(a-$число $\&\ b-$число $\&\ c-$число $\&\ d-$число $\&\ e-$число $\&\ f-$число $\&$ коорд$(E,K) = \text{set}_{xy}(ax^2+bxy+cy^2+dx+ey+f=0\ \&\ x-$число $\&\ y-$число$)\ \&\ b^2-4ac=0\ \&\ \neg(4acf+bde-ae^2-cd^2-fb^2=0)))$

5. Числатомы. Вывод группы равенств, связывающих невырожденные числовые атомы с параметрами координат. Пример:

   $\forall_{ABCDKa}($прямкоорд$(K)\ \&\ K=(A,B,C)\ \&\ D-$точка $\&\ \neg(D=A)\ \&\ \angle(BAD) = a\ \&$ однасторона$(C,D,$прямая$(AB))\ \&$ коорд$(D,K) = (b,c) \to b = l(AD)\cos a\ \&\ c = l(AD)\sin a)$

6. точкитела. Переход от координатного к бескоординатному описанию множества точек. Пример:

   $\forall_{AKab}(a-$число $\&\ b-$число $\&$ систкоорд$(K)\ \&$ коорд$(A,K) = \text{set}_{yx}(x=a\ \&\ b<y\ \&\ y-$число$) \to \exists_{BC}(B =$ тчкоорд$(K,(b,a))\ \&\ C =$ тчкоорд$(K,(b+1,a))\ \&\ A =$ луч$(BC)))$

19.6 Утверждение без переменных

Утверждение без переменных имеет характеристику "конст". Пример -

$\text{ctg}\,\pi/6 = \sqrt{3}$

19.7 Копия теоремы

Если теорема является копией теоремы по ссылке "x1 - x2", где x1 - логический символ, x2 - номер узла статьи этого символа, то она снабжается характеристикой "копия(x1 x2)". Обычно теорема копируется в тех случаях, когда она получается в одной ячейке вывода, а затем становится заглавной теоремой другой ячейки. Обычно вывод в некоторой ячейке базы теорем обрывается из-за того, что предыстория получения текущей теоремы блокирует дальнейшие применения к ней приемов вывода. В новой ячейке предыстория отсутствует, и получение следствий может быть продолжено на бóльшую глубину.

19.8 Характеризация антецедентов

1. уравндробь. Среди антецедентов теоремы имеется равенство для координат множества объектов. Пример:

   $\forall_{AKabc}(a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ \text{Прямая}(A)\ \&\ \text{систкоорд}(K)\ \&\ \text{коорд}(A,K) = \text{set}_{xy}(ax + by + c = 0\ \&\ x - \text{число}\ \&\ y - \text{число}) \to \neg(a^2 + b^2 = 0))$

2. смпропорц(x1 x2 x3). x1-й антецедент представляет собой соотношение пропорциональности для невырожденных числовых атомов x2 и x3. Пример:

   $\forall_{ABCDab}(a - \text{число}\ \&\ b - \text{число}\ \&\ A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ D - \text{точка}\ \&\ \neg(B = C)\ \&\ \neg(A = B)\ \&\ \angle(ABD) = \angle(DBC)\ \&\ D \in \text{прямая}(AC)\ \&\ al(AB) = bl(AD)\ \&\ \neg(\text{прямая}(AB) = \text{прямая}(BC)) \to al(BC) = bl(CD))$

   Характеристика - "смпропорц(одиннадцать $l(AB)$ $l(AD)$)".

3. главнчлен(x1). Наиболее сложный терм антецедентов - выражение, причем он единственный и имеет единственное вхождение. Глубина этого выражения больше 1, а оценка сложности больше 4. x1 - заголовок данного выражения. Пример:

   $\forall_{abcf}(f - \text{функция}\ \&\ a - \text{число}\ \&\ b - \text{число}\ \&\ (a,b) \subseteq \text{Dom}(f)\ \&\ \text{Val}(f) \subseteq \mathbb{R}\ \&\ 0 < c - a\ \&\ 0 < b - c\ \&\ f(c) = \inf(\text{образ}(f,(a,b)))\ \&\ \text{дифференцируема}(f,c) \to \text{производная}(f,c) = 0)$

   Характеристика - "главнчлен(инф)".

4. тождфунк. Имеется антецедент - равенство, содержащее функциональные переменные, причем каждая такая переменная f входит в него только как $f(\ldots)$. Пример:

   $\forall_{abf}(f - \text{функция}\ \&\ a - \text{число}\ \&\ b - \text{число}\ \&\ a < b\ \&\ [a,b] \subseteq \text{Dom}(f)\ \&\ \text{Val}(f) \subseteq \mathbb{R}\ \&\ \text{непрерывно}(f,[a,b])\ \&\ \forall_c(c \in (a,b) \to \text{дифференцируема}(f,c))\ \&\ f(a) = f(b) \to \exists_c(c \in (a,b)\ \&\ \text{производная}(f,c) = 0))$

19.9 Протоколы

Протоколы определяют различные особенности алгоритмизации того или иного раздела или конкретного понятия. Они оформляются в виде термов специального вида,

сопровожденных характеристикой "протокол". Типы таких термов будут приведены в следующей главе.

Некоторые протоколы представляют собой установки на цикл логического вывода теорем. Такая установка является заглавной "теоремой" ячейки логического вывода. Она может сопровождаться характеристикой "раздел(x1)", уточняющей разде x1, в котором предпринимается вывод.

19.10 Квазипротоколы

Квазипротоколы получаются из протоколов, у которых существенная часть текста представляет собой терм предметной области. Чтобы облегчить восприятие таких протоколов, их терм оформляется как теорема, а оставшаяся часть размещается в виде набора характеристик. Это и называется квазипротоколом. Иногда указанный терм и в самом деле представляет собой что-то вроде теоремы, но все же является лишь теоремой приема. Вывод следствий из него не осуществляется, хотя он (как, впрочем, и "обычные" протоколы) может создаваться процедурой программирующего логического вывода.

1. теоремаприема. Эта характеристика явно указывает на квазипротокол. Она блокирует попытки автоматического пересоздания его характеристик по теореме.

   Пример:

   $\forall_{abf}(b = \sum_{i=a}^{\infty} f(i) = \sum_{i=a}^{\infty} f(i) = b)$

   Квазипротокол представляет собой теорему приема, обращающегося к нормализатору "суммаряда" для вычисления суммы ряда. Дополнительные характеристики уточняют это обстоятельство.

2. Разбор случаев.

   (a) разборслучаев(x1). Импликация для разбора случаев в посылках. x1 - список уточняющих контекст фильтров. Если инициализация разбора случаев осуществляется текущим выражением A, то в начале списка x1 помещается терм "контрольвывода(A)".

   Пример:

   $\forall_{EKabcdefp}$(прямкоорд(K) & коорд(E, K) = $\text{set}_{xy}(ax^2{+}bxy{+}cy^2{+}dx{+}ey{+}f = 0$ & $x -$ число & $y -$ число) & $p = b^2 - 4ac \rightarrow 0 < p \ \vee \ p = 0 \ \vee \ p < 0)$

   Характеристика - "разборслучаев(тип(исследовать)цель(линия) не(контекст(посылка(x7) заголовок(x7 гипербола парабола эллипс) равно(первыйтерм(x7)E))) не(легковидеть($0 < p$)) не(легковидеть($p < 0$)) не(заголовок(p 0)) не(контекст(посылка(x7) вид(x7 равно(E x8)))))". Соответствующий прием инициирует разбор случаев при анализе свойств заданной уравнением линии второго порядка, чтобы различить случаи эллипса, гиперболы и параболы.

(b) Подслучаи(x1). Импликация для разбора случаев в условиях. x1 - список уточняющих контекст фильтров. Если инициализация разбора случаев осуществляется текущим выражением A, то в начале списка x1 помещается терм "контрольвывода(A)". Пример:

$\forall_n(n - \text{целое} \to n - \text{even} \ \vee \ \neg(n - \text{even}))$

Характеристика - "Подслучаи(контрольвывода($(-1)^n$) не(цель(редакция)) не(известно(n)))".

3. Ввод новых объектов.

(a) актив(x1). Импликация, консеквент которой представляет собой конъюнкцию термов вида "актив(...)", вводящая в рассмотрение новые объекты. x1 - список уточняющих контекст фильтров. При отсутствии таких фильтров вместо "актив(x1)" берется характеристика "актив". Если инициализация ввода объектов осуществляется текущим выражением A, то в начале списка x1 помещается терм "контрольвывода(A)". Пример:

$\forall_{ABCPQ}$(окружность(PQ)вписана в фигура(ABC) & актив($\angle(ACB)$) $\to$ актив($l(AB)$))

Характеристика - "актив(неизв(терм($\angle(ACB)$)))".

(b) вспомобъекты(x1). Импликация, вводящая в рассмотрение новые объекты для использования определения некоторого объекта x1. Пример:

$\forall_{ABCDEK}(D-$точка & систкоорд(K) & $K = (A, B, C)$ & $\neg(D \in$ прямая(AC)) $\to E-$точка & $\neg(D = E)$ & $E \in$ прямая(AC) & прямая(DE) $\parallel$ прямая(AB))

Характеристика - "вспомобъекты(коорд(D, K))".

(c) прямкоорд(x1). Импликация вводит в рассмотрение систему координат. x1 - список уточняющих контекст фильтров. Если инициализация осуществляется текущим выражением A, то в начале списка x1 помещается терм "контрольвывода(A)". Если фильтр B вида "контекст(...)" переброшен в указатели приема, то он регистрируется в списке x1 как "указатель(B)". Пример:

$\forall_{ABCDEFGKa}$(куб(a) & ребро(отрезок(AB), a) & ребро(отрезок(AC), a) & ребро(отрезок(AD), a) & разныеточки(B, C) & разныеточки(B, D) & разныеточки(C, D) $\to E -$ точка & $F -$ точка & $G -$ точка & $K = (A, E, F, G)$ & прямкоорд(K) & $E \in$ прямая(AB) & $F \in$ прямая(AC) & $G \in$ прямая(AD) & точкалуча(A, B, E) & точкалуча(A, C, F) & точкалуча(A, D, G))

Характеристика - "прямкоорд(не(Входит(прямкоорд списокпосылок)) контекст(посылка(x5) вид(x5 актив(вектор(AX))))не(равно(B C)) не(равно(B D)) не(равно(C D)))".

(d) координлоск(x1). Импликация вводит в рассмотрение координаты множества объектов. x1 - список уточняющих контекст фильтров. Если инициализация осуществляется текущим выражением A, то в начале списка x1 помещается терм "контрольвывода(A)". При пустом списке x1 вместо терма "координлоск(x1)" берется логический символ "координлоск". Если фильтр

B вида "контекст(...)" переброшен в указатели, то он регистрируется в списке x1 как "указатель(B)". Пример:

$\forall_{ABEK}$(асимптота(прямая(AB), E) $\rightarrow$ актив(коорд(прямая(AB), K)))

Характеристика - "координплоск(контрольвывода(коорд(E, K)))".

(e) параметры(x1). Импликация, вводящая в рассмотрение координатный набор, выраженный через новые параметры. x1 - список уточняющих контекст фильтров. Если инициализация осуществляется текущим выражением A, то в начале списка x1 помещается терм "контрольвывода(A)". При пустом списке x1 вместо терма "параметры(x1)" берется логический символ "параметры". Если фильтр B вида "контекст(...)" переброшен в указатели, то он регистрируется в списке x1 как "указатель(B)". Пример:

$\forall_{ABCDEKPQabcdef}(C \in$ прямая(AB) & $C \in$ прямая(DE) & $P \in$ прямая(AB) & коорд(P, K) = (a, b) & $Q \in$ прямая(DE) & коорд(Q, K) = $(c, d) \rightarrow$ $e -$ число & $f -$ число & коорд(C, K) = (e, f))

Характеристика - "параметры(контрольвывода(коорд(прямая(AB),K)) указатель(контекст(посылка(x7) позиция(x8 x7) вид(x8 коорд(прямая(CD),K)))) не(равно(терм(прямая(AB)) терм(прямая(DE)))))". Вводится в рассмотрение координатный набор для общей точки двух прямых, уравнения которых упоминаются в задаче, если на каждой из прямых имеется точка с уже введенным в рассмотрение координатным набором.

4. Упрощение термов.

(a) связьпеременная(x1). Тождество для упрощения выражения относительно переменных связывающей приставки. x1 - направление замены. Пример:

$\forall_{abcdpq}(ap + b = 0$ & $\neg(a = 0)$ & $q = ad - bc \rightarrow cp + d = q/a)$

Характеристика - "связьпеременная(второйтерм)". Соответствующий прием проверяет, что преобразуемое вхождение подчинено квантору существования, причем p идентифицируется с произведение всех сомножителей, содержащих переменные связывающей приставки. Оно нелинейно относительно этих переменных. Первый антецедент идентифицируется с утверждением из контекста. Оно тоже располагается под указанным квантором. Выражение a не содержит переменных связывающей приставки. Последний антецедент выделен указателем "идентификатор". Его правая часть обрабатывается нормализатором разложения на множители. Результат q является линейным относительно переменных связывающей приставки.

Глава 20

Протоколы базы теорем

Для создания приема, помимо теоремы, бывают нужны также некоторые общие характеристики той предметной области, к которой теорема относится. Такие характеристики размещаются в базе теорем и называются протоколами базы теорем. Иногда протокол создается, чтобы фиксировать выбор некоторой версии создания приемов по теоремам данной предметной области, иногда - сам может стать источником приемов. Некоторые протоколы служат для организации какого-либо цикла логического вывода, порождающего новые теоремы. Во всех этих случаях протокол представляет собой терм чисто технического характера. Он вводится через тот же интерфейс, что и обычная теорема, и всегда снабжается характеристикой "протокол". В отдельных случаях допускаются дополнительные характеристики. Как и теоремы, протоколы могут создаваться приемами программирующего логического вывода. Эти приемы описываются в следующем томе монографии. Совокупность протоколов, относящихся к предметной области, может рассматриваться как некоторая схема алгоритмизации данной предметной области. По этой причине та часть системы вывода теорем, которая создает протоколы, получила название алгоритмизатора. На текущий момент этот модуль, как и сама система протоколов, находится лишь в зачаточном состоянии.

В этой главе мы приведем краткое описание типов протоколов, используемых системой.

20.1 Общие свойства понятий

1. степень(x1 x2 x3). Протокол указывает, что для ассоциативной операции x1 введена двуместная операция x2 типа "степени". x3 - номер операнда, соответствующего показателю степени. Пример: "степень(умножение степень 2)".

2. актив(x1). Протокол указывает на режим ввода посылки "актив(x1)" при обнаружении в задаче выражения x1. Пример: "актив(прямая(ab))".

3. развертка(x1 x2). Протокол указывает, что для двуместной ассоциативно - коммутативной операции x1 введена соответствующа яоперация x2 над семейством операндов. Пример: "развертка(суммавсех плюс)".

4. опред(x1 x2 x3). Протокол указывает, что при выполнении условия x1 значение объекта x3 однозначно определяется значением объекта x2. Здесь x2, x3 - переменные, x1 - утверждение с этими переменными. Пример: "опред(Отображение(x1 x2 x3)x1 x2)".

5. тождвывод(x1 x2). Протокол указывает, что для операции x1 создается прием, усматривающий ее значение из кванторного тождества, имеющегося в контексте. x2 - список символов, которые не должны встречаться в выражении, определяющем значением операции. Пример: "тождвывод(услвероятн услвероятн вероятность)".

6. размерность(x1 x2). x1 есть операция над x2-параметрическим семейством. Пример: "размерность(двойнойинтеграл 2)".

7. коммутативно(x1). x1 - коммутативный символ. Имеютя в виду как операции, так и симметричные отношения. Пример: "коммутативно(параллельны)".

8. одз(x1 x2). Утверждение x2 определяет о.д.з. для терма x1. Пример: "одз(прямая(AB)и(точка(A) точка(B)не(равно(A B))))".

9. ограничитель(x1 x2). x1 - символ одноместной операции. x2 - утверждение с единственной переменной x, интерпретируемой как операнд x1, указывающее некоторые дополнительные ограничения, достаточно часто выполняющиеся при использовании операции x1. Пример: "ограничитель(тангенс и(меньшеилиравно(0 x)меньшеилиравно(x пи)))". Величины углов в планиметрии располагаются в данном диапазоне, и для него имеет смысл создавать специальные приемы.

10. схемаоперандов(x1 x2). x1 - символ, обладающий симметрией, определяемой термом x2. Пример: "схемаоперандов(биссектриса набор(0 2 4 набор(1 1 3)))". Протокол указывает, что в терме "биссектриса($ABCD$)" второй и четвертый операнды зафиксированы на своих позициях, а первый и третий можно переставлять.

11. сокращтеор(x1 x2). Операция x1 введена для сокращенной записи выражений, содержащих операцию x2. Пример: "сокращтеор(услвероятн вероятность)".

12. блокраздела(x1 x2). Хотя операция x1 и определяется через более простые операции списка x2, но эти операции относятся к разным кластерам приемов, и сведение x1 к x2 без особых к тому причин не допускается. Пример: "блокраздела(скалумнож угол уголмежду)". Работа со скалярным произведением чаще осуществляется в координатах, а не в геометрических понятиях.

13. функция(x1 x2). Для термов x2 с заголовком "отображение" предусмотрена специальная идентификация, определяемая указателем "x1(v)", где v - вхождение терма x2 в теорему. Пример: "функция(матрица отображение(x1 x2 и(принадлежит(x1 номера(1 x3)) принадлежит(x2 номера(1 x4))) значение(x5 набор(x1 x2))))". Для идентификации определяющих прямоугольную матрицу функций от двух переменных используется указатель "матрица(v)".

14. логсимвол(x1). Протокол, инициирующий рассмотрение всех связанных с символом x1 теорем для общей алгоритмизации этого символа. Пример: "логсимвол(умножение)". Запуск процедуры логического вывода на этом протоколе, просмотрев имеющиеся теоремы, создаст протоколы для инициализации пакетных нормализаторов "нормумножение" и "упрощумножение".

15. раздел(x1). Протокол, инициирующий рассмотрение всех теорем раздела x1 для общей алгоритмизации этого раздела. Пример: "раздел(элементарнаяалгебра)". Запуск процедуры логического вывода на этом протоколе приведет к созданию нескольких протоколов "унификация". Например, будет создан протокол "унификация(секанс косеканс синус косинус тангенс котангенс тригаргумент)". Он указывает на целесообразность приведения соответствующих операций к общему аргументу. При этом "тригаргумент" - название идентифицирующего терма ГЕНОЛОГа, перечисляющего аргументы данных операций в текущем терме задачи.

16. сравн(x1 x2). Для заголовка x1 числового атома целесообразно создание приемов, усматривающих строгие либо нестрогие неравенства с этим атомом в одной части и константой x2 в другой. Пример: "сравн(угол $\pi/2$)".

17. родобъекта(x1). x1 - название одного из основных типов объектов. Пример: "родобъекта(натуральное)".

18. Новый(x1). x1 - понятие, для которого целесообразно создание приема расшифровки по определению. Пример: "Новый(гипсинус)".

19. взаимодействие(x1 x2). x2 - список символов операций, с которыми предположительно может "взаимодействовать" операция x1. В него не включаются операции, определяемые через x1, однако может включаться сама операция x1. Примеры: "взаимодействие(синус синус косинус)", "взаимодействие(факториал факториал)".

20.2 Нормализаторы

Нормализатор общей стандартизации

1. нормализация(x1 x2). x2 - название нормализатора общей стандартизации выражений либо утверждений с заголовком x1. Для нормализатора общей стандартизации равенств значением x1 служит тип объектов, соединяемых равенством. Пример: "нормализация(плюс нормплюс)".

2. общнорм(x1 x2). x1 - выражение $f(y_1...y_m)$, x2 - условия на контекст, при которых обработка выражения x1 нормализатором общей стандартизации обеспечивает вычисление значения операции f. Пример: "общнорм(прямоепроизведение(x1 x2) заголовок(x1 перечень) первыйсимвол(x1 набор) заголовок(x2 перечень) первыйсимвол(x2 набор))". Нормализатор общей стандартизации преобразует произведение двух конечных списков в список пар.

3. перем(x1). x1 - название нормализатора общей стандартизации выражений с заголовком A. Для этого нормализатора предусмотрен прием, использующий посылку вида "$A(\ldots) = X$" для замены текущего выражения на переменную X. Пример: "перем(нормпромежуток)". Если в задаче по физике временной промежуток обозначен переменной, то эта переменная используется везде для ссылки на данный промежуток.

4. вычмн(x1 x2 x3). Для константных термов с заголовком x1 нормализатор общей стандартизации обеспечивает вычисление их значения, имеющего тип x2,

если x3 - набор типов константных операндов. Пример: "вычмн(плюс десчисло десчисло десчисло)".

Нормализатор сокращенной перезаписи

1. нормупростить(x1 x2). x2 - название нормализатора сокращенной перезаписи выражений с заголовком x1. Пример: "нормупростить(умножение упрощумножение)".

Нормализатор приведения к заданным заголовкам

1. нормзаголовок($g_1 \ldots g_n f$). Протокол определяет название f нормализатора, преобразующего термы к одному из заголовков $g_1, \ldots, g_n$. Пример: "нормзаголовок(умножение дробь степень видумножение)".

2. нормразделение(x1 x2 x3). x1 - вид утверждения либо выражения, которое с помощью нормализатора приведения к заданным заголовкам x3, применяемого к идентифицируемому с переменной x2 выражению, преобразуется к виду, допускающему декомпозицию. Пример: "нормразделение(непересек(x1 x2) x2 видобъединение)".

3. блокнеизвестных(x1 x2 x3). x1 - шаблон выражения, обрабатываемого нормализатором x2 приведения к заданным заголовкам, для которого блокируются дальнейшие преобразования ввиду того, что текущий результат достаточен для внешних целей. x3 - фильтр, уточняющий контекст. Пример: "блокнеизвестных($a \sin x \cos x + b \sin x + b \cos x + c$)) видумножение и(не(известно(x))известно(a)известно(b)известно(c)))". В указанной ситуации легко усматривается квадратное уравнение относительно суммы синуса и косинуса.

4. видобъекта(x1 x2 x3). x1 - название нормализатора приведения к заданным заголовкам, x2 - один из этих заголовков, x3 - список фильтров "контекст(вид(A ...))", определяющих условия на вид входного терма A, при котором можно ожидать результативной работы нормализатора с получением заголовка x2. В этом списке в качестве A используется служебный символ "фикс". Возможно отсутствие x2. Тогда x3 определяет условия, при которых можно ожидать получение какого-либо из заданных заголовков нормализатора. Пример: "видобъекта(видумножение дробь контекст(вид(фикс $a + b/c$) заменазнака(минус b)))". В данном случае сложение дробных выражений обычно дает дробное выражение.

5. блокир(x1 x2 x3). Остановка попыток приведения к заданным заголовкам терма x1 нормализатором x2. x3 - конъюнкция фильтров. Пример: "блокир(плюс(умножение(x1 x2)x3) видумножение и(не(контекст(вид(x1 степень(x4 2)))) не(контекст(вид(x1 степень(x4 3)))) не(контекст(вид(x1 степень(x4 5)))) контекст(вид(x1 степень(x4 x5)) единица(1 x5) не(контекст(вид(x3 плюс(x6 умножение(x7 степень(x4 x8)))) единица(1 x7 x8) заменазнака(минус x7) единица(0 x6)))) не(контекст(тригаргумент(корень x4))) не(контекст(операнд(теквхожд x4) обобщмножитель(x4 x5) символ(x5 плюс))) не(контекст(обобщмножитель(x2 x4) символ(x4 плюс))) не(контекст(обобщмножитель(x1 x4) символ(x4 плюс))) не(контекст(вид(x1 степень(x5 x6)) десчисло(x5) единица(1 x6))) не(контекст(вид(теквхожд плюс(дробь(x5 x6) x7)) заменазнака(минус x5))) коммент(константа) коммент(

редуцирование)))". Собраны эвристические условия, заставляющие подозревать, что попытка разложения на множители выражения "плюс(умножение(x1 x2)x3)" окажется неудачной.

6. заголовок(x1 x2 x3). Нормализатор x2 приведения к заданным заголовкам применяется в задачах на преобразование с целью x3 к условию, представимому в виде терма x1. Пример: "заголовок(плюс(x1 x2) видумножение разложитьнамножители)".

Нормализатор стандартной формы

1. станддн(x1 x2 x3). x1 - название нормализатора стандартной формы, имеющей корневую ассоциативно-коммутативную операцию x2 и внутреннюю ассоциативно-коммутативную операцию x3. Пример - "станддн(стандплюс плюс умножение)".

2. стандартизация(x1 x2). x1- название нормализатора стандартной формы, x2 - список логических символов, для которых предусмотрена инициализация попытки обращения к данному нормализатору. Пример - "стандартизация(стандобъединение объединение пересечение разность прямоепроизведение прообраз образ)".

3. стандтерм(x1 x2 x3). Указание на контекст, в котором выполняется стандартизации с помощью нормализатора стандартной формы. x2 - преобразуемый терм, x1 - надтерм терма x2, в рамках которого выполняется стандартизация, x3 - заголовок нормализатора. Пример: "стандтерм(рациональное($a(b+c)^d$) $a(b+c)^d$ стандплюс)".

4. свертка(x1 x2). x2 - название специального нормализатора свертки выражений, приведенных к виду стандартной формы x1. Пример - "свертка(станддн двгруппировки)". Нормализатор "двгруппировки" предпринимает попытки булевой "факторизации" для уменьшения длины формулы алгебры логики.

Нормализатор упрощения относительно неизвестных

1. упрощнеизв(x1 x2). x2 - название нормализатора упрощения относительно неизвестных выражений с заголовком x1. Пример: "упрощнеизв(логарифм уравнлогарифм)".

2. нормнеизв(x1 x2 x3). x1 - название нормализатора, выполняющего разрешение утверждений вида x2 относительно неизвестных, указанных термом x3 вида "неизвестные(...)". Пример: "нормнеизв(нормМаксимум Максимум(x1 x2 x3 x4) неизвестные(x3 x4))".

Нормализатор вычисления

1. вычисление(x1). x1 - название нормализатора вычисления x1. Пример: "вычисление(собствзначения)".

2. симв(x1 x2 x3 x4). Для нормализатора x1, применяемого к терму вида x2, предпочтительно преобразование переменной x3 к заголовку, принадлежащему списку символов x4. Пример: "симв(нормнижняягрань нижняягрань(x1 образ(x2 x3))x3 перечень класс пусто)".

3. числпарам(x1 x2). x1 есть название нормализатора вычисления, предназначенного для выражения термов вида x2 через известные параметры и неизвестные внешней задачи на описание, рассматриваемой относительно текущей задачи на исследование. Пример: "числпарам(смпериод длина(промежуток(x1 x2 1 1)))".

4. нормвыч(x1 x2 x3). x1 есть название нормализатора вычисления, который следует применять к подвыражению условия x2 задачи на описание, идентифицированному с переменной x3. Пример: "нормвыч(нормквадрформа ортканоничвид(x6 x23 x24) x6)".

Нормализатор выделения заданных подтермов

1. извлечфунк(x1 x2). x1 - название нормализатора извлечения заданного подтерма, заголовок которого принадлежит списку символов x2. Пример: "извлечфунк(извлечение плюс умножение степень дробь синус косинус логарифм тангенс)".

Нормализатор упрощения дизъюнкции

1. нормили(x1 x2). x1 - название нормализатора упрощения дизъюнкций, в которых встречаются предикаты списка x2. Пример: "нормили(склейканеравенств меньше меньшеилиравно)".

Специальная стандартизация

Протоколы этого раздела имеют предварительный характер.

1. станд(x1). Применение последовательности нормализаторов списка x1 для специальной стандартизации. Пример: "станд(стандменьше)".

2. пересечениесерий(x1 x2 x3). Для объединения двух параметрических описаний неизвестной, вид которых определяется конъюнкцией x1, используется нормализатор x2. x3 - неизвестная. Пример: "пересечениесерий($\exists_n(n-$целое & $f(n) \leq x$ & $x \leq g(n))$ & $\exists_m(m-$целое & $p(m) \leq x$ & $x \leq q(m))$ пересечениесерий x)".

3. стандменьше(x1 x2). x2 - заголовок нормализатора стандартизации условий с известными параметрами для отношений, имеющих заголовок x1. Пример: "стандменьше(менше стандменьше)".

4. спускоперандов(x1 x2). В нормализаторе x1 предусмотрены приемы устранения вложенных операций списка x2. Пример: "спускоперандов(Нормммногочлен плюс)".

5. смвывод(x1 x2). Нормализатор x1 переводит утверждение, заголовок которого принадлежит списку x2, в ослабленное либо (при наличии комментария "доказать") усиленное утверждение, в котором исключена сложная операция. Пример: "смвывод(исклцелаячасть меньше меньшеилиравно)".

20.3 Синтезаторы

1. синтезатор(x1 x2 вход(x3) выход(x4)x5). x1 - название синтезатора, реализующего утверждение x2; x3 - список входных переменных; x4 - список выходных переменных. x5 - список термов, уточняющих тип синтезатора. Пример: "синтезатор(компсвязности компсвязности(x1 x2) вход(x1) выход(x2) перечисление)". Синтезатор перечисляет компоненты связности множества точек евклидова пространства.

20.4 Вспомогательные задачи

1. конст(x1 x2). x1 определяет вид константного подвыражения посылки, для которого целесообразна попытка упрощения с помощью вспомогательной задачи. x2 - конъюнкция фильтров, определяющих некоторое условие на x1, которое после обращения к задаче должно оказаться ложным. Пример: "конст(синус(x1) контекст(подчинено(x2 теквхожд) символ(x2 арксинус арктангенс арккотангенс арккосинус)))".

2. вспомописание(x1 неизвестные(x2 x3)). Указание на создание приема, обращающегося к задаче на описание для разрешения условия x1, неизвестными которого служат только переменные x2 и x3, относительно x2. Пример: "вспомописание($ab^cd^e = f$, неизвестные(b, d))".

3. вспомнеизв(x1 неизвестные(x2)). x1 - конъюнкция условий задачи на описание, которые целесообразно пытаться разрешить относительно неизвестных подвыражений x2. Пример: "вспомнеизв($ax + by + cz + d = e$ & $fx + gy + hz + p = q$ & $rx + sy + tz + u = v$, неизвестные(x, y, z))".

4. упрощение(x1 x2 x3). Целесообразность попытки упрощения терма x2, являющегося подтермом терма x1, с помощью вспомогательной задачи на преобразование. x3 - конъюнкция фильтров. Пример: "упрощение(set$_{xy}$($ax + by + c = 0$ & x – число & y – число),c, контекст(позиция(x8 c)вид(x8 $(e/f) + g$) заменазнака(минус e)))".

5. преобразовать(x1). Для выражений вида x1 целесообразна попытка вычисления их значения с помощью вспомогательной задачи на преобразование, имеющей цель "упростить". Пример: "преобразовать(объем(A))".

6. решение(x1 x2). Уравнение вида x1 в посылках задачи на исследование целесообразно разрешать относительно неизвестной x2. Пример: "решение($ab + c = d$, a)". С протоколом связан прием, разрешающий посылку задачи на исследование относительно той неизвестной, по которой она линейна. Применение приема сопровождается множеством дополнительных ограничений.

7. допконтекст(x1 x2 x3). Доказательство утверждений вида x1 при истинности фильтров x2 целесообразно передавать вспомогательной задаче на доказательство, сопровождаемой комментарием x3. Пример: "допконтекст(меньше(x1 x2) контекст(входит(переменная(x3) корень) переменная(x3)) комментарий(1 нижняягрань x3))". С протоколом связан прием, выбирающий переменную, по которой будет предприниматься дифференцирование, и обращающийся к задаче на доказательство неравенства с помощью производных.

20.5 Вид ответа

1. группнеизв(x1 x2). x1 - список шаблонов утверждений с единственной неизвестной y, для которых возможна свертка их в ответ относительно y. x2 - название раздела, к которому относятся данные утверждения. Роль y в протоколе играет переменная с номером 1. Пример: "группнеизв(принадлежит(x1 x2) не(принадлежит(x1 x2)) не(равно(x1 x2)) теориямножеств)".

20.6 Параметризация

1. параметризация(P). Протокол указывает на необходимость ввода параметров типа P при работе с условиями вида $P(A)$. Пример: "параметризация(целое)".

2. парамописание(P x). Протокол указывает на целесообразность попытки перехода от условия P задачи на описание, в котором переменная x идентифицируется с невырожденным неизвестным подвыражением, к параметрическому описанию относительно вспомогательного параметра y, путем разрешения утверждения "$P(y)$ & $x = y$" относительно некоторой входящей в x неизвестной. Пример: "парамописание(целое(x1) x1)". На данном протоколе создан прием, преобразующий в параметрическое описание условие задачи на описание вида "t – целое", где выражение t содержит некоторую неизвестную y. Выбирается новая целочисленная переменная n, уравнение $t = n$ разрешается относительно y, и по найденному явному выражению для y через n формируется параметрическое описание данной неизвестной с варьируемом параметром n.

20.7 Стандартизация

1. стандподст(x $f(\ldots x \ldots)$ A). Операнд x операции f является стандартизируемым в контексте, определяемом фильтром A. При решении задачи реализуется тенденция к отождествлению таких операндов для операций f. Пример: "стандподст(x1 степень(x1 x2) и(тип(описать) условие не(известно(x2))))". Основания степеней с неизвестными показателями при решении уравнений целесообразно пытаться сделать одинаковыми. Другой пример - "стандподст(x1 логарифм(x1 x2) истина)".

2. блокзамен(x1 x2). x1 - вид термов, появление которых в контексте x2 запрещается. x2 - конъюнкция фильтров и указателей идентификации. Пример: "блокзамен(tg($a - 2b\pi/c$) и(натуральное(b)натуральное(c) единица(1 b c)))". Протокол учитывается при выводе теорем. Он заставляет избавляться от минуса под тангенсом с помощью формул приведения.

3. унификация(x1 x2). Список x1 перечисляет такие одноместные операции, что при решении задач на описание целесообразно сведение их к общему аргументу (унификация аргументов). x2 - название идентифицирующего терма ГЕНОЛОГа, перечисляющего аргументы операций списка x1 в заданном терме. Характеристика "символ(A)", если она есть, указывает одноместную операцию A, допускающую вынесение из-под операций списка x1. Пример: "унификация(секанс косеканс синус косинус тангенс котангенс тригаргумент)".

4. числзнач(x1). x1 - числовой атом, для которого предусмотрен прием, усматривающий в контексте равенство, выражающее этот атом через численные параметры, и реализующий замену. Пример: "числзнач(вероятность(A B))".

5. подобныечлены(x1 x2). x1 - числовой атом, для которого предусмотрены специальные приемы приведения подобных членов. x2 - либо терм "не($A_1 \ldots A_n$)", и тогда коэффициентами при x1 считаются произвольные выражения, не содержащие символов $A_1, \ldots, A_n$ либо набор символов $A_1, \ldots, A_n$, и тогда коэффициентами считаются произвольные выражения, не содержащие невырожденных числовых атомов, заголовки которых отличны от $A_1, \ldots, A_n$. Если x2 отсутствует, то коэффициентами считаются произвольные выражения, не имеющие невырожденных числовых атомов. Пример: "подобныечлены(расстояние(x1 x2) не(расстояние длина))".

6. группировка(x1). x1 - числовой атом, для которого предусмотрен прием догруппировки (если коэффициент при x1 имеет вид суммы, то с ней группируются прочие члены с сомножителем x1). Пример: "группировка(расстояние(x1 x2))".

7. раскрытьскобки(x1). x1 - числовой атом, для которого предусмотрены приемы раскрывания скобок. Пример: "раскрытьскобки(расстояние(x1 x2))".

8. исключ(x1 x2). Предпринимается ориентация равенств, направленная на исключение выражений с заголовком x1. x2 - элементы спецификации приема. Пример: "исключ(область см(условие тип(описать) корень))". В условиях задач на описание области отображений заменяются на их явные выражения. Для этого выражение с заголовком "область" перемещается в левую часть равенства.

9. использ(x1 x2). Предпринимается ориентация равенств, направленная на переход к использованию логического символа x1. x2 - элементы спецификации приема. Пример: "использ(отрезок см(тип(исследовать) корень или(не(цель(текстоваязадача)) комментусловия(ориентацияравенства))) переменная)". В посылках задач на исследование переменные, обозначающие отрезки, заменяются на явные задания отрезков. Для этого выражение с заголовком "отрезок" перемещается в правую часть равенства.

10. едн(x1 x2 x3). Существует стандартизация утверждений с неизвестными, шаблон которых задается термом x1, делающая избыточной переменную x3, так что вместо нее в теоремах приемов следует подставлять единицу. x2 - список переменных, хотя бы одна из которых должна идентифицироваться с содержащим неизвестные выражением. Пример: "едн($a/b + c = d$, a, b)". Протокол используется при создании теорем приемов.

11. существпосылки(x1). Посылки с заголовком x1 являются часто используемыми, и развертка их по определению нецелесообразна. Пример: "существпосылки(описана)". Протокол используется при создании приемов.

20.8 Разрешение относительно заданного терма

1. нормуравн(x1). Для числового атома x1 предусмотрены приемы разрешения относительно x1 простейших уравнений. Пример: "нормуравн(угол(x1 x2 x3))".

2. консеквент(x1). Если x1 - заголовок неизвестного консеквента кванторной импликации в условиях задачи на описание, имеющей известные антецеденты, то целесообразно разрешить этот консеквент относительно единственной неизвестной. Пример: "консеквент(меньшеилиравно)". После разрешения неравенства относительно неизвестной появляется возможнотсь избавиться от квантора, переходя к рассмотрению известных точных верхней либо нижней граней.

3. извлечение($A(\ldots t(h(x))\ldots), x, t, P$). Для условий задачи на описание, имеющих вид A, предпринимается попытка усмотрения того, что вся неизвестная их часть выражается через заданный терм $h(x)$, содержащий неизвестную x. Для $h(x)$ вводится вспомогательная неизвестная y, условие разрешается относительно y, и в результат обратно подставляется $h(x)$. t - вспомогательная функциональная переменная внутри шаблона A, P - дополнительные условия на вспомогательную неизвестную, которая здесь обозначена той же буквой x, что и исходная неизвестная. Пример: "извлечение($a < f(\sin h(x) + \cos h(x))$, x, f, $x - mbox$)". Соответствующий прием усматривает возможность замены суммы синуса и косинуса на новую неизвестную.

4. найдено(x1 x2 x3). Утверждение x1 рассматривается как определяющее неизвестную x2 через ранее найденные объекты, если истинна конъюнкция условий x3. Пример: "найдено($D \in$ окружность(AB)$\cap$прямая(AC), D, $A-$точка & $B-$точка & $C-$точка & $D-$точка & $\neg(A = C)$ & $\neg(A = B)$ & $D \in$ окружность(AB) & $A \in$ отрезок(CD))".

20.9 Вывод в условиях

1. коррекцияодз(x1 x2 x3). Коррекция о.д.з. путем вывода утверждения в условиях задачи на описание. x1 - вид терма, x2 - выводимое сопровождающее утверждение, x3 - конъюнкция фильтров, уточняющих контекст. Пример: "коррекцияодз(дробь(x2 x1) не(равно(x1 0)) истина)". Соответствующий прием проверяет наличие комментария "коррекцияодз". Если затем не усматривается, что знаменатель дробь ненулевой, то создается дополнительное условие, явно на это указывающее.

20.10 Вывод в посылках

1. стандтеор(x1). Указание на целесообразность вывода промежуточных утверждений, вид которых задается шаблоном x1. Такие утверждения часто используются в антецедентах теорем приемов. Пример: "стандтеор(не(принадлежит(x1 интервал(x2 x3))))". В таком виде обычно формулируются условия принадлежности точки x3 лучу x1-x2.

20.11 Разбор случаев

1. подслучаи(x1 x2 x3). Указание на целесообразность разбора случаев по дизъюнкции x2 для достижения цели x1. x3 - список утверждений, дополняющих контекст. В качестве x1 используются следующие термы: "упрощение(t)" - упрощение выражения t, "числатомы($B_1 \ldots B_n$) вывод соотношения, связывающего

числовые атомы $B_1, \ldots, B_n$. Примеры: "подслучаи(упрощение($(a/b)^c$), $0 < b \vee b < 0$)", "подслучаи(числатомы($\angle(BAC), \angle(CAD), \angle(BAD)$), $\in$ отрезок(BD) $\vee$ $B \in$ отрезок(CD) $\vee$ $D \in$ отрезок(BC), $C \in$ прямая(BD), $A-$точка, $B-$точка$C-$точка$D-$точка)". В последнем протоколе разбор случаев позволяет установить, нужно ли брать сумму либо разность углов для получения третьего угла.

20.12 Пакеты продукций

1. циклзнач(x1 x2). x1 - название пакета продукций, использующего цикл. x2 - список термов следующих видов: "условие(...)" - утверждения, определяющие связь между входными и выходными параметрами оператора, "посылки(...)" - утверждения, определяющие условия на входные данные оператора, "вход(...)" - список входных параметров оператора, "выход(...)" - список выходных параметров оператора, "параметры(...)" - список вспомогательных параметров оператора, "контекст(...)" - утверждения, определяющие связь вспомогательных параметров с входными и выходными параметрами, "индикатор(...)" - утверждения, используемые продукциями, истинность которых устанавливается продукциями же. Для таких утверждений в программе вводятся специальные переменные - индикаторы истинности. При работе продукций значения входных переменных изменяются так, что из истинности утверждений "условие(...)" для текущих значений входных переменных и некоторых значений выходных вытекает их истинность для исходных значений входных переменных и тех же значений выходных. Пример: "циклзнач(линсист условие(равно(умножматр(x1 x23)x2) матр(x23 промежуток(минусбеск плюсбеск 0 0)x14 1)) посылки(матр(x1 промежуток(минусбеск плюсбеск 0 0)x14 x14)натуральное(x14)матр(x2 промежуток(минусбеск плюсбеск 0 0)x14 1)) вход(x1 x2) выход(x23) параметры(x13) контекст(натуральное(x13)вертикнули(x1 x13)) индикатор(диагмакс(x1 x13)))". Пакет продукций "линсист" решает матричное уравнение $ax = b$, где a - вещественная квадратная матрица размера $n \times n$, b и x - вещественные столбцы высоты n. Вспомогательный параметр m указывает число "пройденных" при диагонализации столбцов; утверждение "вертикнули(a, m)" означает, что матрица a имеет нули под главной диагональю в столбцах, номера которых меньше m. Утверждение "диагмакс(a, m)" означает, что максимальное значение модуля элементов матрицы a, расположенных в столбце с номером m не выше главной диагонали, достигается для ее диагонального элемента.

2. циклпарам(x1 x2). Аналогично предыдущему. Отличие лишь в том, что на каждом шаге истинность утверждений "условие(...)" для исходных значений входных переменных и некоторых значений выходных должна вытекать из истинности утверждений "контекст(...)" для текущих значений, а не из утверждений "условие(...)". Пример - "циклпарам(обрматрица условие(равно(x2 степеньматр(x1 минус(1)))) посылки(матр(x1 промежуток(минусбеск плюсбеск 0 0) x14 x14) натуральное(x14)) вход(x1) выход(x2) параметры(x3 x4 x13) контекст(равно(умножматр(x3 x2) x4) матр(x3 промежуток(минусбеск плюсбеск 0 0)x14 x14) матр(x4 промежуток(минусбеск плюсбеск 0 0)x14 x14) натуральное(x13) вертикнули(x3 x13)) индикатор(диагмакс(x3 x13)))".

3. значения(x1 x2). x1 - название пакета продукций, не использующего цикл. x2 - список термов следующих видов: "условие(...)" - утверждения, определяющие

связь между входными и выходными параметрами оператора, "посылки(...)" - утверждения, определяющие условия на входные данные оператора, "вход(...)" - список входных параметров оператора, "выход(...)" - список выходных параметров оператора. Пример: "значения(числкоэфф условие(равно(x3 числкоэфф(x1 x2))) посылки(число(x1) натуральное(x13) натуральное(x14) матр(x2 промежуток(минусбеск плюсбеск 0 0) x14 x13)) вход(x1 x2) выход(x3))".

4. повтор(x1 x2). x1 - название пакета продукций, использующего цикл, реализуемый до выполнения заданного условия. x2 - список термов следующих видов: "условие(...)" - утверждения, определяющие связь между входными и выходными параметрами оператора, "посылки(...)" - утверждения, определяющие условия на входные данные оператора, "вход(...)" - список входных параметров оператора, "выход(...)" - список выходных параметров оператора. Пример: "повтор(экспматр условие(равно(x3 точность(экспматр(x1)x2))) посылки(число(x2) матр(x1 промежуток(минусбеск плюсбеск 0 0) x14 x14) натуральное(x14)) вход(x1 x2) выход(x3))". Здесь x1 - квадратная вещественная матрица, для которой требуется вычислить экспоненту с точностью x2.

20.13 Вычисления

1. вычпрог($f(x_1 \dots x_n)P_1(x_1) \dots P_n(x_n)$). f - символ операции; $x_1, \dots, x_n$ - переменные; $P_1, \dots, P_n$ - типы значений этих переменных, для которых существует программа, вычисляющая значение операции f. Пример: "вычпрог(плюс(x1 x2) десчисло(x1) десчисло(x2))".

2. тождфунк(x1). Для вычисления операции x1 над функцией, обозначенной переменной, целесообразна попытка найти в контексте кванторное тождество, определяющее значения это функции. Пример: "тождфунк(пределпослед)".

3. вычислить(x1). Для выражений с заголовком x1 целесообразна попытка вычисления с помощью вспомогательной задачи. Пример: "вычислить(интеграл)".

4. вычисл($f(s_1 \dots s_n)$). Для вычисления операции f целесообразно преобразование ее i-го операнда к заголовку s_i. Пример: "вычисл(умножматр(строки строки))".

5. выч(A вход($x_1 \dots x_n$) выход($y_1 \dots y_m$) B). Существует процедура, перечисляющая значения выходных переменных $y_1, \dots, y_m$ при заданных значениях входных переменных $x_1, \dots, x_n$, удовлетворяющие утверждению A. B - список утверждений "тип(z,t)", уточняющих типы t константных значений входных и выходных переменных z. Пример: "выч(равно(x1 умножение(x2 x3)) вход(1) выход(x2 x3) тип(x1 целое) тип(x2 целое) тип(x3 целое))". Существует процедура, перечисляющая разложения целого числа в произведение двух целочисленных сомножителей.

20.14 Подготовка к вычислению

1. Класс(A n). A есть выражение, содержащее переменную x1. Если вместо этой переменной подставлено выражение вида "$\mathrm{set}_{x_1 \dots x_n}(f(x_1 \dots x_n) = g(x_1 \dots x_n)$ &

$B(x_1 \dots x_n))$", то для вычисления значения выражения A рекомендуется разрешить уравнение под описателем "класс" относительно некоторой (по умолчанию - последней) переменной. Пример: "Класс(точки(x1 x2)3)". Соответствующий прием подготавливает задачу к вычислению площади поверхности, разрешая ее уравнение относительно последней переменной.

20.15 Установки на вывод теорем

Некоторые ячейки логического вывода в базе теорем имеют в своем первом пункте не теорему, а протокол, организующий какой-либо специализированный цикл логического вывода. Обычно исходными данными этого цикла служат теоремы, расположенные в последующих пунктах данной ячейки вывода и снабженные уточняющими их роль характеристиками. Подробнее об организации таких ячеек вывода будет рассказано в томе монографии, посвященном логическому выводу в базе теорем.

1. стандформа($A\ B_1(n_1) \dots B_k(n_k)$). Вывод упрощающих тождеств для нормализатора стандартной формы A. $B_i(n_i)$ - указатель оценки стоимости n_i операции B_i. Пример: "стандформа(стандобъединение объединение(1) пересечение(1) разность(1))".

2. факторизация(x1). Вывод тождеств для приведения к заголовку x1. Пример: "факторизация(умножение)".

3. функции(x1 x2 x3). Вывод тождеств для вычисления заданной характеристики простейших функций, определяемых операциями некоторого раздела. x1 - шаблон вычисляемой характеристики функции, x2 - переменная, обозначающая в нем рассматриваемую функцию, x3 - конъюнкция ограничений на параметры шаблона x1. Для указания раздела служит характеристика протокола "раздел(...)". Пример: "функции(образ(f, $[a,b]$), f, $f-$функция & $\mathrm{Val}(f) \subseteq \mathbb{R}$ & $\mathrm{Dom}(f) \subseteq \mathbb{R}$ & $a-$число & $b-$число & $a \le b$ & $c-$boolean & $d-$boolean)". Здесь c, d - указатели типа концов промежутка. Используется характеристика "раздел(элеменатрнаяалгебра)".

4. взаимнооднозначно(x1). Вывод утверждений о взаимной однозначности отображений, определяемых операциями раздела x1. Пример: "взаимнооднозначно(комплексныечисла)".

20.16 Ввод вспомогательных обозначений

В разделе собраны протоколы, на которых основаны приемы, вводящие вспомогательные обозначения для различных объектов в процессе решения задачи.

1. функподст(x1 x2). x1 - терм с подтермом "отображение(...)", для которого целесообразно ввести вспомогательное обозначение. x2 - условия на контекст. Предполагается, что x1 идентифицируется с подтермом условия задачи на преобразование. Пример: "функподст(inf(образ($\lambda_x(u(x), v(x)), a)$) и(корень не(входит(целаячасть корень))))".

2. обозначение(x1 x2). x1 - обозначаемое выражение, x2 - условия на контекст, в котором целесообразен ввод вспомогательного обозначения для x1. Пример: "обозначение(set_x(правсмежнкласс(a, G, H)), и(тип(доказать) контекст(операнд(x2 теквхожд)символ(x2 мощность))))".

3. обознач(x1 x2 x3). x1 - обозначаемое выражение, x2 - конъюнкция условий на параметры выражения x1, от которых будет зависеть обозначающая функциональная переменная. Эти условия содержат только данные параметры. x3 - условие на контекст, в котором целесообразен ввод функционального вспомогательного обозначения для x1. Пример: "обознач($\det(\lambda_{ij}(A(i,j), i \in \{1,\ldots,an+b\}$ & $j \in \{1,\ldots,an+b\}))$, $n-$ натуральное, и(тип(преобразовать) целое(a) целое(b) или(условие комментусловия(определитель Замена)) коммент(определитель значение теквхожд) не(заголовок($A(i,j)$, строки))))". Соответствующий прием вводит обозначение для определителя n-го порядка, чтобы вычислить его по рекурсии.

20.17 Координаты

1. точки(A B). Логический символ A - название координат. Выражение $B(x,K)$ обозначает множество объектов, координаты (типа A) которых в системе координат K образуют множество x. Пример - "точки(коорд точки)".

2. тчкоорд(A B). Логический символ A - название координат. Выражение $B(x,K)$ обозначает объект, координаты которого в системе координат K равны x. Пример: "тчкоорд(коорд, тчкоорд)".

20.18 Проверочные операторы

1. легковидеть(x1 x2). Для проверки утверждений вида x1 введен проверочный оператор с заголовком x2. Пример: "легковидеть(меньше(x1 x2) усмменьше)".

2. проверка(x1 x2). Для проверки утверждений вида x1 введен усиленный проверочный оператор с заголовком x2. Пример: "проверка(меньше(x1 x2) провменьше)".

20.19 Развертка

1. список(x1 x2) - x1 есть описатель "класс(...)", для которого предусмотрена идентификация с разверткой в конечный список, x2 - конъюнкция условий на параметры выражения x1, при которых развертка целесообразна. Пример: "список(класс(x23 перестановка(x23 номера(1 x14))) и(натуральное(x14) меньше(x14 5)))".

20.20 Идентифицирующие операторы

1. усм(x1 x2). Для обработки утверждения x1 предусмотрен идентифицирующий оператор, причем x2 - термы "вход(...)", "выход(...)", определяющие, какие

переменные рассматриваются в качестве входных, а какие - в качестве выходных. Терм "выход(...)" может отсутствовать. Примеры: "усм(принадлежит(x1 отрезок(x2 x3)) вход(x1 x2 x3))", "усм(принадлежит(x1 отрезок(x2 x3)) вход(x2 x3) выход(x1))".

Глава 21

Автоматическая характеризация теорем

Характеристики теорем создаются автоматически двумя способами. Во-первых, имеется процедура "характеризатор", создающая список характеристик из общих соображений. Во-вторых, прием процедуры вывода теорем может создать характеристики теоремы в обход характеризатора, располагая более точной информацией о цели вывода теоремы. Например, при выводе формулы корней квадратного уравнения будет создана лишь характеристика, явно указывающая на использование этой формулы для разрешения уравнения относительно заданной перемнной. Если предоставить свободу действий характеризатору, то были бы предложены другие, в общем-то, бесполезные характеристики. Например, было бы создано предложение использовать формулу "в обратном порядке", сворачивая дизъюнкцию равенств для корней в более компактно записываемое исходное уравнение. Тем не менее, первичный анализ определений и аксиом предметной области должен выполнять именно характеризатор, намечая цели для последующего логического вывода и позволяя создавать простейшие приемы.

Интерфейс характеризации теоремы

Еще раз напомним команды интерфейса, связанные с характеризацией теоремы. Чтобы из просмотра теоремы в базе теорем перейти в режим ручного ввода характеристик текстовым редактором, достаточно нажать "к" (кир.). Для получения справок о типах характеристик из просмотра теоремы нажимается "Ctr-к" (кир.), переводящее в оглавление типов характеристик. Если левой кнопкой мыши нажать на характеристику в списке прорисованных под теоремой характеристик, то снизу появляется информация о данной характеристике. Чтобы ее убрать, достаточно нажать "пробел" либо нажать левую кнопку мыши в нижней части экрана.

Чтобы обратиться к процедуре "характеризатор" для повторного создания характеристик, следует из просмотра теоремы нажать "Х" (кир.). Предварительно следует вручную удалить из списка характеристик элемент "пв". Такой элемент блокирует автоматическюу рехарактеризацию. Он создается процедурой вывода теорем для замораживания предложенного ею списка. Блокировку рехарактеризации вызывают также элементы с заголовками "блок", "протокол", "тожд", "станд", "теорема-приема". Часть характеристик теоремы в процессе рехарактеризации сохраняются, остальные - предварительно удаляются, а затем могут быть восстановлены характеризатором. Обязательно сохраняются характеристики с заголовками "определение",

"блокраздела", "блокприемов", "копия", "обознач", "авт", "посылки".

Чтобы протестировать работу характеризатору, вместо "X" нужно нажать "Ctr-x" (кир.). Тогда будут происходить выходы в отладчик ЛОСа при создании характеризатором каждой новой характеристики. На экране появится фрагмент программы оператора "характ", регистрирующего эту характеристику. Под программой - горизонтальная черта, а под ней указана (в виде значения переменной x1) сама характеристика. Для просмотра участка программы характеризатора, создавшего ее, нужно нажать "Page Up", и далее пользоваться обычными средствами отладчика ЛОСа. Для перехода к просмотру очередной характеристики последовательно нажимаются клавиши "0"(ноль) и "Enter". Чтобы оборвать цикл просмотра и выйти в главное меню, нажимается "Esc".

При развитии программы характеризатора может понадобиться поиск теорем, в которых срабатывает тот или иной его прием. Для этого нужно создать контрольную точку "трассировка(стоп 0)" в точке программы характеризатора, где располагается прием, выбрать в оглавении базы теорем пункт, являющийся корневым для ветви поиска, и нажать "X". Начнется пролистывание теорем данной ветви и запуск характеризатора на них. Результаты работы характеризатора зарегистрированы нигде не будут, но отладчик ЛОСа при каждой попытке применения приема будет показывать соответствующий кадр.

Выйти на программу интерфейса, выполняющую обращение к характеризатору при ручном его запуске из просмотра теоремы, можно через пункт "База теорем" - "Характеризация теорем" - "Обращение к характеризации теоремы" оглавления программ. Эта программа обслуживает нажатия клавиш "X", "Ctr-x" из просмотра теоремы. В соседнем пункте "Тестирование характеризации теорем раздела" расположена программа, обращающаяся к тестированию характеризатора для выбранной ветви оглавления базы теорем.

21.1 Процедура "характеризатор"

Обращение к процедуре характеризатора имеет вид "характеризатор(x1 x2 x3)", где x1 - теорема, x2 - исходный список ее характеристик. Выходной переменной x3 присваивается пополненный в результате характеризации список x2. Выйти на начальную точку программы характеризатора можно через пункт "База теорем" - "Характеризация теорем" - "Характеризатор" - "Исходная точка" оглавления программ.

Прежде всего, отбрасываются квазипротоколы с символом "преобр" в антецедентах. Для них характеризатор сразу выдает неизмененный ответ x2. Далее переменной x4 присваивается вхождение консеквента теоремы x1. Если теорема не имеет вид кванторной импликации, то переменной x4 присваивается вхождение ее первого символа. Дальнейшие действия зависят от того, какой символ расположен по вхождению x4.

Характеризатор представляет собой достаточно большую базу приемов, каждый из которых анализирует теорему независимо от других. При обучении системы новые приемы размещались в программе несколько хаотично. Поэтому рассмотрение их удобнее будет привязать не к последовательному прохождению всей программы, а к оглавлению этих приемов, хранящемуся там же, где и указанная выше ссылка на исходную точку программы.

Все вводимые характеристики добавляются к списку x2, который в конце и выдается как результат. Добавление очередной характеристики H выполняется оператором

"характ(H)". Значение x2 ему передавать не нужно, так как он самостоятельно находит значение x2 во внешнем кадре глобального стека интерпретатора и корректирует его.

Программа характеризатора легко восстанавливается по описанию характеристик теорем. Поэтому мы здесь ее не рассматриваем. Подробное рассмотрение данной программы можно найти в 8 томе монографии "Компьютерное моделирование логических процессов".

Глава 22

Создание спецификаций приемов

22.1 Интерфейс получения спецификаций для текущей теоремы в базе теорем

Как уже говорилось выше, имеется "ручной" интерфейс создания спецификаций приемов по теоремам. Из просмотра теоремы в базе теорем нажимается "г", после чего на экране прорисовываются теорема приема и его спецификация для первого элемента списка созданных спецификаций. Переходы между элементами этого списка выполняются с помощью клавиш "курсор вверх" - "курсор вниз". Фактически под теоремой приема прорисовывается не элемент "тип(...)", указывающий тип приема, а текст для названия этого типа. Под этим текстом изображаются (если они есть) остальные элементы спецификации. Теорема, текст названия типа приема и прочие элементы спецификации отделены друг от друга горизонтальными линиями. Для выхода из просмотра списка созданных спецификаций достаточно нажать "курсор влево".

По текущей просматриваемой спецификации можно создать прием ГЕНОЛОГа. Для этого нажимается клавиша "б". Прием создается, регистируется в буфере базы приемов и компилируется. Если его не перенести из буфера в другой раздел базы приемов, то после нажатия "Shift-O" (кир.) из оглавления базы приемов буфер приемов сбрасывается, а все созданные и зарегистрированные в нем приемы удаляются вместе со своими программами.

Если по спецификации нужно создать прием не в буфере базы приемов, а сразу зарегистрировать его в основной части этой базы, то из просмотра спецификации нажимается "с" (кир.). Автоматически выполняется переход в оглавление базы приемов. В этом оглавлении нужно создать новый либо выбрать старый концевой пункт, зайти в него и нажать "ш". Появится первый из приемов, созданных по спецификации. Если его нужно сохранить, то обычным образом нажимается F3 (сохранение и компиляция) либо F4 (сохранение без компиляции). Если данная версия приема не нужна, то для перехода к следующей версии нажимается "ш". Возвращение к предыдущим версиям списка не предусмотрено.

Если нужно изменить спецификацию ранее созданного приема, либо связать его с текущей просматриваемой спецификацией, то нажимается клавиша "у". Она переводит в оглавление базы приемов, где следует найти ранее созданную версию приема, войти в ее просмотр и нажать "Ctr-ц". Спецификация будет присвоена данной версии. Данная возможность практически не используется и обладает тем недостатком, что при

рассогласовании обозначений переменных в автоматически созданной теореме приема и "старой" теореме приема никакой их коррекции в элементах спецификации не производится.

Программу, запускающую обращение к процедуре спецификатора по текущей просматриваемой теореме и реализующую указанный выше интерфейс, можно найти в пункте "База теорем" - "Спецификатор" - "Обращение к спецификатору из просмотра теоремы" оглавления программ. Она не содержит каких-либо принципиальных моментов, и мы ее не рассматриваем.

22.2 Процедура "приемы"

Для создания спецификаций приемов по заданной теореме служит процедура "приемы(x1 x2 x3 x4 x5)". Ей передаются следующие входные данные: x1 - теорема, x2 - ссылка на узел теоремы в базе теорем, x3 - набор характеристик теоремы, x4 - установка на синтез приемов. Ссылкой на узел теоремы служит, как обычно, терм "теорема(A_1, A_2)", где A_1 - логический символ, A_2 - номер узла в статье символа A_1. Выходной переменной x5 передается набор пар (теорема приема - спецификация приема).

Набор x4 почти не используется. Если в него помещается элемент "новый", то создаются приемы только тех типов, которые пока не созданы по теореме x1.

Выйти на программу оператора "приемы" можно через пункт "База теорем" - "Спецификатор" - "Процедура "приемы" " оглавления программ.

Переменной x6 присваивается одноэлементый набор, состоящий из накопителя результата. Первоночально накопитель пуст. Переменной x7 присваивается символ "пустоеслово". Если имеется установка "новый", то этой переменной переприсваивается список логических символов, задающих типы приемов, уже созданных по теореме x1. Далее просматривается список характеристик x3. Для текущей характеристики x8 с заголовком x9 создается одноэлементый набор x10, состоящий из накопителя промежуточных результатов создания спецификаций по теореме x1 для характеристики x8. Заполнение этого накопителя происходит при обращении к справочнику "приемы" на символе x9.

Справочник "приемы" обрабатывает следующие входные данные: x1 - теорема, x2 - ссылка "теорема(A_1, A_2)" на ее узел в базе теорем, x3 - набор характеристик теоремы, x4 - текущая характеристика теоремы, x5 - одноэлементный набор, состоящий из накопителя результата. Текущим логическим символом служит заголовок характеристики x4. Справочник формирует спецификации приемов, быть может, модифицируя в каждом случае теорему x1. Накопитель в x5 заполняется парами (модифицированная теорема - спецификация приема).

В нашем случае справочнику передаются входные данные x1,x2,x3,x8,x10. После обращения к нему начинается просмотр элементов x12 накопителя x10. В ряде особых случаев происходит коррекция спецификации, а затем предпринимается обращение к процедуре "теоремаприема", которой передаются теорема пары x12, спецификация приема из этой же пары и введенный выше накопитель результата x6. Эта процедура выполняет преобразования теоремы приема согласно его спецификации и регистрирует полученную пару в накопителе x6. В частности, происходит отбрасывание указывающих о.д.з. антецедентов, избыточных из-за предположений о корректности

контекста срабатывания приема. Более подробно действия процедуры "теоремаприема" рассмотрены в 8 томе монографии "Компьютерное моделирование логических процессов".

После обработки всех характеристик теоремы x1 предпринимается отбрасывание приемов, избыточных ввиду наличия в том же накопителе x6 других приемов. Подробности приведены в 8 томе монографии. Затем выдается результат - содержимое накопителя x6.

22.3 Приемы спецификатора

Основную часть работы по созданию спецификаций приемов выполняет справочник "приемы". Входными данными при обращении к нему служат: x1 - теорема, x2 - ссылка "теорема(A_1, A_2)" на узел данной теоремы в базе теорем, x3 - набор характеристик теоремы, x4 - текущая характеристика теоремы, x5 - одноэлементный набор, представляющий собой накопитель результата и вначале состоящий из символа "пустоеслово". Логическим символом, на котором предпринимается обращение к справочнику, служит заголовок характеристики x4. В набор, зарегистрированный в накопителе x5, заносятся пары (модифицированная теорема приема - спецификация приема).

Если в качестве текущей характеристики выступает символ "протокол", то справочник "приемы" немедленно переадресует обработку справочнику "протокол". Входными данными последнего служат: x1 - протокол базы теорем, x2 - ссылка "теорема(A_1, A_2)" на узел протокола (рассматриваемого как теорема) в базе теорем, x3 - набор характеристик протокола, x4 - одноэлементный набор, являющийся накопителем результата. Логическим символом, на котором происходит обращение к справочнику, служит заголовок протокола. Накопитель x4 заполняется так же, как у справочника "приемы".

Приемы справочников "приемы" и "протокол" расположены в ветви "База теорем" - "Спецификатор" - "Процедуры справочников "приемы" и "протокол" ". В этой ветви приемы обоих справочников объединены. Оглавление ветви практически совпадает с оглавлением логического ассемблера: приемы спецификатора сгруппированы по типам тех приемов решателя, спецификации которых они создают.

Здесь мы лишь проиллюстрируем приемы справочника "приемы" на нескольких примерах. Этих приемов достаточно много, хотя они обычно и несложны. Описание их можно найти в 8 томе монографии.

1. Общая стандартизация (характеристика "нормализация").

   Если теорема имеет характеристику "нормализация(N)" и не имеет характеристики "описатель(...)", то создается спецификация "тип(общнорм)", "направл(N)". Пример:

   $\forall_a(a - \text{число} \rightarrow a \cdot 1 = a)$

   Спецификация - "тип(общнорм)", "направл(второйтерм)".

2. Исключение описателя "класс" (характеристика "описатель").

Характеристика "описатель(N)" указывает на тождество, используемое для перехода к более простым описателям либо для исключения описателей. Проверяется, что заменяющий терм x8 не имеет связанных переменных, а заменяемый - имеет заголовок "класс". Создается спецификация "тип(цепьвоглавлении)", "направл(N)". Пример:

$\forall_{af}(\text{слой}(f,a) = \text{set}_x(x \in \text{Dom}(f)\ \&\ a = f(x)))$

Спецификация - "тип(цепьвоглавлении)", "направл(первыйтерм)".

3. Переход от параметрического задания класса к непосредственному (характеристика "парамописание").

   Характеристика "парамописание(N)" указывает на тождество, преобразующее параметрическое описание класса в его явное задание. Вводится спецификация "тип(константа)", "направл(N)". Пример:

   $\forall_{abcd}(\neg(a^2+c^2=0) \to \text{set}_{xy}(\exists_t(x = at+b\ \&\ y = ct+d\ \&\ t - \text{число})) = \text{set}_{xy}(cx - ay + ad - bc = 0\ \&\ x - \text{число}\ \&\ y - \text{число}))$

   Спецификация - "тип(константа)", "направл(второйтерм)".

4. Переход в задаче на преобразование к уже имевшимся подтермам (характеристика "свертка").

   Характеристика "свертка(N)" означает, что тождество обеспечивает переход к сокращенной записи.

   Проверяется, что теорема не имеет характеристик "нормализация(...)" и "дистрибразвертка(...)". Рассматриваются заменяемый терм t_1 и заменяющий терм t_2. Проверяется, что самый сложный подтерм терма t_2 - он сам и что терм t_1 не содержит символа "вариант". Находятся все самые сложные подтермы $r_1, \ldots, r_m$ терма t_1. Создается спецификация "тип(поглощается)", "направл(M)", "исключение(t_2)", "терм($r_1 \ldots r_m$)".
   Здесь M - направление, противоположное N. Пример:

   $\forall_{abc}(\neg(a-1=0)\ \&\ 0 < a\ \&\ a - \text{число} \to \log_a b / \log_a c = \log_c b)$

   Спецификация - "тип(поглощается)", "направл(первыйтерм)", "исключение($\log_c b$)", "терм($\log_a b, \log_a c$)".

5. Шаг сведения условия задачи на описание к кратным вхождениям единственного неизвестного подтерма (характеристика "склейка").

   Характеристика "склейка(x N)" указывает на тождество, позволяющее перейти от выражения с несколькими вхождениями переменной x к выражению с одним вхождением этой переменной. N - направление замены.

   Проверяется, что теорема не имеет характеристики с заголовком "дистрибразвертка" либо "нормализация". Заменяющий терм имеет единственный содержащий переменную x подтерм t максимальной сложности. Его заголовок не ассоциативен, а число параметров меньше 6. Создается спецификация "тип(Источники)", "направл(N)", "терм(t)". Пример:

$\forall_{abcde}(e \cdot \log_c(a^d) - e \cdot \log_c(b^d) = ed \cdot \log_c(a/b))$

Спецификация - "тип(Источники)", "направл(второйтерм)", "терм($\log_c(a/b)$)".

6. Сокращенная переформулировка выражений при завершающем редактировании (характеристика "свертка").

   Характеристика "свертка(N)" означает, что тождество обеспечивает переход к сокращенной записи.

   Проверяется, что нет характеристики с заголовком "группмножитель" и что теорема не имела обобщения по характеристике "свертка". Тогда создается спецификация "тип(левпозиция)", "направл(N)". Пример:

   $\forall_{aeg}(\text{прообраз}(g, a) \cap \text{прообраз}(g, e) = \text{прообраз}(g, a \cap e))$

7. Группировка относительно выражения с неизвестными (характеристика "склейка").

   Характеристика "склейка(x N)" указывает на тождество, позволяющее перейти от выражения с несколькими вхождениями переменной x к выражению с одним вхождением этой переменной. N - направление замены.

   Находится список $x_1, \ldots, x_k$ всех переменных, выделенных характеристикой "склейка(x_i N)" (при заданном N). Проверяется, что глубина вхождения любой из них в заменяющий терм меньше глубины ее вхождения в заменяемый. Проверяется, что заменяющий терм не имеет унифициуемых аргументов, содержащих одну из данных переменных. Тогда создается спецификация "тип(нормнеизв)", "направл(N)", "неизвестные($x_1 \ldots x_k$)". Пример:

   $\forall_{abc}(0 \le a \ \& \ 0 < b \to (a/b)^c = a^c/b^c)$

   Спецификация - "тип(нормнеизв)", "направл(первыйтерм)", "неизвестные(c)".

8. Уменьшение глубины неизвестной (характеристика "глубина").

   Характеристика "глубина(x N)" указывает, что тождество перестановочного типа уменьшает глубину переменной x до единицы. N - направление замены.

   Проверяется отсутствие характеристики "нормализация" и создается спецификация "тип(точкаокружности)", "направл(N)", "неизвестные(x)". Пример:

   $\forall_{abcde}(0 < e \to e^{c \log_d a/b} = a^{c \log_d e/b})$

   Спецификация - "тип(точкаокружности)", "направл(второйтерм)", "неизвестные(a)".

9. Использование равенств из посылок для определения неизвестного выражения (характеристика "опрнеизв").

   Характеристика "опрнеизв($i_1 \ldots i_n$)" указывает на тождество, позволяющее определить значение неизвестного выражения с помощью равенств из посылок. $i_1, \ldots, i_n$ - список номеров антецедентов, идентифицируемых с посылками.

Создается спецификация "тип(вставка)", "направл(второйтерм)", "антецедент($i_1 \dots i_n$)". Пример:

$\forall_{abcd}(\neg(d = 0)\ \&\ d|a| = c|b| \rightarrow |a/b| = c/d)$

Спецификация - "тип(вставка)", "направл(второйтерм)", "антецедент(2)".

10. Стандартизация с помощью вычислений (характеристика "вычпрог").

    Характеристикой "вычпрог(N F T)" снабжаются тождества либо эквивалентности, которые в случае применения в направлении N обеспечивают упрощение относительно констант, использующее вычисления ГЕНОЛОГа. F - конъюнкция фильтров, уточняющих контекст стандартизации (обычно - типы константных значений переменных), T список подвыражений заменяющего терма, обрабатываемых путем непосредственных вычислений.

    Как и в предыдущем пункте, для подвыражений t списка T вводятся обозначающие их новые переменные x, и равенства $x = t$ добавляются в начало списка антецедентов теоремы. В заменяющей ее части выражения t заменяются на переменные x. Измененная теорема сопровождается спецификацией "тип(исключприем)", "направл(второйтерм)", "см(F)". Отличие от предыдущего пункта заключается в том, что вместо обработки константных подтермов специальными операторами используются стандартные вычисления ГЕНОЛОГа: добавленные к антецедентам равенства выделяются указателем "программа". Пример:

    $\forall_{abcdef}(f = bc\ \&\ d = ac + be \rightarrow a/b + e/c = d/f)$

    Спецификация - "тип(исключприем)", "направл(второйтерм)", "см(десчисло(b десчисло(e) десчисло(a) десчисло(c))".

11. Свертка условного выражения (характеристика "свертка").

    Характеристика "свертка(N)" означает, что тождество обеспечивает переход к сокращенной записи.

    Если заменяемый терм имеет заголовок "вариант", а заменяющий не содержит символа "вариант", то создается спецификация "тип(новооператор)", "направл(N)". Пример:

    $\forall_a(a - \text{число} \rightarrow (-a \text{ при } a < 0,\ \text{иначе } a) = |a|)$

12. Декомпозиция сложной операции (характеристика "нормализация").

    Проверяется отсутствие характеристики с заголовком "описатель". Проверяется, что заменяемый терм бесповторный, а заменяющий - не бесповторный. Создается спецификация "тип(сопровождтерм)", "направл(N)", где N - направление общей стандартизации. Пример:

    $\forall_{abc}(c - \text{rational}\ \&\ \neg(\text{знаменатель}(c) - \text{even}) \rightarrow a^c b^c = (ab)^c)$

    Спецификация - "тип(сопровождтерм)", "направл(первыйтерм)".

13. Выражение координат объекта через указанные в посылках координаты другого объекта (характеристика "систкоорд").

    Проверяется, что консеквент K имеет вид равенства выражения r, обозначающего какие-либо координаты, выражению t с заголовком "набор". Составляется список S всех выражений t', для которых среди антецедентов имеется равенство вида $r' = t'$, где r' обозначает какие-либо координаты и имеет параметры, не входящие в r, а t' имеет заголовок "набор". Проверяется, что список S непуст и содержит все параметры терма t. Находится список $i_1, \ldots, i_n$ номеров антецедентов, использованных при составлении списка S. Затем создается спецификация "тип(Набор)", "направл(второйтерм)", "антецедент($i_1 \ldots i_n$)". Пример:

    $\forall_{ABCKabcd}$(коорд(вектор$(AC), K) = (a, b)$ & коорд(вектор$(CB), K) = (c, d) \rightarrow$ коорд(вектор$(AB), K) = (a + c, b + d)$)

    Спецификация - "тип(Набор)", "направл(второйтерм)", "антецедент(1 2)".

14. Безусловная общая стандартизация одного утверждения (характеристика "общнорм").

    Характеристика "общнорм(N)" указывает на эквивалентность общей стандартизации утверждений, не имеющих вида конъюнкции либо дизъюнкции. Заменяющее утверждение элементарно. N - направление замены.

    Проверяется отсутствие характеристики "усиление". Если теорема имеет антецеденты, не используемые для сопровождения по о.д.з., то проверяется, что число параметров консеквента не менее двух, а у каждого антецедента параметр единственный. Кроме того, проверяется, что не выполнено ни одно из следующих условий:

    (a) Заменяемая часть имеет своим заголовком квантор. В случае квантора существования рассматривается список S конъюнктивных членов подквантворного утверждения, в случае квантора общности - список S антецедентов и консеквента. В этом списке имеет такое утверждение, что некоторая переменная, не относящаяся к кванторной приставке квантора, входит в него неоднократно.

    (b) Заменяемая часть - равенство с невырожденными числовыми атомами, ни один из которых не входит в заменяющую часть. Эта часть не является равенством.

    Тогда создается спецификация "тип(нормэкв)", "направл(N)". Пример:

    $\forall_{abc}(a + b = a + c \leftrightarrow b = c)$

15. Элементарная переформулировка с исключением сложного понятия (характеристика "уменьшение").

    Характеристика "уменьшение(N)" указывает на эквивалентность, исключающую символы с наибольшей оценкой сложности. N - направление замены.

    Проверяется, что теорема не имеет характеристик с заголовками "общнорм" и "развертка", причем обе части эквивалентности суть элементарные утверждения. Для каждой переменной x, входящей в заменяющее утверждение и не

встречающейся в подтермах заменяемого утверждения, имеющих максимальную сложность, проверяется выполнение следующих условий:

(a) Наибольшая из оценок сложности содержащих x подтермов заменяемого утверждения меньше наибольшей оценки сложности содержащих x подтермов заменяющего.

(b) Существует вхождение x в заменяющее утверждение, расположенное внутри его подтерма, не имеющего своим заголовком ни один из символов "равно", "набор", "значение".

Если они выполнены для какого-либо x, то спецификация не создается. Иначе создается спецификация "тип(обрыв)", "направл(N)". Пример:

$\forall_{ab}$(верхняягрань$(a,b) \to$ наибольший$(a,b) \leftrightarrow a \in b)$

16. Конъюнктивная декомпозиция элементарного утверждения (характеристика "и").

Характеристика "и(N)" указывает на эквивалентность, декомпозирующую элементарное утверждение в конъюнкцию бесквантерных утверждений. N - направление замены.

Проверяется, что заменяющее утверждение не содержит символа "или". Если заменяемое утверждение содержит обозначение каких-либо координат, а заменяющее - обозначение $Q(\ldots)$ отдельной координаты, то проверяется наличие антецедента, содержащего символ Q. Затем создается спецификация "тип(огрсверху)", "направл(N)". Пример:

$\forall_{abc}(a \in b \setminus c \leftrightarrow \neg(a \in c)\ \&\ a \in b)$

17. Кванторная свертка (характеристика "кванторнаясвертка")

Характеристика "кванторнаясвертка(N)" указывает на эквивалентность для кванторной свертки. N - направление замены для исключения квантора.

Создается спецификация "тип(кванторнаясвертка)", "направл(N)". Пример:

$\forall_{ab}(a \subseteq b \leftrightarrow \forall_c(c \in a \to c \in b))$

18. Попытка использования явного параметрического описания при получении частичного ответа (характеристика "параметризация").

Характеристика "параметризация" указывает на эквивалентность с квантором существования в одной из своих частей, которую можно неизбыточным образом использовать для получения явного параметрического описания.

В качестве направления замены N выбирается то, которое приводит к появлению квантора существования. Проверяется, что заменяемая часть представляет собой элементарное утверждение. Тогда создается спецификация "тип(параметризация)", "направл(N)". Пример:

$\forall_{ab}(a \subseteq b \leftrightarrow \exists_c(a = b \cap c\ \&\ c - \mathrm{set}))$

Спецификация - "тип(параметризация)", "направл(второйтерм)". Соответствующий прием применяется в задачах на описание, имеющих цель "пример" либо "параметризация", причем a - неизвестная.

19. Разрешение условия задачи на описание либо посылки задачи на исследование относительно заданных неизвестных (характеристика "глуб").

    Характеристика "глуб(x N)" указывает на эквивалентность, заменяемая часть которой - элементарное утверждение, имеющее вхождения переменной x, глубина которых (с отбрасыванием внешнего отрицания, если оно есть) больше 1, а заменяющая построена при помощи логических связок из утверждений, содержащих каждое не более одного вхождения переменной x, и притом глубины 1. N - направление замены.

    Проверяется, что теорема не имеет характеристик "и(...)", "или(...)", "общнорм(N)", "сложнреш". Составляется список $x_1, \ldots, x_n$ всех переменных x_i, встречающихся в характеристиках теоремы, имеющих вид "глуб(x_i N)". Затем создается спецификация "тип(глуб)", "неизвестные($x_1, \ldots, x_n$)", "направл(N)". Пример:

    $\forall_{abc}(c - \text{число} \to a + b = c \leftrightarrow a = c - b)$

    Спецификация - "тип(глуб)", "направл(второйтерм)", "неизвестные(a)".

20. Преобразования условия задачи на описание либо посылки задачи на исследование, исключающее сложное выражение с неизвестными (характеристика "неизвоценка").

    Характеристика "неизвоценка(N F)" указывает на эквивалентность, применение которой в направлении N позволяет получить более простые выражения с неизвестными. F - фильтр, уточняющий контекст.

    Создается спецификация "тип(прообраз)", "направл(N)", "см(F)". Пример:

    $\forall_{abcde}(e - \text{rational} \ \& \ \neg(\text{числитель}(e) - \text{even}) \ \& \ \neg(\text{знаменатель}(e) - \text{even}) \to ab^e + cd^e < 0 \leftrightarrow a^{1/e}b + c^{1/e}d < 0)$

    Спецификация - "тип(прообраз)", "направл(второйтерм)", "см(и(не(контекст(операнд(фикс(0 1 1)х6)вид(х6 умножение(х7 х8)) не(известно(х8)) не(контекст(вид(х8 степень(х9 х10)) не(известно(х9)))) единица(1 х7) заменазнака(минус х7))) известно(e)))". Соответствующий прием исключает степенные выражения с неизвестными основаниями b, d, в предположении, что a, c, e известны.

21. Переформулировка подутверждения условия задачи через координаты (характеристика "числопред").

    Характеристика "числопред(N)" указывает на эквивалентность, выражающую утверждение через параметры координат. N - направление замены.

    Создается спецификация "тип(сравн)", "направл(N)", "антецедент($i_1 \ldots$ i_n)", где $i_1, \ldots, i_n$ - номера всех антецедентов, представляющих собой равенства для координат. Пример:

    $\forall_{ABCKabcdxy}(\text{коорд}(A, K) = (a, b) \ \& \ \text{коорд}(B, K) = (c, d) \ \& \ \text{коорд}(C, K) = (x, y) \to C \in \text{прямая}(AB) \leftrightarrow (c - a)(y - b) = (d - b)(x - a))$

    Спецификация - "тип(сравн)", "направл(второйтерм)", "антецедент(1 2 3)".

22. Непосредственное усмотрение истинности либо ложности (характеристика "элементарно").

    Характеристика "элементарно" указывает на кванторную импликацию без существенных посылок, имеющую элементарный консеквент.

    Создается спецификация "тип(элементызадачи)". Пример:

    $\forall_b(\text{непересек}(b, \emptyset))$

    Соответствующий прием имеет заголовок "второйтерм" и заменяет утверждение непересечения с пустым множеством на константу "истина".

23. Соотношение пропорциональности для двух числовых атомов (характеристика - "пропорция").

    Характеристика "пропорция(A B)" указывает на соотношение пропорциональности для числовых атомов A, B.

    Создается спецификация "тип(оценка)". Пример:

    $\forall_{ABCDEF}(\text{разныеточки}(A, C)\ \&\ \text{разныеточки}(B, C)\ \&$
    $\text{разныепрямые}(\text{прямая}(AC), \text{прямая}(BC))\ \&\ l(AE) = l(CE)\ \&$
    $l(BD) = l(CD)\ \&\ D \in \text{прямая}(BC)\ \&\ E \in \text{прямая}(AC)\ \&$
    $F \in \text{прямая}(AD)\ \&\ F \in \text{прямая}(BE) \to 2l(EF) = l(BF))$

    Прием выводит соотношение для длин отрезков медиан, возникающих при их пересечении.

24. Вывод неравенства для невырожденных числовых атомов (характеристика "числоценка").

    Характеристика "числоценка" указывает на простую импликацию, дающую неравенство для невырожденных числовых атомов.

    Создается спецификация "тип(элемент)". Пример:

    $\forall_{ABCD}(\text{ромб}(ABCD)\ \&\ \angle(BAD) < \pi/2 \to \pi/2 < \angle(ABC))$

25. Вывод многоместного отношения (характеристика "вывод").

    Характеристика "вывод" указывает на простую импликацию, которая может представлять интерес для создания приема вывода, не вводящего в рассмотрение новых сложных объектов.

    Проверяется, что консеквент имеет более двух корневых операндов и что теорема не имеет характеристики с заголовком "варопер". Тогда создается спецификация "тип(унификация)". Пример:

    $\forall_{ABCDE}(\text{окружность}(DE)\ \text{вписана в фигура}(ABC) \to \text{биссектриса}(BACD))$

26. Разбор случаев в посылках (характеристика "дизъюнкция").

    Характеристика "дизъюнкция" указывает на кванторную импликацию, у которой консеквент имеет вид дизъюнкции.

    Создается спецификация "тип(проверка)". Пример:

    $\forall_{ABC}$(актив($l(AB)$) & актив($l(AC)$) & актив($l(BC)$) & $B \in$ прямая(AC) $\rightarrow$ $b \in$ отрезок(AC) $\vee$ $C \in$ отрезок(AB) $\vee$ $A \in$ отрезок(BC))

    Одно из расстояний имеет тип "неизв", одно - известно, а третье уже рассматривается в задаче.

27. Выражение координат объекта через параметры уравнений для координат множеств объектов (характеристика "систкоорд").

    Характеристика "систкоорд" указывает на тождество для определения координат объекта.

    Проверяется наличие в антецедентах равенства для координат множества объектов, у которого параметры правой части пересекаются с параметрами консеквента. Затем создается спецификация "тип(равныедлины)". Пример:

    $\forall_{ABKabcd}$(прямкоорд(K) & коорд(окружность(AB), K) = $\mathrm{set}_{xy}(ax^2 + ay^2 + bx + cy + d = 0$ & $x -$ число & $y -$ число) $\rightarrow$ коорд(A, K) = $(-b/(2a), -c/(2a))$ & $\neg(a = 0)$)

28. Обратный вывод с переходом к более простым понятиям при подборе примера (характеристика "подбор").

    Характеристика "подбор(t N)" указывает на кванторную импликацию, обеспечивающую попытку обратного вывода с переходом к более простым понятиям. t - переменная либо терм, указывающие неизвестное выражение. N - набор номеров заменяющих антецедентов.

    Если t - неоднобуквенный терм, то вместо него далее берется указатель его вхождения в консеквент теоремы. Создается спецификация "тип(перечислфрагментов)", "подборзначений(N)", "см(не(известно(t)))". Пример:

    $\forall_{mn}(m -$ целое & $n -$ целое & $0 < n - m$ & $0 < m \rightarrow \neg(n|m))$

    Спецификация - "тип(перечислфрагментов)", "подборзначений(3 4)", "см(не(известно($n|m$)))".

29. Прием проверочного оператора (характеристика "легковидеть(...)").

    Характеристика "легковидеть(P m)" указывает на простую импликацию, которую можно неизбыточным образом использовать в проверочном операторе с заголовком P, причем ее m -й антецедент - непосредственно идентифицируемый, а остальные антецеденты обрабатываются проверочными операторами.

    Проверяется, что все антецеденты элементарны, и создается спецификация "тип(спуск)", "оператор(P)", "антецедент(m)". Если имеется характеристика "Спуск"

либо "Спуск($n_1 \dots n_k$)", то к спецификации добавляется элемент "указатель(спуск)" либо, соответственно, "указатель(спуск($n_1 \dots n_k$))". Напомним, что такие характеристики указывают на возможность обрыва дальнейших попыток применения прочих приемов проверочного оператора при полной либо частичной идентификации антецедентов данного приема. Пример:

$\forall_{abc}(a < b \ \& \ b + c \leq 0 \rightarrow a + c < 0)$

Спецификация - "тип(спуск)", "оператор(усмменьше)", "антецедент(1)".

30. Прием нормализатора общей стандартизации выражений (характеристика "нормализация").

    Характеристика "нормализация(N)" указывает на тождество, обеспечивающее общую стандартизацию. N - направление замены.

    Проверяется, что теорема не имеет характеристики вида "описатель(...)". Находится заголовок s заменяемой части. Определяется нормализатор P общей стандартизации выражений с заголовком s. Затем создается спецификация "тип(норм)", "оператор(P)", "направл(N)". Пример:

    $\forall_{ab}(a \cup (b \setminus a) = a \cup b)$

    Спецификация - "тип(норм)", "оператор(нормобъединение)", "направл(второйтерм)".

31. Прием синтезатора (характеристика "синтезатор").

    Характеристика "синтезатор(P)" указывает на теорему приема пакетного синтезатора P.

    Создается спецификация "тип(синтезатор)", "оператор(P)". Пример:

    $\forall_{abcd}(a \in b \ \& \ c \in d \rightarrow (a, c) \in b \times d)$

    Спецификация - "тип(синтезатор)", "оператор(выборточки)". Для выбора элемент прямого произведения выбираются элементы сомножителей.

32. Прием анализатора, выводящий следствия (характеристика "анализатор").

    Характеристика "анализатор(P)" указывает на прием анализатора P, предназначенный для вывода следствий.

    Создается спецификация "тип(Неизвестные)", "оператор(P)". Пример:

    $\forall_{ABCpq}(\text{актив}(\angle(ACB)) \ \& \ \text{актив}(l(AC)) \ \& \ \text{актив}(l(BC)) \ \& \ p = \sin(\angle(ACB) + \angle(ABC)) \ \& \ q = \sin(\angle(ABC)) \rightarrow pl(AC) = ql(BC))$

    Спецификация - "тип(Неизвестные)", "оператор(синусы)". Прием выводит соотношение пропорциональности, используя теорему синусов.

Глава 23

Общая схема логического вывода в базе теорем

Источник приема часто представляет собой теорему, полученную из "обычной" теоремы цепочкой модификаций и обобщений, направленных на расширение области применимости приема. Рассмотрим простой пример. Среди базисных теорем тригонометрии имеется тождество, связывающее между собой значения синуса и косинуса:

$\forall_a((\sin a)^2 + (\cos a)^2 = 1)$

Это тождество подсказывает прием общей стандартизации, применяемый слева направо. Чтобы такой прием срабатывал в ситуациях, когда перед квадратами находятся одинаковые коэффициенты, тождество следует несколько обобщить:

$\forall_{ab}(b(\sin a)^2 + b(\cos a)^2 = b)$

Следующий шаг обобщения - учет случаев, когда коэффициент b имеет сомножителями некоторые степени синуса и косинуса a:

$\forall_{abcdef}(c - d = 2 \ \& \ e - b = 2 \rightarrow f(\cos a)^c(\sin a)^b + f(\cos a)^d(\sin a)^e = f(\sin a)^b(\cos a)^d)$

Заметим, что прием обеспечивает усмотрение вырожденных нулевых значений показателей степени b, e. Наконец, если коэффициенты при слагаемых различны, но "подобны", т.е. имеют вид pK и qK, где p, q - десятичные константы одинакового знака, то можно так обобщить тождество, чтобы оно применялось к слагаемому с меньшим по модулю численным коэффициентом p и части другого слагаемого, имеющей такой же численный коэффициент. В результате получаем окончательную версию теоремы, являющуюся источником приема решателя:

$\forall_{abcdefghijk}(a - b = 2 \ \& \ c - d = 2 \ \& \ (0 < e \ \& \ 0 < f \ \& \ g = \min(e, f) \ \vee \ e < 0 \ \& \ f < 0 \ \& \ g = \max(e, f)) \ \& \ h = e - g \ \& \ i = f - g \rightarrow ej(\cos k)^c(\sin k)^b + fj(\cos k)^d(\sin k)^a = hj(\cos k)^c(\sin k)^b + ij(\cos k)^d(\sin k)^a + gj(\cos k)^d(\sin k)^b)$

Этот пример показывает типичную ситуацию: хотя источник приема и является логически корректным истинным утверждением, он может сильно отличаться от базисной теоремы предметной области за счет ввода большого числа обобщающих "технических" параметров. Иногда таких параметров нет, а источник приема представляет собой просто результат варьирования базисной теоремы с помощью некоторых других ранее освоенных теорем.

Переход от базисных теорем к теоремам, являющимся источниками приемов, проиллюстрированный на приведенном выше примере, будем называть программирующим логическим выводом. Этот вывод является первым звеном в цепочке алго-

ритмизации знаний. Базисные теоремы берутся из учебных пособий и прежде всего должны быть адаптированы к последующему созданию приемов. Так как решатель задач дает огромное количество теорем приемов, он фактически оказывается задачником на программирующий логический вывод. Для каждой теоремы приемов должна быть прослежена цепочка переходов, выводящая ее из некоторых базисных теорем, и предложены приемы программирующего вывода, обеспечивающие реализацию данной цепочки. Это напоминает ситуацию, наблюдавшуюся при обучении решателя по стандартным задачникам.

Помимо программирующего логического вывода, база теорем требует также логического вывода, приводящего к обнаружению новых базисных теорем. Этот логический вывод можно назвать исследовательским. Его алгоритмизация означает поиск объяснений того, как можно было бы открыть уже известные основополагающие факты предметной области, и создание соответствующих приемов. Первоначально предполагалось, что обучение логической системы каждому из этих двух типов логического вывода будет происходить по отдельности. Однако, в процессе работы обнаружилось, что граница между ними весьма условная, и удобнее объединить обе группы приемов логического вывода в одну. Приемы исследовательского вывода, приводящие к открытию формулы корней квадратного уравнения либо формулы Кардано, либо к открытию теоремы Пифагора, не сильно отличаются по своей трудоемкости и своим принципам от типичных приемов программирующего вывода. Объединение их в рамках общей процедуры упрощает ее тестирование по мере обучения.

Как уже говорилось в предыдущем томе, база теорем разбита на так называемые ячейки логического вывода. Каждая ячейка представляет собой концевой раздел оглавления базы теорем, в первом пункте которого собраны теоремы, из которых должны быть выведены теоремы остальных пунктов данного раздела. При выводе могут быть использованы любые другие теоремы базы теорем. В процессе обучения системы логического вывода и тестирования ее при необходимости вручную создаются блокировки, не разрешающие при выводе теоремы использовать ее саму. Если система выводит новые теоремы, то такие блокировки не нужны.

База теорем первоначально не была разбита на ячейки логического вывода. Эти ячейки стали создаваться постепенно, в процессе развития приемов программирующего вывода. В настоящее время часть теорем погружена в эти ячейки, а часть остается вне них. Предполагается, что в конечном счете все теоремы окажутся только внутри ячеек логического вывода.

Процедура логического вывода получает в качестве своего входного данного ссылку на ячейку логического вывода. Она представляет собой базу приемов, реализованных на ЛОСе. Прием может искать в базе теорем дополнительные теоремы, необходимые для вывода, двумя способами.

Во-первых, имеется огромное количество справочников поиска теорем. Справочнику сообщается некоторый логический символ, и он выдает набор ссылок на теоремы заданного типа, связанные с этим символом. Все приемы справочников поиска теорем реализованы на ГЕНОЛОГе. Они создаются автоматически по характеристикам теоремы. Новые справочники поиска теорем создаются при обучении системы по мере надобности. Обычно сначала вводятся один-два приема нового справочника, необходимые для проработки текущего примера вывода теоремы. Чрезмерное пополнение запаса этих приемов может сильно активизировать действия процедуры вывода и создать лавину дополнительных следствий. Поэтому подключение новых приемов

поиска теорем целесообразно сопровождать эвристическими ограничениями активности приемов вывода теорем. Многообразие справочников поиска теорем образует систему быстрого поиска в базе теорем.

Во-вторых, для поиска нужных теорем можно использовать оглавление базы теорем и просматривать все теоремы выбранных разделов подряд, отбирая из них нужные. В первую очередь, при этом используются характеристики теорем. Это - система медленного поиска. Однако, она удобна тем, что не требует заблаговременного создания специальных приемов. В действительности, данный поиск не очень сильно замедляет работу процедуры вывода теорем, но лишь из-за того, что к нему прибегает малая доля приемов.

Процедура логического вывода работает со списком теорем, в который изначально заносятся стартовые теоремы ячейки логического вывода. Эта процедура представляет собой базу приемов. Каждый прием имеет свой уровень срабатывания, а теоремы списка сопровождены весами. Работает простейшая схема сканирования теорем списка: выбирается теорема, вес которой равен текущему уровню сканирования, и к ней применяются все приемы с данным уровнем срабатывания. После этого вес теоремы увеличивается на единицу. Новые теоремы заносятся в конец списка и получают вес 0. Таким образом, сканирование списка теорем происходит "из конца в начало". Приемы сопровождены эвристическими фильтрами, ограничивающими определенные последовательности их применений. Это приводит к тому, что вывод следствий обрывается за конечное число шагов. При появлении слишком длинных циклов вывода вводятся новые эвристические фильтры. Если представляют интерес следствия какой-либо теоремы, полученной в цикле вывода, то для ее копии создается независимая ячейка вывода.

Каждый прием вывода теорем связан с какой-то конкретной характеристикой теоремы, так что вывод следствий теоремы происходит при сканировании списка ее характеристик.

Теоремы, полученные процедурой вывода, сохраняются в оперативной памяти и поначалу не регистрируются в файлах. Это упрощает отладку процедуры. Отбор и сохранение теорем выполняются другими процедурами.

23.1 Интерфейс логического вывода в базе теорем

Оглавление приемов вывода теорем

Приемы вывода теорем реализованы на ЛОСе, и оглавление этих приемов представляет собой ветвь оглавления программ. Чтобы попасть в корень данной ветви, нужно пройти из корня оглавления программ по пути "База теорем" - "Программирующий вывод"/ - "Приемы вывода теорем".

Для последнего перехода в действительности имеются три возможности: "Приемы вывода теорем (справочник ПРОГРВЫВОД)", "Приемы вывода теорем (справочник ТЕОРЕМЫ)" и "Приемы вывода теорем (процедура БЛОКВЫВОДА). В первом из этих трех пунктов собрана основная часть приемов вывода. Они активируются при рассмотрении характеристики теоремы либо квазипротокола. Во втором пункте представлены приемы, активируемые заголовками протоколов базы теорем. Наконец, в последнем пункте - совсем немного приемов, активируемых при рассмотрении любой теоремы безотносительно к ее характеристикам.

При переходе в пункт "Приемы вывода теорем (справочник ПРОГРВЫВОД)" появляется меню, заголовки пунктов которого суть названия характеристик теорем. В ветви каждого такого пункта сгруппированы приемы вывода теорем, активируемые рассмотрением соответствующей характеристики. Концевые пункты этой ветви позволяют обычным образом (клавиша "курсор вправо") перейти к просмотру программы приема вывода. Обычно контрольная точка "прием(N)", появляющаяся при этом, расположена в конце программы приема - вблизи точки регистрации выведенной им теоремы в списке результатов вывода. Чтобы найти начало программы, нужно найти предшествующую точке "прием(N)" точку "ктл(N)". Иногда несколько операторов "ктл", относящихся к разным приемам, склеены в одну дизъюнкцию. Для нужд отладки в программы приемов вывода теорем вставляются также контрольные точки "ктд(xi)". Они располагаются сразу после того, как переменной xi присваивается дополнительная теорема, участвующая в выводе.

Контрольные точки "ктл" и "ктд" нужны для отсечения ненужных попыток процедуры вывода, если требуется воспроизвести лишь вывод какой-то конкретной теоремы. Программы этих операторов проверяют наличие "ключей" - комментариев, ограничивающих вывод, и блокируют дальнейшие действия, если номер контрольной точки либо ее теорема не совпадают с указанными в ключе.

Концевая точка оглавления приемов вывода соответствует единственному приему вывода. Можно посмотреть список всех теорем базы теорем, для вывода которых (на последнем либо одном из промежуточных шагов) данный прием необходим. Для этого на концевом пункте оглавления нажимается клавиша "л". Список теорем появляется после некоторой паузы. Клавишами "курсор вверх" - "курсор вниз" можно менять выделенную теорему списка. Ее номер перекрашивается в синий цвет.

Если нужно посмотреть сохраненное в архиве базы теорем дерево вывода этой теоремы, нажимается клавиша "курсор вправо". Дальнейшие действия для этого случая описаны ниже в подразделе "Работа с архивом базы теорем". Чтобы вернуться из просмотра дерева вывода в список теорем, нажимается Esc.

Если нужно выяснить, не утерялся ли вывод каких-либо теорем списка, нажимается "Ctr-т". Тестирование может занять сравнительно много времени. По его завершении после номера теоремы, чей вывод по-прежнему достигается, ставится знак "плюс", в противном случае - знак "минус".

Если нужно перейти от выделенной теоремы в первый пункт той ячейки вывода, где она зарегистрирована, нажимается "Ctr-курсор вправо".

Наконец, для возвращения в оглавление приемов вывода нажимается Esc.

Запуск логического вывода в ячейке вывода

Чтобы запустить логический вывод в ячейке логического вывода, нужно зайти в просмотр любой теоремы первого пункта этой ячейки и нажать "л". Процесс вывода на экране не отображается. В некоторых случаях он может занимать достаточно много времени.

Чтобы проконтролировать текущее состояние вывода, нужно нажать "Break" и выйти в отладчик ЛОСа. Поднявшись по цепочке стековых кадров до программы символа "прогрвывод", можно просмотреть значение переменной x5 - списка четверок, представляющих теоремы, полученные на текущий момент. Если нужно посмотреть

текущую обрабатываемую теорему (из нее выводятся следствия), следует выйти на просмотр программы, к которой обратилась программа "прогрвывод". В кадре этой программы переменная x2 имеет своим значением текущую теорему, а переменная x5 - пару, первым элементом которой служит указанный выше список четверок.

По завершении цикла вывода теорем на экран выводится их список. После номера той теоремы, которая уже имелась в базе теорем (не обязательно в данной ячейке вывода) ставится знак "плюс". Клавишами "курсор вверх - кусор вниз" можно выделять теорему данного списка. Номер выделенной теоремы перекрашивается в синий цвет. Для выхода из просмотра списка достаточно нажать "курсор влево". Так как при этом все результаты цикла вывода будут утеряны, нажимать на клавиши курсора следует осторожно.

Чтобы посмотреть историю вывода выделенной теоремы, нажимается "курсор вправо". После этого появляется кадр, в котором выделенная теорема перерисована сверху. Под ней (отделенный горизонтальной чертой) изображен список характеристик, предложенный процедурой вывода. Под списком характеристик располагается название примененного приема вывода, получившего теорему. В качестве такого названия фигурирует текст того пункта оглавления приемов вывода теорем (ветви оглавления программ), который ссылается на программу приема. Наконец, в самом низу приводятся одна либо несколько теорем, являющихся источниками рассматриваемой теоремы. Верхняя из них - основная, остальные - дополнительные.

Основая теорема либо ранее получена в данном цикле вывода, либо является одной из исходных теорем ячейки. Дополнительная теорема может быть взята из любого места базы теорем либо тоже являться ранее полученной теоремой цикла вывода.

Чтобы посмотреть, как была получена основная теорема, снова нажимается "курсор вправо", и т.д. вплоть до источника вывода. На каждом шаге схема прорисовки такая же, как вначале. Для возвращения в предыдущий кадр нажимается "курсор влево". При выходе в общий список теорем нужна осторожность, чтобы следующим нажатием "курсор влево" не потерять результаты вывода.

Если нужно подробнее проанализировать, как работал прием вывода, можно из просмотра его кадра вывода нажать "п". Тогда весь цикл вывода в данной ячейке вывода будет повторен, вплоть до срабатывания данного приема при выводе им рассматриваемой теоремы. В этот момент будет осуществлен выход в отладчик ЛОСа, причем на момент выполнения программы "ктв". Эта аббревиатура расшифровывается как "контрольная точка вывода". Нажатием "Page Up" выходим в программу "регтеор", регистрирующую полученную теорему в списке вывода.Еще одним нажатием "Page Up" попадаем в программу приема вывода на момент регистрации им с помощью процедуры "регтеор" выведенной теоремы. Просматривая в этом кадре отладчика ЛОСа значения программных переменных, можно получить определенную информацию о срабатывании приема вывода. Обычно описанная возможность используется при отладке.

Если после просмотра срабатывания приема через отладчик ЛОСа нужно вернуться к просмотру общего списка выведенных теорем, следует нажать "0" и "Enter". Тогда прерванный цикл вывода будет продолжен, а по его завершении снова окажется прорисован тот кадр, где было нажато "п".

Можно сохранить в буфере базы теорем все полученные в цикле вывода новые теоремы, т.е. теоремы, которые пока отсутствуют в базе теорем. Для этого из просмотра

списка выведенных теорем достаточно нажать "б". Автоматически произойдет переход в просмотр буфера базы теорем: появится пункт оглавления, расположенный внутри буфера и имеющий заголовок "Вывод теорем". Можно выбрать те из теорем буфера, которые представляют интерес, и перенести их в один из концевых пунктов той ячейки логического вывода, где теоремы были выведены. Перенесение выполняется обычными средствами: из просмотра теоремы нажимается Insert, затем находится нужный концевой пункт в ячйке вывода, и на нем снова нажимается Insert. Следует зарегистрировать хэш новых теорем, иначе система не будет их видеть на посоедующих циклах вывода. Для регистрации выбирается любой надраздел ячейки вывода (включая пункт оглавления, через который непосредственно эта ячейка достигается) и нажимается "Ш".

Чтобы быстрее найти нужные теоремы в буфере базы теорем, можно из ветви "Вывод теорем" нажать "т". Появится полный список теорем этой ветви. Непосредственно через данный список отобрать теорему и перенести ее не удастся, но по номеру теоремы в списке легко найти ее в ветви буфера.

Для расчистки буфера базы теорем, достаточно нажать "О" (кир.) из любой точки оглавления базы теорем.

Чтобы выявить теоремы ячейки логического вывода, которые не были выведены, достаточно из просмотра списка выведенных теорем нажать "к". Появится список всех теорем ячейки, причем после номера выведенных теорем будет стоять "плюс", а после номера невыведенных - "минус". Клавиша "курсор влево" вернет в просмотр списка выведенных теорем.

После запуска цикла вывода в ячейке логического вывода могут оказаться утеряны выводы каких-либо теорем этой ячейки либо, наоборот, ранее не выводившиеся теоремы будут выведены. Чтобы распознавать эти события, служит окраска нижней горизонтальной линии под характеристиками теоремы. Если теорема ни разу ранее не выводилась и не выводится сейчас, то цвет этой линии - черный. Если теорема сейчас была выведена, то цвет - зеленый. Если вывод теоремы был утерян в точности на последнем запуске цикла вывода, цвет нижней линии становится синим. Наконец, если вывод утерян уже давно, цвет линии - красный.

Так как пролистывание всех теорем ячейки вывода может оказаться слишком долгим, создан другой способ контроля выводимости теорем. Если из оглавления базы теорем нажать клавишу "л", появляется табло, на котором отображается общая информация о выводе теорем во всех разделах базы теорем. Здесь имеются следующие пункты:

1. Число утерянных на последнем цикле вывода теорем.
2. Общее число теорем, ранее выводившихся, но с утратой выводимости на текущий момент (включая только что утерянные).
3. Число теорем, вывод которых был восстановлен на последнем цикле.
4. Число теорем, которые только что стали выводимыми системой.
5. Число прерванных выводов. Обычно вывод прерывается, если он становится в разы более трудоемким, чем на предыдущем цикле. Иногда прерывание связано с какой-то ошибкой программы, вызывающей выход в отладчик ЛОСа при ее выполнении.

6. Контроль прерванных выводов. При выборе этого пункта система пытается повторно запустить вывод в тех ячейках, где он был прерван. После такого запуска есть шанс, что число прерванных выводов будет меньше, либо их вовсе не останется. Это объясняется тем, что система при серийной многопоточной обработке выводов в базе теорем иногда дает сбои неизвестной природы, которые при повторном запуске отсутствуют.

7. Наибольшие трудоемкости. Приведены пять самых больших величин трудоемкости вывода в ячейках.

8. Наибольшие замедления. Приведены пять самых больших увеличений трудоемкости на последнем цикле серийного вывода (см. ниже).

9. Наибольшие ускорения. Приведены пять самых больших уменьшений трудоемкости на последнем цикле серийного вывода (см. ниже).

10. Наибольшие серии. Приведены пять самых больших длин списков выведенных теорем по всем ячейкам базы теорем.

11. Увеличение серий. Приведены пять самых больших увеличений длин списков выведенных теорем на последнем цикле серийного вывода.

12. Уменьшение серий. Приведены пять самых больших уменьшений длин списков выведенных теорем на последнем цикле серийного вывода.

Чтобы перейти к просмотру теорем либо ячеек вывода, упомянутых в перечисленных пунктах (кроме пункта "контроль прерванных выводов"), достаточно нажать клавишу для буквы, указанной после соответствующего пункта в скобках. Например, для пункта "прерванные выводы" - "п". Чтобы перейти к следующему элементу просматриваемого списка, нужно нажать клавишу "ш". Выйдя на нужную теорему либо ячейку, далее можно проводить необходимый анализ дефекта и его устранение.

Работа с архивом базы теорем

Можно сохранить в архиве базы теорем деревья вывода тех теорем ячейки вывода, которые система уже умеет получать. Это помогает при отладке, если система перестает выводить теорему, ранее ею выводившуюся. Для сохранения нужно запустить цикл вывода теорем в ячейке при помощи клавиши "Л".

Чтобы выйти на дерево вывода теоремы, нужно из просмотра ее в базе теорем нажать "д". Здесь не придется ждать так долго, как при запуске цикла вывода. Кадр, отображающий корневую вершину дерева вывода, будет прорисован сразу. В верхней части экрана расположится запись теоремы, выводимой в данной точке дерева, под ней - список характеристик, еще ниже - ссылка на примененный прием вывода, после которой идет текст того пункта оглавления приемов вывода, где расположен этот прием. Наконец, в самом низу располагаются теоремы, из которых была выведена верхняя теорема кадра. Первая из них - основная, прочие - дополнительные. Всем этим теоремам соответствуют другие вершины дерева вывода. Клавишами "курсор вверх" - "курсор вниз" можно выбрать одну из нижних теорем (выбранная теорема выделяется синим цветом). Для перехода в ее вершину дерева вывода нажимается "курсор вправо". Возвращение - по "курсор влево". В корне дерева вывода клавиша "курсор влево" заблокирована. Чтобы вернуться в просмотр теоремы, придется

нажать Esc. Заметим, что указанным образом можно посмотреть вывод не только теорем, относящихся к данной ячейке вывода, но и любых дополнительных теорем, если их выводы уже были зарегистрированы в архиве базы теорем.

В качестве ссылки на прием вывода, прорисовываемой в кадре просмотра вершины дерева вывода, используется тройка A, m, n. Здесь A - тот логический символ (характеристика теоремы), в программе которого расположен прием. n - номер контрольной точки "прием(n)", выделяющей конец программы приема. m - уровень срабатывания приема.

В дереве вывода теоремы можно выделить главный путь - последовательность основных теорем, прослеживаемых от корня дерева до одной из исходных теорем ячейки вывода. Предусмотрена возможность выхода в отладчик ЛОСа для просмотра подробностей вывода теорем главного пути, а также некоторых дополнительных теорем - тех, которые получены в цикле вывода для данной ячейки вывода. Находясь в вершине дерева вывода, где получается одна из таких теорем T, для выхода в отладчик ЛОСа можно нажать одну из клавиш "ш", "щ", "ъ".

В первом случае произойдет прерывание внутри программы оператора "ктл(N)", соответствующего точке входа в программу приема вывода перед получением теоремы T. Из нее нужно выйти наружу в программу приема вывода. Значением переменной x2 здесь станет основной источник теоремы T.

После нажатия "щ" произойдет прерывание внутри программы оператора "ктд(xi)", соответствующего той точке программы приема вывода, получившего теорему T, где уже выбрана теорема - дополнительный источник вывода. Выйдя наружу из программы оператора "ктд", окажемся в программе приема вывода, а дополнительная теорема будет значением переменной xi.

Наконец, после нажатия "ъ" произойдет прерывание внутри программы оператора "ктв", соответствующего тому месту программы приема вывода, где теорема T уже получена и регистрируется. Обычно внешняя по отношению к программе "ктд" программа - программа оператора "регтеор", выполняющего регистрацию теоремы T в списке результатов вывода. Выйдя из нее наружу, окажемся в программе приема вывода.

Клавиша "ъ" обычно нажимается для проверки того, что теорема по-прежнему выводится. Если выхода в отладчик ЛОСа не произошло, то способность выводить эту теорему утрачена. Для уточнения причин можно нажать "щ" и проследить дальнейшие действия программы. Если даже до момента выбора дополнительной теоремы дело не доходит, можно нажать "ш" и проследить работу приема с самого начала. В совсем затруднительных случаях следует вставить контрольную точку "трассировка(стоп 0)" в программу приема вывода и вести отладку при повторных запусках вывода в ячейке.

Если выход в отлачик ЛОСа состоялся и нужно вернуться в просмотр дерева вывода, нажимаются клавиши "0" и затем - "Enter".

Серийный запуск логического вывода в базе теорем

По мере того, как появляются новые приемы вывода теорем или изменяются старые, может происходить существенное замедление вывода в отдельных ячейках или утрачиваться ранее имевшиеся выводы. Даже изменение приемов основного решателя

может давать тот же эффект, если эти приемы срабатывают или перестают срабатывать во вспомогательных задачах приемов вывода теорем. Не стоит слишком откладывать устранение таких аварий, чтобы не утратить контроль за ситуацией. Однако, для этого необходимо своевременно их обнаруживать. Для такого обнаружения созданы различные виды серийной прокрутки логического вывода, тестирующие вывод во всех ячейках базы теорем или в заданной их части.

Простейшая серийная прокрутка - последовательная. Если нужно протестировать лишь часть базы теорем, то предварительно с помощью "Ctr-1" выделяется начальный пункт тестирования в оглавлении базы теорем, с помощью "Ctr-2" - последний пункт. Оба эти пункта должны располагаться в одном и том же меню. Они могут совпадать. Для полной прокрутки ничего предварительно нажимать не надо. Если используются 2 машины, далее нажимается "Ctr-3", если 3 машины - "Ctr-4". Иначе нажатие пропускается. Запуск прокрутки осуществляется нажатием клавиши "Ctr-л" из любого места оглавления базы теорем.

В процессе последовательной прокрутки на экране будут появляться стартовые теоремы ячеек вывода. Эта смена особой информации не несет и показывает лишь, что система не зависла. При зависании можно либо устраивать внутренний перезапуск программы нажатием клавиши "Esc", либо, если программа вообще была прервана операционной системой, запускать ее заново. В обоих случах цикл будет продолжен с повторной попытки вывода в той же ячейке. Чтобы оборвать цикл вывода, нужно выйти в отладчик ЛОСа нажатием "Break", после чего нажать "Ctr-з" и "Esc". По завершении цикла можно посмотреть результаты, нажав "л" из оглавления базы теорем. Появится табло результатов вывода, рассмотренное выше.

Последовательная прокрутка выводов теорем чрезвычайно трудоемка. Поэтому она практически не используется, а вместо нее применяется параллельная прокрутка.

Чтобы пользоваться параллельной прокруткой, необходимо предварительно подготовить директорию, в которой расположены основные папки GEN, INF, LOS, TCH, TER, TXT логической системы и ее исполняемый файл *logsyst.exe*. Как именно - указано в разделе "Распараллеливание работы системы" первой главы.
Затем можно запускать параллельную прокрутку. Это выполняется из любого места оглавления базы теорем нажатием "Ctr-п". После небольшой паузы экран системы исчезнет, а вместо него в правой части экрана появятся m небольших окон - по числу параллельных потоков. Пауза означает копирование всей текущей версии системы (указанные выше 6 папок) в каждую из папок EXi. Исполняемый файл при этом не копируется (он открыт в текущем потоке). Работа распределяется между потоками примерно поровну, с учетом имеющихся данных о трудоемкости предыдущих выводов в ячейках.

В окне параллельного потока при прокрутке будут прорисованы: в левой части - трудоемкость вывода в текущей ячейке вывода, в правой части - название потока EXi. Главный поток расположен сверху и обозначен $EX0$. Трудоемкости меняются в окне по ходу работы, и таким образом можно проверять, что поток не завис.

По мере выполнения работы, дополнительные потоки исчезают. Когда работу завершает основной поток, он перекрашивается в зеленый цвет. Возвращение в обычный формат его экрана произойдет автоматически после завершения работы всех потоков. Если при этом система обнаружит, что какие-то выводы были прерваны, она прежде всего попытается их повторить, так что в итоговом табло будут отображены

лишь те прерванные выводы, которые при повторной попытке снова были прерваны. Обычно это означает резко возросшую трудоемкость вывода в ячейке и требует специального анализа.

Если потоки остановлены через диспетчер задач, то для восстановления работоспособности системы нужно запустить главный поток - logsyst.exe и сразу же нажать "Break", "Ctr-з", "Esc", "з", "Ctr-ы". Затем закрыть программу и снова зайти в нее.

По мере обучения системы выводу теорем, трудоемкость полного цикла тестирования увеличивается. Время прокрутки становится значительным, даже с использованием восьмипоточного распараллеливания. С другой стороны, отказ от полного тестирования чреват потерей ранее проработанного обучающего материала. При подключении нового приема вывода теорем легко спровоцировать неограниченный либо чрезмерно длительный вывод в отдельных ячейках, либо пустить процесс вывода в ячейке по другому руслу и за счет этого потерять теоремы. Оба явления подсказывают коррекции приемов и являются важным фактором обучения системы. Поэтому пока приходится мириться с длительными прокрутками. Возможно, в конце концов придется разбить все многообразие приемов вывода на различные группы и в каждой группе тестировать их независимо. Трудность заключается в том, что часто для получения теоремы нужна последовательная работа нескольких приемов вывода, и их всех нужно будет относить к одной группе.

Пока для ускорения тестирования и облегчения отладки был создан архив базы теорем. Он позволяет существенно ускорить хотя бы проверку того, что ранее выводившиеся теоремы не утеряны. Ускорение обеспечивается отсечением всех ветвей вывода, не приводящих к теоремам ячейки. Впрочем, это дает не слишком много: основная часть доработки приема вывода приходится на устранение резких замедлений в ячейках из-за спровоцированнной им гиперактивности. Для обнаружения таких замедлений необходима полная прокрутка.

Выше уже говорилось, что для регистрации в архиве базы теорем результатов вывода в отдельной ячейке достаточно запустить вывод в ней клавишей "Л". Чтобы зарегистрировать выводы во всех ячейках, предусмотрена возможность запуска последовательной прокрутки клавишей "Ctr-а (кир.)". При параллельной прокрутке регистрация результатов в архиве базы теорем не предусмотрена.

Сокращенное последовательное тестирование логического вывода в базе теорем, использующее архив базы теорем, запускается нажатием "Ctr-н". Сокращенное параллельное тестирование запускается нажатием "Ctr-у (кир.)". Оба вида тестирования на порядок быстрее, чем без использования архива. Однако, точки резкого замедления не отслеживаются.

Для просмотра данных о результатах сокращенного тестирования нужно из оглавления базы теорем нажать "Л". Появится табло, на котором будет указано количестов невыведенных теорем и потерянных выводов. Также будут указаны пять наибольших замедлений в режиме сокращенного вывода.

На практике сокращенное тестирование оказалось невостребованным. При этом сохраненные в архиве базы теорем деревья вывода весьма полезны при анализе причин потери вывода теоремы.

23.2 Программы, запускающие цикл вывода теорем

Выйти в точку обработки команд, запускающих логический вывод по текущей теореме, можно через пункт "База теорем" - "Программирующий вывод" - "Запуск программирующего вывода по текущей теореме" оглавления программ. После расчистки комментариев к исходной задаче и установке на счетчике шагов интерпретатора ЛОСа значения 1 происходит обращение к процедуре "прогрвывод", которой передается ссылка на текущий концевой терминал оглавления базы теорем. Этот терминал представляет собой первый пункт некоторой ячейки вывода. Вся основная работа по выводу теорем выполняется процедурой "прогрвывод", реализующей цикл обращений к справочнику "прогрвывод". В программах данного справочника и сгруппированы приемы вывода теорем.

Остановимся на процедуре "прогрвывод" несколько подробнее. Обращение к ней имеет вид "прогрвывод(x1 x2 x3)", где x1 - ссылка на концевой терминал оглавления базы теорем, x2 - набор установок на вывод. Процедура реализует цикл логического вывода для теорем терминала x1. Выходной переменной x3 присваивается набор троек (новая теорема - список ее характеристик - блок вывода) для полученных результатов. Блок вывода представляет собой набор информационных элементов, указывающих способ получения теоремы и ранее предпринимавшиеся шаги вывода следствий из нее.

В процессе работы процедуры "прогрвывод" полученные теоремы представляются не тройками, как указано выше, а четверками: (вес - теорема - список характеристик - блок вывода). Вес используется при сканировании. Список этих четверок ниже будем называть списком вывода.

В блоке вывода используются элементы следующих типов:

1. (источник A_1 A_2). A_1 представляет собой левый край той четверки списка вывода, из которой была получена текущая четверка. A_2 - набор $(B_1 \dots B_n)$ либо логический символ B_1, характеризующий способ получения. B_1 - логический символ, являющийся ссылкой на примененный прием вывода. $B_2, \dots, B_n$ - дополнительные теоремы, использованные приемом. Для одного и того же символа характеристики, по которому происходит обращение к справочнику "прогрвывод", различные приемы должны иметь различные символы B_1. Для приемов, относящихся к различным характеристикам, возможны совпадения этих символов.

2. (следствие A). A есть левый край некоторой четверки списка вывода, полученной из данной четверки.

3. (обобщение A_1 A_2). Имел место вывод следствия из текущей теоремы, обобщающий ее в соответствии с характеристикой A_1. A_2 - набор информационных элементов, уточняющих способ обобщения. В частности, A_2 содержит элемент (следствие A_3), где A_3 - левый край четверки для обобщающей теоремы.

4. (прием A). A есть набор ссылок "прием(...)" на вспомогательные приемы, созданные по теореме на период вывода. Обычно такие приемы создаются в специальных ячейках, содержащих не стартовые теоремы, а установку на цикл вывода. В процессе вывода немедленно создаются приемы, редуцирующие новые теоремы с учетом ранее выведенных. По завершении цикла вывода эти приемы удаляются.

5. (титр A). A есть пара (логический символ S - номер контрольной точки "прием(...)"), ссылающая на примененный прием вывода. Этот прием относится к программе символа S, а контрольная точка расположена непосредственно перед обращением к регистрации выведенной теоремы в списке вывода.

6. исключение. Теорема не включается в результат вывода, и новые следствия из нее не выводятся.

7. Плюс. Теорема выведена не как следствие исходных теорем, а получена некоторым специальным образом согласно установке на вывод. Имеется в виду, что у некоторых ячеек вывода первый пункт содержит не стартовые теоремы, а некоторую техническюу установку на цикл вывода теорем.

8. (начало A). Данная теорема является исходной теоремой вывода и A - ссылка "теорема(A_1 A_2)" на нее в базе теорем.

9. регвывод. Пометка теоремы, при получении которой был использован анализируемый прием вывода - см. установку на вывод (регвывод A).

10. выход. Теорема включается в результат, хотя она и была обобщена.

11. результат. Теорема включается в результат, но следствия из нее не выводятся.

12. (характ A). A есть набор характеристик текущей теоремы, если она является одной из исходных теорем вывода.

13. деблок. Уже предпринималось деблокирование приемов, основанных на данной теореме. Имеется в виду следующее. При тестировании вывода следствий исходных теорем ячейки вывода обычно блокируются все приемы, основанные на теоремах ячейки. Иначе при редуцировании эти теоремы "уничтожали" бы самих себя. Однако, после того, как некоторая теорема уже была выведена, специальные операторы, используемые приемами вывода, могут разблокировать основанные на ней приемы. Пометка "деблок" предотвращает повторную разблокировку.

14. склейка. Теорема уже участвовала в склейке нескольких теорем (частных случаев) при выводе их обобщения.

15. (уровень A). A есть уровень срабатывания приема, создавшего данную теорему.

16. (логвывод A). A есть пара (B_1, B_2), где B_1 - набор вхождений левых краев четверок списока вывода, выведенных непосредственно из текущей четверки. B_2 - 0 либо ссылка "теорема(...)" на текущую теорему в базе теорем. Элемент используется при регистрации вывода в архиве базы теорем.

17. (доптеор A). A есть список теорем текущего цикла вывода, использованных при решении вспомогательных задач, обеспечивших вывод текущей теоремы. Дело в том, что при завершающем редактировании выведенной теоремы часто используется вспомогательная задача, которой передается список вывода и которая может его использовать в своих преобразованиях. Список A пополняет список дополнительных теорем, применявшихся при выводе текущей теоремы.

18. функ. Блокировка примерки определения свойства функции на элементарные операции раздела.

Обычно список x2 установок на вывод, передаваемых процедуре "прогрвывод", пуст. Однако, если ячейка вывода вместо стартовых теорем содержит установку на цикл логического вывода, то эта установка конвертируется приемами вывода в набор элементов приводимых ниже типов, который и становится значением переменной x2 в процессе работы процедуры.

1. (стандформа A). Вывод упрощающих тождеств для нормализатора стандартной формы A.

2. (оценка A_1 A_2). В наборе A_1 перечисляются символы операций, рассматриваемых при выводе упрощающих тождеств стандартной формы, в наборе A_2 - оценки их стоимости (десятичные числа).

3. (число A). Обрыв цикла вывода после получения A теорем. A - десятичное число.

4. стандподст. При выводе упрощающих тождеств стандартной формы используются исходные тождества приведения к стандартной форме.

5. (переменные A). При выводе упрощающих тождеств стандартной формы отбрасываются все результаты, имеющие более A переменных.

6. редуцирование. Предпринимается отбрасывание тех теорем, которые изменяются другими выведенными тождествами лио эквивалентностями общей стандартизации.

7. (факторизация A_1 A_2 A_3). Вывод тождеств приведения к заголовку A_1. A_2 - пара (теорема - набор характеристик) для дистрибутивной развертки выражений с заголовком A_1, A_3 - набор пар (теорема - набор характеристик) для упрощающих тождеств, применяемых к заменяемой части.

   В случае вывода тождеств сокращенного умножения берется обобщенная дистрибутивная развертка с двумя либо более сомножителями, имеющими, вообще говоря, более двух слагаемых каждый, а в качестве упрощающих тождеств - тожество $a - a = 0$. Далее предпринимаются попытки применения его к многочленной части первого тождества, чтобы в итоге пропало как можно большее число слагаемых. Таким образом легко "открываются" тождества для разности квадратов, суммы и разности кубов, и ряд других подобных тождеств.

8. начало. Указатель на имевший место цикл создания приемов вспомогательных пакетов вывода.

Выйти на начало программы "прогрвывод" можно через пункт "База теорем" - "Программирующий вывод" - "Процедура ПРОГРВЫВОД" - "Исходная точка" оглавления программ.

Прежде всего, предпринимается обращение к процедуре "блокраздела", блокирующей срабатывания всех приемов, источниками которых служат теоремы текущей ячейки логического вывода. Это необходимо, чтобы при выводе какой-либо теоремы не использовалсь она сама. Фактически, блокировка необходима лишь для проверки того, что система не утратила способности воспроизводить ранее проработанный обучающий материал.

Переменной x4 присваивается содержимое терминала x1, переменной x5 - пустой накопитель списка вывода - набора четверок (вес теоремы - теорема - список ее характеристик - блок вывода). Просматриваются все элементы набора x4, представляющие собой ссылки на теоремы. Для каждой теоремы A в накопитель x5 заносится четверка (0 A B C), где B - список характеристик теоремы, C - блок вывода, состоящий из элементов (характ B), (логвывод (пустоеслово S)), (начало S). Здесь S - ссылка "теорема(D_1 D_2)" на теорему в базе теорем. Она либо совпадает с элементом набора x4, либо отличается от него заголовком.

После перехода через "ветвь 2" идет цикл сканирования списка вывода. Переменной x6, определяющей текущий уровень сканирования, присваивается 0. Далее идет оператор "повторение", к которому будут происходить откаты при сканировании. После него переменной x7 - индикатору пополнения списка вывода на очередном цикле сканирования - присваивается 0.

Просматриваются все четверки x8 списка вывода, вес которых равен x6. Сразу же вес текущей четверки увеличивается на 1. Переменной x9 присваивается теорема четверки x8, переменной x10 - список ее характеристик, переменной x11 - блок вывода. Если в блок вывода входит символ "исключение" либо "результат", то переход к следующей четверке x8.

Далее рассматриваются два случая:

1. Теорема не имеет характеристики "протокол", т.е. является "обычной" теоремой либо квазипротоколом. Тогда предпринимается просмотр ее характеристик x12. Переменной x13 присваивается пара, первым элементом которой служит список вывода, а вторым элементом - индикатор пополнения этого списка при рассмотрении характеристики x12, изначально нулевой. Происходит обращение к справочнику "прогрвывод" на логическом символе, являющемся заголовком характеристики x12.

   Программа справочника "прогрвывод" объединяет группу приемов вывода теорем, активируемых при рассмотрении характеристик с заданным заголовком. Этому справочнику передаются следующие входные данные: x1 - текущая четверка (вес - теорема - характеристики - блок вывода), x2 - теорема, x3 - список ее характеристик, x4 - блок вывода, x5 - пара (список вывода - индикатор изменений), x6 - уровень сканирования, x7 - текущая характеристика. Программа справочника реализует вывод следствий из теоремы, связанный с данной характеристикой, и регистрацию результатов в накопителе x5. Приемы вывода теорем, реализованные в программах справочника, описаны в 9 томе монографии "Компьютерное моделирование логических процессов". На текущий момент их насчитываается свыше 1500. В этой книге мы приведем лишь несколько примеров таких приемов. Если цикл вывода должен быть прерван, то значением R обращения к справочнику служит 2. Если необходим откат к началу цикла сканирования и понижение уровня до нуля, то $R = 3$. Если необходим немедленный переход к анализу следующей четверки списка вывода, то $R = 4$.

   После обращения к справочнику проверяется второй элемент пары x13. Если он равен 1, т.е. список вывода изменился, то индикатор x7 устанавливается на единицу, переменной x5 переприсваивается первый элемент пары x13, а переменной x11 - новая версия блока вывода, расположенная в последнем разряде четверки x8. Если результат обращения к справочнику равен 2, то цикл вывода

обрывается, и происходит откат к переходу через "ветвь 1". Иначе - откат к переходу через "ветвь 4".

2. Теорема имеет характеристику "протокол", т.е. является протоколом базы теорем.

   Протоколы представляют собой термы технического характера, несущие самую разнообразную функциональную нагрузку. Они дают общую характеристику раздела либо отдельных понятий, фиксируют решения об алгоритмизации раздела, задают целевые установки на циклы логического вывода и т.д. Хотя сами протоколы не являются теоремами, запуск циклов логического вывода по ним выполняется точно так же, как по теоремам. Необходимая информация при этом извлекается из базы теорем путем просмотра разделов ее оглавления либо с помощью справочников поиска теорем.

   В случае протокола пара, первым элементом которой служит список вывода, а вторым элементом - индикатор пополнения этого списка, присваивается переменной x12. Для вывода теорем из протокола используется уже не справочник "прогрвывод", а справочник "теоремы". Обращение к нему происходит на логическом символе - заголовке протокола x9. Этот справочник имеет следующие значения своих входных переменных: x1 - текущая четверка (вес - протокол - характеристики - блок вывода), x2 - протокол, x3 - список его характеристик, включающий символ "протокол", x4 - блок вывода, x5 - пара (список вывода - индикатор изменений), x6 - уровень сканирования. Программа справочника реализует вывод теорем, определяемый протоколом, и регистрацию результатов в накопителе x5. Приемы вывода теорем, реализованные в программах справочника "теоремы", можно найти в 9 томе монографии. Смысл значений 2,3,4, выдаваемых справочником "теоремы" в качестве результатов, тот же, что для справочника "прогрвывод".

   Заметим, что новые протоколы, как и новые теоремы, генерируются приемами вывода обоих справочников. В процессе развития базы приемов и тестирования ее на задачах ранее принятые решения об алгоритмизации раздела могут изменяться. Пока эти механизмы не разработаны.

   После обращения к справочнику "теоремы" проверяется второй элемент пары x12. Если он равен 1, т.е. список вывода изменился, то индикатор x7 устанавливается на единицу, переменной x5 переприсваивается первый элемент пары x12, а переменной x11 - новая версия блока вывода, расположенная в последнем разряде четверки x8. Если результат обращения к справочнику равен 2, то цикл вывода обрывается, и происходит откат к переходу через "ветвь 1". Иначе - откат к переходу через "ветвь 4".

Если произошел откат к переходу через "ветвь 4", т.е. справочник "прогрвывод" либо "теоремы" завершил обработку теоремы, то проверяется, равен ли текущий уровень сканирования x6 нулю. Если равен, то предпринимается обращение к процедуре "блоквывода", реализующей такие приемы вывода следствий теоремы, которые не связаны ни с какой конкретной характеристикой теоремы. Этих приемов совсем немного, и в этой книге мы их не приводим.

Входными данными процедуры "блоквывода(x1 x2)" служат: пара x1 (список вывода - индикатор изменений) и текущая четвертка x2 списка вывода.

В нашем случае первому аргументу передается пара, присвоенная непосредственно до обращения переменной x12, второму аргументу - четверка x8. После обращения к процедуре проверяется второй элемент пары x12. Если он равен 1, т.е. список вывода изменился, то индикатор x7 устанавливается на единицу, переменной x5 переприсваивается первый элемент пары x12, а переменной x11 - новая версия блока вывода, расположенная в последнем разряде четверки x8.

Далее происходит откат к переходу через "ветвь 3". Если индикатор изменения x7 равен 1, то текущий уровень сканирования переустанавливается на 0 и происходит откат к оператору "повторение", после которого сканирование списка вывода возобновляется. Если индикатор x7 нулевой, то происходит выбор следующей четверки x8 списка вывода, вес которой равен текущему уровню сканирования.

По окончании сканирования списка вывода - переход через "иначе 2". Если текущий уровень сканирования меньше 6, то он увеличивается на 1, и происходит откат к оператору "повторение" для возобновления сканирования списка вывода на большем уровне. Наконец, по достижении уровня 6 происходит откат к переходу через ветвь 1, где вывод следствий обрывается и начинается обработка списка вывода.

В комментарии (прогрвывод A) к посылкам исходной задачи сохраняется копия списка вывода x5. Она понадобится для учета вывода в архиве базы теорем.

Предпринимается расчистка списка вывода. Прежде всего, после контрольной точки "прием(5)" выполняется отбрасывание теорем, имеющих альтернативную версию, у которой консеквент тот же самый, а список антецедентов - подмножество списка антецедентов данной теоремы. Если теоремы совпадают, то одна из них удаляется, а ее характеристики присоединяются к характеристикам оставшейся теоремы. Удаление теоремы состоит в том, что вместо ее четверки в списке вывода помещается 0. По завершении цикла - переход через "ветвь 1" к контрольной точке "прием(97)".

Здесь предпринимается отбрасывание теорем, изменяемых выведенными тождествами и эквивалентностями общей стандартизации. Рассмотрим этот цикл подробнее. Текущая четверка списка вывода присваивается переменной x7. Переменной x8 присваивается теорема четверки, переменной x9 - список характеристик.

Если в списке x9 имеется характеристика "нормализация(...)", указывающая, что x8 - тождество общей стандартизации, то проверяется, что либо оценка сложности заменяющего терма меньше оценки сложности заменяемого, либо число вхождений переменных в заменяющий терм меньше чем в заменяемый. Просматриваются другие теоремы T списка вывода, имеющие заголовок "длялюбого", у которых все заголовки характеристик принадлежат списку "антецедент", "спуск", "упрощение", "группировки", "исклтерм", "нормализация", "общнорм". Если у x8 есть существенные посылки, то у T они тоже должны быть. Не рассматриваются теоремы T, у которых заголовок ссылки на примененный прием - "поглощает" либо "Сокращ". Предпринимается попытка применить тождество x8 к консеквенту теоремы T. Если это удается, то теорема T отбрасывается (ее четверка заменяется на 0).

Если в списке x9 есть характеристика "кванторнаясвертка(...)", указывающая, что x8 - тождество для замены кванторного утверждения на бескванторное, то просматриваются другие теоремы T списка вывода. Если какая-либо из них имеет такую же характеристику, причем единственную, то сравниваются заменяемые термы обеих

теорем. Если эти термы - кванторные импликации, отличающиеся переобозначением без отождествлений своих свободных переменных, причем оценка сложности заменяющего терма теоремы T не меньше, чем у теоремы x8, то теорема T отбрасывается.

По завершении цикла из списка вывода удаляются все нули, и переход через "ветвь 1". Здесь заменяются на ноль все четверки списка вывода, у которых среди характеристик имелся символ "Исключение". Далее - снова переход через "ветвь 1". Здесь происходит расчистка протоколов "нормзаголовок", созданных в цикле вывода. Отбрасывается тот протокол, чье множество заголовков - подмножество заголовков другого протокола. Снова происходит исключение нулей из списка вывода.

Далее - еще один переход через "ветвь 1". Здесь рассматриваются два случая, когда теорему, обобщенную в цикле вывода, целесообразно сохранить. В первом случае теорема имеет характеристики "группировки" и "нормализация(...)", а число ее переменных не более 3. Элемент "исключение" ее блока вывода заменяется на "выход", и характеристики полагаются состоящими из единственного символа "перестановки". Во втором случае теорема имела характеристики "группировки", "варьир(...)" и не имела характеристики "нормализация(...)". Действия аналогичны предыдущим; в характеристиках сохранится только "варьир(...)".

По завершении перечисленных коррекций списка вывода, после контрольной точки "прием(100)", реализуется выдача результата. Просматривается список вывода. Отбираются такие его четверки x7, для которых выполнены следующие требования:

1. Блок вывода не содержит элементов "исключение", "Исключение".
2. Либо блок вывода имеет элемент (источник ...), либо он имеет элемент "Плюс".
3. Если теорема имеет не менее четырех переменных, была обобщена и ее блок вывода не имеет элемента "выход", то ее источник в списке вывода не был обобщен.

У этих четверок отбрасывается первый элемент (вес), и составленный из них список x6 выдается как результат вывода. Предварительно расчищается буфер фильтрации, выполнявший блокировку приемов, основанных на теоремах ячейки вывода.

23.3 Вспомогательные процедуры, используемые приемами вывода теорем

Приемы вывода теорем реализованы на ЛОСе. Чтобы упростить создание и чтение их программ, было создано множество вспомогательных процедур. Перечислим основные из них. Выйти на список этих процедур и их программы можно через пункт "База теорем" - "Программирующий вывод" - "Вспомогательные процедуры логического вывода" оглавления программ.

Процедура "смтеор"

При выводе следствий теоремы бывает необходимо привлечение дополнительных теорем. Наиболее часто используемый способ их поиска - справочники поиска теорем.

Входным данным x1 любого такого справочника служит одноэлементный набор, образованный изначально пустым накопителем ссылок "теорема(A_1 A_2)" на теоремы из базы теорем, отбираемые согласно определенному принципу по текущему логическому символу, на котором происходит обращение к справочнику. Список справочников поиска теорем, созданных на текущий момент, приведен в 9 томе монографии.

Прием вывода теорем обращается к справочникам поиска теорем через процедуру "смтеор(x1 x2 x3 x4)". Здесь x1 - набор логических символов, представляющих собой названия справочников поиска теорем, либо единственный такой символ. x2 - логический символ. Процедура перечисляет теоремы x3, определяемые справочниками списка x1 по логическому символу x2. Одновременно переменной x4 присваивается список характеристик теоремы x3, к которому добавлена ссылка "теорема(A_1 A_2)" на теорему x3 в базе теорем.

Процедура "тождвывод"

Эта процедура - наиболее часто используемое правило вывода, заключающееся в тождественном либо эквивалентном преобразовании фрагмента одной теоремы при помощи другой теоремы. Обращение к ней имеет вид "тождвывод(x1 x2 x3 x4 x5)", где x1 - кванторная импликация, x2 - вхождение в него некоторого подтерма, x3 - тождество либо эквивалентность, x4 - набор дополнительных установок на вывод. Перечисляются варианты x5 вывода следствий из x1 путем унификации терма x2 с одной из частей тождества либо эквивалентности x3.

Используются следующие дополнительные установки на вывод:

1. направл(A). Символ A ("первыйтерм" либо "второйтерм") указывает направление применения тождества либо эквивалентности x3.

2. модификатор. Указание на ввод при преобразованиях вспомогательной переменной - модификатора (дополнительный параметр корневой ассоциативно - коммутативной операции каждой из частей тождества x3).

3. существпосылки. Блокируется невырожденная подстановка в существенные посылки теорем x1, x3.

4. (переменная x A). x есть переменная утверждения x1, участвующая в унификации. После каждой текущей унификации, порождающей очередную версию следствия, элемент A заменяется на терм, подставляемый вместо переменной x. Таким образом, данный элемент играет роль дополнительного выходного канала, позволяющего контролировать варианты унификации.

5. (перем x A). Аналогично предыдущему, но x есть переменная утверждения x3.

6. (фикс A). A есть набор переменных терма x1, вместо которых подстановка при унификации не выполняется.

7. новаяпеременная. Блокировка ввода новых переменных при унификации.

8. варунифик. Варьирование преобразующего тождества либо эквивалентности оператором "варунифик".

9. (пересекаются x y). Термы, унифицированные с переменными x, y теоремы x1, имеют общую переменную.

10. разбиение. При унификации ассоциативно-коммутативной операции f от двух переменных x, y с операцией $f(z, t_1, t_2)$, где z - переменная, происходит разбиение z в "произведение" двух новых переменных z_1 и z_2, относимых к x и y.

Несмотря на обилие типов установок, обычно используется только элемент "(направл ...)".

Процедура "выводпосылки"

Эта процедура реализует еще одно правило вывода - обобщенную транзитивность импликации. У первой кванторной импликации выделяется некоторый антецедент, который унифицируется с консеквентом другой кванторной импликации. Процедура используется существенно реже, чем процедура "тождвывод".

Обращение к процедуре имеет вид "выводпосылки(x1 x2 x3 x4 x5)". Здесь x1 - кванторная импликация, x2 - вхождение ее антецедента, x3 - другая кванторная импликация. x4 - набор дополнительных установок на вывод следствий. Перечисляются варианты x5 вывода следствий из x1 путем унификации антецедента x2 с консеквентом импликации x3.

Пока был введен единственный тип установки набора x4 - элемент (разбиение A). Он означает, что переменные антецедентов импликации x3 после унификации должны обеспечивать разбиение набора A переменных антецедента x2.

Выйти на начало программы "выводпосылки" можно через пункт "База теорем" - "Программирующий вывод" - "Процедура ВЫВОДПОСЫЛКИ" оглавления программ.

Процедура "нормтеорема"

Прежде чем выдавать очередную теорему в качестве результата, прием логического вывода обычно ее упрощает и стандартизирует. Для этой цели используются процедуры "нормтеорема" и "Нормтеорема". Начнем с рассмотрения первой из них.

Обращение к процедуре имеет вид "нормтеорема(x1)". Она представляет собой операторное выражение, значением которого служит результат обработки теоремы x1 вспомогательной задачей на преобразование.

Выйти на начало программы "нормтеорема" можно через пункт "База теорем" - "Программирующий вывод" - "Процедура нормтеорема" оглавления программ.

Прежде всего, удаляются комментарии к посылкам исходной задачи, сохраняющие результаты обращений к пакетным операторам. Затем вводится задача на преобразование x2, имеющая единственную посылку "истина" и условие x1. Цели этой задачи - "упростить", "нормтеорема", "редуцирование". Цель "нормтеорема" определяет особый режим обработки, ориентированный на преобразование теоремы к "стандартному" виду. Для такой обработки приходится создавать специальные приемы ГЕНОЛОГа, относящиеся к самым разным предметным областям. Эта же цель блокирует ряд слишком активных общих приемов, чтобы они не портили теорему. В частности, блокируются перестановки частей равенств и эквивалентностей, что позволяет сохранить исходные указания ориентации преобразований, отраженные в сопровождающих теорему характеристиках. Цель "редуцирование" осталась от старых версий вывода теорем. Она указывала на режим упрощения теоремы другими

ранее выведенными теоремами, чтобы исключать избыточность. В настоящей версии сохранились отдельные приемы, учитывающие данную цель.

После того, как задача x2 создана, запускается процесс ее решения. Максимальный уровень равен 4. Ответ задачи выдается в качестве результата.

Процедура "Нормтеорема"

Процедура "Нормтеорема" представляет собой усиление процедуры "нормтеорема", позволяющее передавать задаче на преобразование дополнительные указания. Кроме того, эта процедура позволяет при обработке новой теоремы пользоваться теоремами, выведенными до нее в том же самом цикле логического вывода. Никаких приемов по таким предшествующим теоремам на этот момент не создается. Заметим, что обычно используется процедура "нормтеорема", и лишь в особых случаях прием обращается к ее усилению.

Обращение к процедуре имеет вид "Нормтеорема(x1 x2)". Это операторное выражение, значением которого служит результат обработки теоремы x1 вспомогательной задачей на упрощение, в комментарии которой заносятся элементы набора x2.

В наборе x2 выделим, прежде всего, элемент (смтеор A_1 A_2 A_3). Он представляет собой комментарий к задаче на преобразование, имеющей цель "нормтеорема", включающий дополнительную обработку теоремы за счет других теорем, уже выведенных в текущем цикле вывода. Здесь A_1 - набор четверок (вес - теорема - характеристики - блок вывода), перечисляющий ранее выведенные оператором "прогрвывод" теоремы. A_2 - терм либо логический символ, уточняющий тип обработки. Пока используется только символ "описатель". A_3 - вначале пустой список, который будет заполняться процедурой "Нормтеорема" парами (заголовок преобразуемого терма - список пар (теорема - направление преобразования)). Эти пары соответствуют ситуациям типа общей стандартизации и используются общим приемом решателя, когда он начинает работать с задачей на преобразование теоремы.

Кроме того, в наборе x2 могут появляться произвольные другие комментарии. Например, используется комментарий "свертка", указывающий тенденцию к сокращенной переформулировке термов, а также комментарий "числовойатом", используемый при обработке соотношений с невырожденными числовыми атомами. Ряд комментариев набора x2 дополняет комментарий (смтеор ...). Так, комментарий "общнорм" подключает к преобразованиям эквивалентности общей стандартизации, выведенные в том же цикле логического вывода; комментарий "упрощдн" подключает тождества, уменьшающие оценку сложности.

Как и программа оператора "нормтеорема", рассматриваемая программа прежде всего удаляет комментарии к посылкам исходной задачи, сохраняющие результаты обращений к пакетным операторам. Затем вводится задача на преобразование x3, имеющая единственную посылку "истина" и условие x1. Цели этой задачи - "упростить", "нормтеорема", "редуцирование". Список ее комментариев - x2.

Если имеется комментарий (смтеор A_1 описатель пустоеслово), то предпринимается заполнение последнего (пустого) элемента этого комментария.

Просматриваются четверки x5 набора A_1. Второй элемент такой четверки - теорема T, третий - список ее характеристик. Рассматриваются следующие случаи:

1. Теорема T либо имеет характеристику "нормализация(N)", либо в набор x2 входит символ "общнорм", а теорема имеет характеристику "общнорм(N)", либо в набор x2 входит символ "упрощдн", а теорема имеет характеристику "упрощение(N)", либо теорема имеет характеристику "описатель(N)", либо она имеет характеристику "сокращ(N)". В случае характеристики "сокращ" проверяется, что теорема не заменяет бесповторный терм на небесповторный. В этой ситуации теорема и направление преобразования N регистируются в структуре данных A_3 комментария (смтеор ...) так, как указано выше в описании этого комментария.

2. Теорема T имеет комментарий "числатом", причем правая часть ее тождества не имеет переменных. Тогда она вместе с направлением замены "слева направо" регистрируется в структуре данных A_3 комментария (смтеор ...).

3. Теорема T имеет комментарий (спуск ...). Ее консеквент не имеет заголовка "не". Тогда она регистрируется в структуре данных A_3 таким образом, как если бы представляла собой эквивалентность своего консеквента константе "истина".

4. Теорема T имеет комментарий "пример". Если ее консеквент B не имел заголовка "не", то он заменяется на эквивалентность $B \leftrightarrow$ истина. Если он имел вид "не(C)", то заменяется на $C \leftrightarrow$ ложь. Результат замены регистрируется в структуре данных A_3.

По окончании цикла просмотра четверок x5 - переход через "иначе 3". Здесь корректируются теоремы структуры данных A_3, имеющие антецеденты "функция(f)". Предпринимается попытка перейти от них к теоремам с функциональными переменными.

По окончании формирования структуры данных A_3 - обращение к решению задачи x3 с максимальным уровнем 4. Ответ выдается как результат обработки теоремы.

Процедура "регтеор"

После того, как прием вывел новую теорему и предпринял ее упрощение, эта теорема должна быть зарегистрирована в списке вывода. Необходимо определить ее характеристики, а также проверить неизбыточность. Эти действия реализуются процедурой "регтеор", обычно завершающей программу приема вывода теорем.

Обращение к процедуре имеет вид "регтеор(x1 x2)", где x1 - новая теорема, x2 - набор информационных элементов, уточняющих способ регистрации теоремы. В этом наборе допускаются элементы следующих типов:

1. (характ $A_1 \ldots A_n$). Отбираются только те предлагаемые характеризатором характеристики теоремы x1, заголовки которых попадают в список $A_1, \ldots, A_n$.

2. характ. Использование характеризатора отменяется. Характеристиками теоремы становятся только те, которые явно указаны в элементе (характеристика ...); см. ниже.

3. (входит $A_1 \ldots A_n$). Проверяется,что характеризатор получил характеристику с заголовком, принадлежащим списку $A_1, \ldots, A_n$. Иначе теорема не регистрируется.

4. (источник A). Указание способа получения теоремы. В сочетании с логическим символом, к программе которого относится прием вывода, представляет собой разновидность ссылки на этот прием. Элемент A - логический символ s либо набор вида $(st_1 \dots t_m)$, где s - логический символ; $t_1, \dots, t_m$ - дополнительные теоремы, использованные при выводе. Символ s называем типом источника теоремы. Чтобы исключить чрезмерную активность приемов вывода, обычно в начале их программы располагается проверка неиспользования в текущей линии вывода источников заданных типов.

5. (характеристика $A_1 \dots A_n$). Характеристики $A_1, \dots, A_n$ дополнительно заносятся в список характеристик теоремы x1.

6. обобщение. Теорема представляет собой обобщение источника по текущей характеристике.

7. вспомприемы. По теореме x1 создаются вспомогательные приемы, необходимые для продолжения цикла вывода. Этот элемент используется только в специальных циклах вывода, стартующих не с обычной теоремы, а с установки на цикл вывода. Например, при выводе упрощающих тождеств какой-либо стандартной формы.

8. новтеор. Проверяется, что новая теорема не является частным случаем ранее выведенной в данном цикле теоремы.

9. исключение. Та теорема, по которой была выведена теорема x1, исключается из дальнейших выводов и не включается в результат.

10. (смхаракт A). Проверяется, что характеризатор усматривает характеристику A теоремы x1, после чего список характеристик полагается состоящим из характеристики A, к которой добавляются все характеристики с заголовками, перечисленными в элементе (характ ...).

11. контроль. Дополнительный контроль неизбыточности теоремы.

12. (выход $A_1 \dots A_n$). Прием выполнил обобщение теоремы, однако исходная теорема включается в итоговый список, если она имела характеристику с заголовком, принадлежащим списку $A_1, \dots, A_n$.

13. сохр. Прием выполнил обобщение теоремы, однако вывод следствий из обобщенной теоремы продолжается.

14. выход. Прием выполнил обобщение теоремы, однако обобщенная теорема включается в итоговый список теорем.

15. (непересек $A_1 \dots A_n$). Проверяется, что среди характеристик новой теоремы нет ни одного из элементов $A_1, \dots, A_n$.

16. (направл A). Проверяется наличие характеристики с заголовком A, причем отбрасываются все характеристики, у которых направление замены отличается от направления характеристики A.

17. (примечание $A_1 \dots A_n$). $A_1, \dots, A_n$ суть элементы, указывающие способ обобщения. См. элемент (обобщение ...) блока вывода.

18. (списокпеременных $A_1 \ldots A_n m$). Проверяется, что если все характеристики новой теоремы имеют только заголовки $A_1, \ldots, A_n$, то число переменных теоремы меньше m.

19. (исходный $A_1 \ldots A_n$). $A_1, \ldots, A_n$ суть исходные характеристики новой теоремы, передаваемые характеризатору.

20. (исключ A). Если среди характеристик исходной теоремы имеется элемент $A(n)$, где направление n не совпадает с направлением текущей характеристики, то исходная теорема не помечается в своем блоке вывода элементом "исключение", иначе - помечается.

21. антецеденты. Контроль отсутствия чрезмерно сложных антецедентов теоремы.

22. смпропорц. Контроль отсутствия ранее выведенного эквивалентного соотношения пропорциональности. Рассматриваются теоремы, имеющие соотношение пропорциональности в консеквенте и одно соотношение - в антецедентах.

23. результат. Теорема включается в результирующий список, но следствия из нее не выводятся.

24. стандхаракт. Специальная характеризация для тождеств стандартной формы.

25. Исключение. Теорема не включается в результат, но следствия из нее выводятся.

26. (Напр A). Игнорируются все характеристики с направлением замены A.

27. теоремаприема. Отменяется проверка отсутствия такой же теоремы в накопителе, так как основная смысловая нагрузка приходится на ее характеристики, связанные с созданием приема специального типа.

28. (смхаракт). Разрешается регистрация теоремы x1, даже если в накопителе уже есть такая же теорема, но у нее нет характеристики с заголовком A.

Выйти на начало программы "регтеор" можно через пункт "База теорем" - "Программирующий вывод" - "Вспомогательные процедуры логического вывода" - "Процедура РЕГТЕОР" оглавления программ.

Процедура "новтеор"

Обращение к процедуре имеет вид "новтеор(x1)", где x1 - четверка (вес - теорема - список ее характеристик - блок вывода) для теоремы, выведенной приемом логического вывода. Процедура проверяет, что эта теорема не не является частным случаем другой теоремы из списка вывода. Если не является, то выход по "истина", иначе - по "ложь". Если некоторая старая теорема из списка вывода оказывается частным случаем теоремы x1, то ее блок вывода сопровождается элементом "исключение".

Выйти на начало программы "новтеор" можно через пункт "База теорем" - "Программирующий вывод" - "Вспомогательные процедуры логического вывода" - "Процедура НОВТЕОР" оглавления программ.

Процедура "нормантецеденты"

В некоторых случаях обращение к процедурам "нормтеорема", "Нормтеорема" нежелательно, так как они могут испортить специальным образом созданный консеквент. Однако, в этих ситуациях часто бывает приемлемым провести общую стандартизацию хотя бы антецедентов. Такая стандартизация выполняется процедурой "нормантецеденты".

Обращение к процедуре имеет вид "нормантецеденты(x1 x2)", где x1 - список утверждений, представляющих собой антецеденты формируемой кванторной импликации. x2 - список свободных переменных консеквента. Процедура реализует программное выражение, значением которого служит упрощенный и стандартизированный список x1. По мере возможности, отбрасываются антецеденты, свободные переменные которых не входят в список x2.

Выйти на начало программы "нормантецеденты" можно через пункт "База теорем" - "Программирующий вывод" - "Вспомогательные процедуры логического вывода" - "Процедура "нормантецеденты" " оглавления программ.

Процедура "источники"

Блок вывода каждой теоремы списка вывода, кроме стартовых теорем, содержит элемент (источник v A), определяющий ту теорему, из которой она была выведена, а также указатель A способа вывода. Иногда бывает необходимо получить всю последовательность таких указателей A, начинающуюся со стартовой теоремы. Для этого служит процедура "источники". Обращение к ней имеет вид "источники(x1)". Здесь x1 - блок вывода некоторой теоремы. Обращение представляет собой программное выражение, а значением его служит набор указателей способов получения теорем вдоль цепочки вывода теоремы x1. Эта теорема соответствует концу набора. Программа процедуры невелика. Она просто прослеживает цепочку ссылок по элементам (источник ...).

Надобность в данной процедуре возникает в связи с необходимостью обрыва слишком длинных или малополезных цепочек вывода. В начале программы почти каждого приема вывода проверяется отсутствие тех или иных конкретных элементов в цепочке указателей способа вывода.

Процедура "Источники"

В отдельных случаях бывает нужна не цепочка указателей способа вывода, а цепочка самих блоков вывода. Тогда применяется процедура "Источники". Обращение к ней имеет вид "Источники(x1)", где x1 - блок вывода некоторой теоремы. Значением этого программного выражения служит набор блоков вывода теорем вдоль цепочки вывода теоремы x1. Эта теорема соответствует концу набора. Программа процедуры аналогична предыдущей.

Процедура "сокращантецеденты"

Процедура предпринимает попытку явного разрешения антецедентов выведенной теоремы относительно некоторых переменных. В каких случаях это целесообразно

делать, решает она сама. Предполагается, что процедура объединит в своей программе несколько различных приемов, выполняющих указанное действие. Обращение к процедуре имеет вид "сокращантецеденты(x1)", где x1 - новая теорема. Пока в процедуре представлен единственный прием. Если консеквент теоремы имеет вид "значение(t, x)", где терм t не содержит переменной x, то предпринимается попытка явного разрешения ее антецедентов относительно x. В случае удачи создается теорема с измененным списокм антецедентов и старым консеквентом. После обработки оператором "нормтеорема" она выдается в качестве результата.

Процедура "исклант"

Для отбрасывания избыточных антецедентов новой теоремы служит процедура "исклант". Обращение к ней имеет вид "исклант(x1)", где x1 - теорема.

Процедура "Исклант"

Эта процедура дополняет процедуру "исклант". Обращение к ней имеет вид "Исклант(x1)", где x1 - теорема.

Процедура "сокращант"

Процедура "сокращант(x1)" удаляет у кванторной импликации x1 те антецеденты, которые являются следствиями других антецедентов. Однократно просматривается список антецедентов теоремы x1. Для каждого антецедента, не используемого для сопровождения по о.д.з., создается вспомогательная задача на доказательство его из остальных антецедентов. В случае успеха этот антецедент заменяется на константу "истина". По завершении цикла все тождественно истинные антецеденты отбрасываются, и формируется итоговая кванторная импликация.

Процедура "исхаракт"

Значением выражения "исхаракт(x1)" служит список характеристик той стартовой теоремы цикла вывода, из которой по цепочке выводов была получена теорема с блоком вывода x1. Этот список определяется при помощи процедуры "Источники".

Процедура "исключтеор"

Обращение к процедуре имеет вид "исключтеор(x1)", где x1 - четверка из списка вывода. В этой четверке, а также во всех достижимых из нее переходами через элементы (следствие ...) блоков вывода четверках регистрируется элемент "исключение". Таким образом, теорема четверки x1 и все выведенные из нее теоремы не включаются в результат вывода. Следствия из них далее не выводятся.

Процедура "нормтеор"

Процедура "нормтеор(x1)" выполняет ослабленную стандартизацию кванторной импликации x1. Она применяется, чтобы не разрушить специальным образом сконструированный консеквент. Антецеденты упрощаются задачами на преобразование, а консеквент - только нормализаторами общей стандартизации и проверочными операторами. Сначала реализуется цикл упрощений антецедентов. Используемые для этого задачи на преобразование имеют цели "упростить", "редуцирование". Уровень

обращения к ним равен 4. Затем реализуется цикл исключения переменных x, для которых имеется антецедент вида $x = t$, где t не содержит x. Наконец, просматриваются подтермы результирующей теоремы, и в случае выражения реализуется обработка их нормализаторами общей стандартизации относительно утверждений их контекста, а в случае утверждения - предпринимаются попытки усмотреть их истинность либо ложность при помощи проверочных операторов. В случае успеха подставляются константы "истина" либо "ложь". Блокируются попытки разрушить антецеденты, указывающие типы данных для переменных теоремы. Обработка продолжается, пока происходят изменения. Затем теорема обрабатывается операторами "стандупорядочение" и "нормлог".

Процедура "деблок"

По мере вывода теорем можно снимать блокировку применения тех приемов, которые основаны на только что выведенных теоремах текущей ячейки вывода. Это необходимо для исключения заведомо избыточных либо не стандартизированных теорем. Данная работа выполняется обращением к оператору "деблок(x1)". Здесь x1 - пара (список вывода - индикатор его изменений), являющаяся значением переменной x5 в программах приемов вывода.

Процедура "блокраздела"

В некоторых случаях прием вывода может временно отключить блокировку приемов, основанных на теоремах, относящихся к текущей ячейке вывода. Вообще, эти блокировки нужны лишь для проверки того, не утратила ли система способность выводить теоремы, которым она уже была обучена. В режиме поиска новых теорем они несущественны. Тем не менее, тестирование аппарата логического вывода необходимо, чтобы при продолжении обучения не утратить достигнутого. Причины таких утрат могут быть совершенно мелкими и нелепыми, но если их во-время не найти и не устранить, то с течением времени придется заново прорабатывать значительную часть ранее проработанного материала. Это заставляет аккуратно относиться к блокировкам и пользоваться рядом вспомогательных процедур для их включения либо отключения. Процедура "блокраздела(x1 x2)" выполняет блокировку приемов, источниками которых служат теоремы той ячейки вывода, к которой относится терминал оглавления базы теорем по ссылке x1. Если x2 не равно 0, то блокировка не распространяется на теоремы самого терминала x1. Программа процедуры "блокраздела" не представляет особого интереса, и подробно излагаться не будет. Она просматривает теоремы ячейки вывода и основанные на них приемы, используя описанные ранее структуры данных базы теорем. Напомним, что собственно блокировка либо разблокировка приемов выполняется непосредственно реализованными операторами ЛОСа "трассировка(фильтр ...)". Для ссылок на приемы используются пары "логический символ - номер узла теоремы приема". Все приемы, характеризуемые такой парой, вне зависимости от их заголовка, оказываются заблокированы одновременно. Данные пары записываются в специальные регистры интерпретатора, которые можно расчистить операторами той же серии.

Процедуры "унификация" и "Унификация"

Процедура "унификация" перечисляет наборы термов, подстановка которых вместо заданных переменных отождествляет заданные пары термов. При поиске подстанов-

ки разрешались некоторые простейшие преобразования термов и ввод дополнительных переменных. За счет этого, результат унификации становится неоднозначным, в отличие от классического случая унификации, рассматриваемого в учебниках по математической логике.

В отдельных случаях объяснение того, как происходит вывод теоремы, потребовало усиления процедуры "унификация", которой разрешено использовать тождества из базы теорем для отождествления заголовков унифицируемых термов. Обращение к такому усилению имеет вид "Унификация(x1 x2 x3 x4)". Здесь x1 - набор установок на унификацию, такой же, как у оператора "унификация". x2 - набор переменных, вместо которых осуществляется унифицирующая подстановка. Предпринимается попытка унификации, быть может, с использованием тождеств, извлекаемых из базы теорем. При необходимости в наборе x1 уточняется способ отбора таких тождеств, хотя возможны действия по умолчанию. Переменной x3 присваивается набор подставляемых вместо переменных набора x2 термов, переменной x4 - набор утверждений, которые должны быть истинны для корректности унификации. Оператор перечисляющий.

Программа оператора попросту обращается к оператору "унификация", списку установок на унификацию которого передается дополнительный элемент (Унификация пустоеслово). Во втором разряде этого элемента будет создаваться набор утверждений x4, сбрасываемый при каждом откате.

Глава 24

Приемы вывода теорем

Основную работу по выводу теорем - как программирующему, так и в целях развития теории, выполняют приемы справочника "прогрвывод". Хотя в настоящее время их уже насчитывается более 1500, попытки объяснить, как была открыта та или иная теорема или как она была адаптирована для создания по ней приемов заставляют постоянно увеличивать это число. Вероятно, до стабилизации еще очень далеко, хотя в большинстве случаев приемы вывода теорем имеют "общелогический" характер и не связаны ни с какой конкретной предметной областью. Впрочем, к разряду общелогических пришлось отнести ряд исключительно часто используемых "предметных" понятий - функции, множества, координаты, числовые атомы и т.п. Несмотря на то, что работа по обучению системы вывода теорем находится лишь на начальной стадии, уже сейчас она способна находить десятки, а иногда и сотни полезных следствий исходной теоремы. Создаются также и явно бесполезные следствия, которые массово отсекаются на этапах примерки и доводки приемов. К тому же всегда можно, доработав прием вывода теорем, добиться исключения бесполезных следствий. Чаще всего они являются результатом пробелов в решателе, не усматривающем возможности упростить выведенную теорему или свести ее к константе "истина".

Кроме справочника "прогрвывод", новые теоремы выводит также справочник "теоремы", работающий с протоколами базы теорем. Приемы обоих справочников изложены в 9 томе монографии "Компьютерное моделирование логических процессов". Каждый прием сопровождается рассмотрением примера. Фактически, описывается ЛОС-программа приема, с сохранением нумерации программных переменных, и по ходу изложения указываются текущие значения таких переменных для рассматриваемого примера.

В общем-то, основную трудность при проработке программирующего вывода представляет даже не создание приема для отдельного перехода от "старых" теорем к новой. Такой прием, при указании исходной и дополнительных теорем, а также результата вывода, нетрудно придумать и самостоятельно. Сложнее находить всю цепочку переходов от стартовой теоремы ячейки вывода к той теореме, открытие которой нужно объяснить. Ее приходится оптимизировать так, чтобы каждый переход был естественным и простым, не требующим сколь-нибудь значительного перебора. В частности, в данной цепочке должны совершенно отсутствовать "провалы и пропасти", когда в доказательстве какой-либо теоремы непонятно откуда вдруг возникают различные вспомогательные объекты. Чтобы определить такую цепочку, иногда приходится существенно пополнять базу теорем и список понятий. Впрочем, хотя и не с первого раза, простое и естественное объяснение хода рассуждений, приводящего

к нужной теореме, почему-то всегда находится. Более того, часто возникает впечатление, что это объяснение единственно возможное. Соответственно, и те приемы вывода, которые накапливаются в системе, вряд ли можно как-то сильно изменить. Хотя обобщить и оптимизировать - наверняка можно и нужно.

24.1 Примеры приемов вывода теорем

Приведем несколько примеров, иллюстрирующих работу системы вывода теорем. Сначала будут даны упрощенные описания соответствующих процедур, а в конце списка приведены несколько полных описаний, взятых из 9 тома монографии.

1. Обобщение сверточного тождества путем ввода бесповторного дополнительного параметра операнда.

   Рассмотрим следующее тождество:

   $\forall_{abc}(a - \text{число } \& \ b - \text{число } \& \ c - \text{число } \& \ 0 < a \ \& \ 0 < b \to a^c b^c = (ab)^c)$

   Одна из его характеристик имеет вид "свертка(второйтерм)", т.е. в направлении слева направо тождество может быть использовано для сжатой перезаписи выражения. Если использовать вспомогательное тождество

   $\forall_{acd}(a - \text{число } \& \ c - \text{число } \& \ d - \text{число } \& \ 0 < a \to (a^d)^c = a^{cd})$,

   то нетрудно вывести следствие, обобщающее первое тождество относительно характеристики "свертка(второйтерм)":

   $\forall_{abcd}(a - \text{число } \& \ b - \text{число } \& \ c - \text{число } \& \ d - \text{число } \& \ 0 < a \ \& \ 0 < b \to a^{cd} b^c = (a^d b)^c)$

   При получении его мы подставили вместо переменной a первого тождества выражение a^d и воспользовались вторым тождеством.

   Такой обобщающий переход нетрудно сформулировать в виде следующего приема программирующего вывода. Если в заменяемой части A тождества свертки встречается выражение $f(x,t)$, где x - переменная, не имеющая других вхождений в A, то при помощи специального справочника поиска теорем находятся всевозможные тождества вида $f(g(u,v),w) = f(u,h(v,w))$, где u,v,w - переменные, h - коммутативно, g имеет единицу по переменной v. Приемы данного справочника создаются заблаговременно, при просмотре базы теорем. В нашем примере роль этого тождества играет повторное возведение в степень. Проверяется, что выражение t не имеет вида $h(\ldots y \ldots)$, где y - переменная, не имеющая других вхождений в A. Затем выбирается новая переменная v, и в обобщаемое тождество вместо x подставляется $g(x,v)$. После применения вспомогательного тождества получаем тождество, у которого в заменяемой части вместо $f(g(x,v),t)$ расположено $f(x,h(v,t))$. Легко видеть, что оно обобщает исходное тождество свертки за счет нового коэффициента v.

2. Обобщение сверточного тождества путем ввода бесповторного внешнего параметра.

Чтобы продолжить обобщение тождества, полученного в предыдущем пункте, достаточно подставить вместо переменной d выражение d/e и использовать следующее вспомогательное тождество:

$\forall_{cde}(\neg(e=0)\ \&\ c-\text{число}\ \&\ d-\text{число}\ \&\ e-\text{число} \to c(d/e)=(cd)/e)$

В результате получим:

$\forall_{abcde}(\neg(e=0)\ \&\ a-\text{число}\ \&\ b-\text{число}\ \&\ c-\text{число}\ \&\ d-\text{число}\ \&\ e-\text{число}\ \&\ 0<a\ \&\ 0<b \to a^{(cd)/e}b^c=(a^{d/e}b)^c)$

Этот переход обеспечивается следующим приемом программирующего вывода. Если в заменяемой части A тождества свертки встречается выражение $f(x,t)$, где x - переменная, не имеющая других вхождений в A, то при помощи справочника поиска теорем находятся всевозможные тождества вида $f(g(u,v),w)=h(f(u,w),v)$, где u,v,w - переменные, причем g и h имеют единицу по переменной v. В нашем примере роль такого тождества играет умножение на дробь. Затем выбирается новая переменная v, и в обобщаемое тождество вместо x подставляется $g(x,v)$. После применения вспомогательного тождества и тривиальных упрощений получается тождество, у которого в заменяемой части вместо $f(x,t)$ расположено $h(f(x,t),v)$. Перед выдачей результата проверяется неизбыточность добавленного параметра v.

Несколько применений описанных двух приемов обобщают рассматриваемое тождество до следующего вида:

$\forall_{abcdefg}(\neg(e=0)\ \&\ \neg(g=0)\ \&\ a-\text{число}\ \&\ b-\text{число}\ \&\ c-\text{число}\ \&\ d-\text{число}\ \&\ e-\text{число}\ \&\ f-\text{число}\ \&\ g-\text{число}\ \&\ 0<a\ \&\ 0<b \to a^{cd/e}b^{cf/g}=(a^{d/e}b^{f/g})^c)$

3. Обобщение сверточного тождества путем ввода повторного дополнительного параметра операнда.

Продолжим цепочку обобщений. Для этого подставим в предыдущее тождество вместо переменной c выражение c/h и используем следующее вспомогательное тождество:

$\forall_{ade}(\neg(a=0)\ \&\ \neg(e=0)\ \&\ a-\text{число}\ \&\ d-\text{число}\ \&\ e-\text{число} \to (d/a)/e)=d/(ae)$

Получим следующий результат, который является окончательным и служит источником приема решателя:

$\forall_{abcdefgh}(\neg(e=0)\ \&\ \neg(g=0)\ \&\ \neg(h=0)\ \&\ a-\text{число}\ \&\ b-\text{число}\ \&\ c-\text{число}\ \&\ d-\text{число}\ \&\ e-\text{число}\ \&\ f-\text{число}\ \&\ g-\text{число}\ \&\ h-\text{число}\ \&\ 0<a\ \&\ 0<b \to a^{cd/(eh)}b^{cf/(gh)}=(a^{d/e}b^{f/g})^{c/h})$

Строго говоря, здесь приходится использовать два вспомогательных тождества - кроме тождества деления дроби, нужно также тождество умножения на дробь.

Соответствующий прием программирующего вывода может быть сформулирован, например, следующим образом. В заменяемой части тождества свертки усматривается подвыражение $f(g(x,t),s)$, где x - переменная, имеющая в этой части несколько вхождений, причем каждое - в контексте $f(g(x,p),q)$. Операция g - ассоциативная и коммутативная. При помощи справочников поиска

теорем находятся тождества вида $g(h(x,y),z) = u(g(x,z),y)$ и $f(u(x,y),z) = f(x,v(y,z))$. Здесь операция h имеет единицу по переменной y; операция v - ассоциативна и коммутативна. Данные тождества суть аналоги тождеств $(x/y)\cdot z = (x\cdot z)/y$ и $(x/y)/z = x/(yz)$ соответственно. Затем выбирается новая переменная w, в обобщаемое тождество вместо переменной x подставляется $h(x,w)$, и применяются указанные выше вспомогательные тождества. Проверяется неизбыточность нового параметра w.

Заметим, что промежуточные теоремы цепочки обобщений отбрасываются, и в качестве результата выдаются лишь окончательные версии. Кроме перечисленных выше, имеется множество других обобщающих приемов. Приведем теперь несколько приемов программирующего вывода, обеспечивающих варьирование теоремы.

4. Попытка опрокинуть тождество общей стандартизации путем сильного упрощения заменяемого терма.

   Рассмотрим тождество для сокращения дробей:

   $\forall_{cdf}(\neg(c=0)\ \&\ \neg(f=0)\ \&\ c-\text{число}\ \&\ d-\text{число}\ \&\ f-\text{число} \to (df)/(cf) = d/c)$

   Это тождество имеет характеристику "нормализация(второйтерм)", т.е. является тождеством общей стандартизации.

   Если воспользоваться другим тождеством общей стандартизации

   $\forall_{ab}(a-\text{число}\ \&\ b-\text{число}\ \&\ \neg(b=0) \to (a/b)b = a),$

   представляющим собой определение деления, то при унификации числителя df заменяемой части первого тождества с заменяемой частью $(a/b)b$ второго можно вывести следствие:

   $\forall_{abc}(a/(bc) = (a/b)/c)$

   Это следствие тоже является тождеством общей стандартизации, однако направление его применения - обратное.

   Прием, реализующий данный вывод, усматривает в заменяемой части тождества общей стандартизации вхождение операции $f(\ldots)$, и обращается к справочнику поиска теорем для обнаружения другого тождества общей стандартизации, обеспечивающего сильное упрощение выражений с заголовком f. Создано несколько справочников такого типа. Например, для поиска тождеств, у которых заменяемая часть имеет имеет несколько переменных, а заменяющая - не более одной. Затем подтерм $f(\ldots)$ заменямой части первого тождества унифицируется с заменяемой частью второго, и после унификации к нему применяется второе тождество. Проверяется, что результатом служит тождество общей стандартизации, имеющее встречное направление.

5. Попытка проварьировать заменяемую часть тождества общей стандартизации.

   Рассмотрим тождество для вынесения наружу минуса из сомножителя:

   $\forall_{ab}(a-\text{число}\ \&\ b-\text{число} \to -ab = (-a)b)$

Оно имеет характеристику "нормализация(первыйтерм)". Если воспользоваться вспомогательным тождеством "перегруппировочного" типа:

$\forall_{ab}(a - \text{число } \& \ b - \text{число} \to -(a-b) = b - a)$,

то нетрудно вывести следующее следствие:

$\forall_{abc}(a - \text{число } \& \ b - \text{число } \& \ c - \text{число} \to -a(b-c) = a(c-b))$

Последнее тождество уже не рассматривается как тождество общей стандартизации, а имеет характеристику "свертка(второйтерм)". Оно используется для сокращенной перезаписи выражения при редактировании ответа задачи. Прием, реализующий данный вывод, аналогичен предыдущему. Отличие заключается лишь в использовании других справочников поиска теорем. Они отбирают тождества, у которых обе части имеют одни и те же переменные, причем каждая переменная - бесповторная. Не обязательно, чтобы такое тождество обеспечивало общую стандартизацию.

6. Извлечение из декомпозирующей эквивалентности импликации для проверочного оператора.

   Рассмотрим определение принадлежности объединению множеств:

   $\forall_{abc}(b - \text{set } \& \ c - \text{set} \to a \in b \cup c \leftrightarrow a \in b \ \vee \ a \in c)$

   Эта эквивалентность имеет характеристику "или(второйтерм)", т.е. представляет собой декомпозицию элементарного утверждения в дизъюнкцию, построенную с помощью логических связок из элементарных утверждений. Она позволяет получить следующие два следствия:

   $\forall_{abc}(b - \text{set } \& \ c - \text{set } \& \ a \in b \to a \in b \cup c)$

   $\forall_{abc}(b - \text{set } \& \ c - \text{set } \& \ \neg(a \in b) \ \& \ \neg(a \in c) \to \neg(a \in (b \cup c)))$

   Первое из них может быть использовано в проверочном операторе, усматривающем принадлежность, второе - в операторе, усматривающем непринадлежность.

   Данный пример приводит к приему программирующего вывода, обрабатывающему эквивалентность вида $A \leftrightarrow (B_1 \ \vee \ \ldots \ \vee \ B_n)$, где A - элементарное утверждение. Действия приема распадаются на две части. Первая часть приводит дизъюнкцию $(B_1 \ \vee \ \ldots \ \vee \ B_n)$ к виду дизъюнкции конъюнкций элементарноых утверждений, и для каждой конъюнкции C выводит следствие $C \to A$. Предварительно проверяется, что существует проверочный оператор, обрабатывающий утверждения вида A. Вторая часть - приводит к виду дизъюнкции конъюнкций элементарных утверждений отрицание утверждения $(B_1 \vee \ldots \vee B_n)$, и для каждой конъюнкции C выводит следствие $C \to \neg(A)$. Как и выше, предварительно устанавливается существование проверочного оператора, обрабатывающего утверждения вида $\neg(A)$.

7. Вывод из определения класса условия принадлежности классу.

   Рассмотрим определение образа множества:

$\forall_{af}(a - \text{set}\ \&\ f - \text{функция}\ \&\ a \subseteq \text{Dom}(f) \to \text{образ}(f,a) = \text{set}_x(\exists_y(y \in a\ \&\ x = f(y))))$

Это тождество имеет характеристику "описатель(первыйтерм)", так как оно может быть использовано для исключения описателя "класс". В обычных ситуациях тождество применяется в направлении "справа налево". Однако, для условия принадлежности образу бывают нужны приемы, выполняющие обратное преобразование. Они основаны на следующем тривиальном следствии исходного тождества:

$\forall_{abf}(f - \text{функция}\ \&\ b - \text{set}\ \&\ b \subseteq \text{Dom}(f) \to a \in \text{образ}(f,b) \leftrightarrow \exists_x(a = f(x)\ \&\ x \in b))$.

Прием программирующего вывода здесь просто преобразует тождество $B = \text{set}_x A(x)$ в эквивалентность $x \in B \leftrightarrow A(x)$.

8. Разбиение теоремы с условным выражением на две теоремы для подслучаев.

   Рассмотрим определение модуля:

   $\forall_a(a - \text{число} \to |a| = (a \text{ при } 0 \le a,\ \text{иначе}\ -a))$

   Для двух различных альтернатив условного выражения выводим следующие два следствия:

   $\forall_a(a - \text{число}\ \&\ 0 \le a \to |a| = a)$

   $\forall_a(a - \text{число}\ \&\ a < 0 \to |a| = -a)$

   Из этого примера легко извлекается прием программирующего вывода, применяемый к произвольной теореме с условным выражением в консеквенте.

9. Использование тождества общей стандартизации для получения тождества, устраняющего сложное подвыражение с неизвестными при навешивании внешней операции.

   Для исключения неизвестных радикалов путем возведения обеих частей уравнения в некоторую степень используется тождество:

   $\forall_{abcd}(\neg(c = 0)\ \&\ a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{число}\ \&\ 0 \le a\ \&\ 0 \le b \to (ba^{d/c})^c = b^c a^d)$

   Оно сопровождается в базе теорем характеристикой "смнеизв(...)", указывающей, что было получено именно для этой цели. В ней уточняется контекст применения тождества: преобразуется подтерм условия задачи на описание, выражение a содержит неизвестные, d - натуральная константа, выражение c не содержит неизвестных, причем либо b тоже не содержит неизвестных, либо c - натуральная константа. Рассматриваемое тождество получается обобщением относительно данной характеристики следующего более простого тождества, которое, собственно, и будет представлять для нас интерес в этом пункте:

   $\forall_{acd}(\neg(c = 0)\ \&\ a - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{число}\ \&\ 0 \le a \to (a^{d/c})^c = a^d)$.

   Оно легко получается из тождества последовательного возведения в степень

$\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ 0 \leq a \rightarrow (a^b)^c = a^{bc})$

с помощью вспомогательного тождества

$\forall_{ab}(a - \text{число} \ \& \ b - \text{число} \ \& \ \neg(b = 0) \rightarrow (a/b)b = a).$

Прием программирующего вывода, выполняющий этот шаг, анализирует подтерм заменяющего терма тождества общей стандартизации (в нашем случае - выражение bc) и обращается к справочнику поиска теорем для нахождения сокращающего тождества, применимого к данному подтерму. Под сокращающим понимается тождество вида $f(x, g(x, y)) = a$, где a - константа либо переменная. Далее находится результат последовательного применения двух тождеств, и проверяется, что он может быть использован для исключения сложного выражения с неизвестными при навешивании на обе части уравнения некоторой внешней операции. Таким образом и появляется изначально характеристика "смнеизв(...)". Заметим, что никаких других характеристик выведенному тождеству не присваивается, и далее оно будет использоваться только по своему прямому назначению.

База теорем предоставляет обучающий материал не только для программирующего вывода, извлекающего источники теорем из базисных теорем раздела, но и для "исследовательского вывода", объясняющего получение одних базисных теорем из других. Граница между программирующим и исследовательским выводами весьма условная. Рассмотрим несколько приемов, которые могут быть отнесены к любой из этих категорий.

10. Вывод упрощающего тождества в стандартной форме путем унификации заменяемых частей двух ранее найденных тождеств.

    Во многих разделах математики упрощение выражений связано с использованием многочленоподобных "стандартных форм". Например, таковы дизъюнктивные и конъюнктивные нормальные формы в алгебре логики и их аналоги в алгебре множеств. Преимущества, которые создает переход к стандартной форме, заключаются в уменьшении расстояний между ее подвыражениями: из любой точки к любой другой можно перейти всего за три шага - два раза через аналог "умножения" и один раз через аналог "сложения". В результате существенно повышается эффективность использования упрощающих тождеств и уменьшается их количество. Чтобы порождать новые такие тождества, задается список базисных тождеств, по которым сразу генерируются приемы вспомогательного пакетного нормализатора N. Этот нормализатор используется только в цикле вывода новых тождеств. Он будет упрощать обе части каждого выводимого тождества, чтобы исключить избыточные следствия. Основной шаг вывода - унификация заменяемых частей двух упрощающих тождеств, позволяющая применить их последовательно - сначала первое тождество применяется в направлении усложнения, затем второе - в направлении упрощения. Данное правило вывода хорошо известно в математической логике. С его помощью можно быстро создавать аппарат упрощения выражений в различных алгебраических системах. Преобразование тождеств в приемы, ввиду тривиальной целевой установки, здесь не вызывает затруднений.

    В качестве примера рассмотрим тождества $ac \ \vee \ bc\overline{a} = ac \ \vee \ bc$ и $p \ \vee \ pq = q$ из алгебры логики, используемые для упрощения дизъюнктивных нормальных

форм. Если проводить унификацию заменяемых частей, вводя вспомогательные переменные для дополнительных дизъюнктивных членов, то в качестве одного из вариантов получим выражение $ac \vee bc\overline{a} \vee b\overline{a}$, При этом заменяющие части обоих тождеств равны, соответственно, $ac \vee bc \vee b\overline{a}$ и $ac \vee b\overline{a}$. Отсюда выводится тождество обобщенного склеивания: $ac \vee bc \vee b\overline{a} = ac \vee b\overline{a}$.

11. Попытка склеить два члена факторизуемого выражения.

    Иногда возникает задача преобразовать выражение к такому виду, у которого заголовком служит заданный логический символ. Например, задача разложения на множители. Для вывода тождеств, обеспечивающих приведение к заданным заголовкам, используется процедура, аналогичная описанной в предыдущем пункте. Сначала порождаются простейшие тождества данного типа. В случае разложения на множители для этого берется произведение двух или более сумм различных переменных, и предпринимается раскрывание скобок. На период вывода по найденным тождествам создается нормализатор приведения к заданным заголовкам, используемый для отбрасывания тождеств, сводящихся к ранее полученным. Основной шаг вывода - применение к нескольким членам заменяемой части текущего тождества вспомогательного тождества, обеспечивающего взаимное уничтожение этих членов. В результате промежуточные члены, необходимые для непосредственного разложения, оказываются "спрятаны", и возникает ценное новое тождество.

    В качестве примера рассмотрим тождество $(a+b)(c+d) = ac+bc+ad+bd$. При использовании вспомогательного тождества $p - p = 0$ можно унифицировать ad с p, bc - с $-p$ и получить следствие $(a + b)(a - b) = a^2 - b^2$. Таким же образом порождаются остальные тождества сокращенного умножения. Кроме них, удается получить множество других полезных для ускоренного разложения на множители формул. Обнаруживаются и такие экзотические соотношения, как $a^5 - b^5 - ba^4 = (a^2 + b^2 - ab)(a^3 - b^3 - ab^2)$.

12. Попытка проварьировать эквивалентность, разрешающую относительно неизвестной, с помощью тождества, создающего кратные вхождения переменной.

    Прием рассматривает эквивалентность, разрешающую относительно неизвестной x некоторое утверждение A с единственным вхождением этой неизвестной. Внутри A выбирается содержащее x подвыражение t, имеющее самый сложный в эвристическом смысле заголовок. Справочник поиска теорем перечисляет тождества, позволяющие преобразовать t в выражение t' с несколькими вхождениями переменной x. В исходной эквивалентности предпринимается замена t на t', и после несложной стандартизации выдается результат. Цель данного вывода - получить эквивалентность, разрешающую, хотя бы в частном случае, утверждение с несколькими вхождениями неизвесетной. Обычно она подлежит цепочке обобщений.

    В качестве примера возьмем эквивалентность для разрешения простейшего степенного уравнения, имеющего рациональный показатель степени с четным числителем:

    $\forall_{abc}(a - \text{число} \;\&\; b - \text{число} \;\&\; c - \text{число} \;\&\; b - \text{rational} \;\&\; \text{числитель}(b) - \text{even} \;\&\; \neg(b = 0) \to a^b = c \leftrightarrow 0 \leq c \;\&\; (a = c^{1/b} \vee a = -c^{1/b}))$.

Вспомогательным тождеством будет тождество для квадрата суммы

$\forall_{ab}(a - \text{число } \& \ b - \text{число} \to (a+b)^2 = a^2 + 2ab + b^2)$,

позволяющее перейти от выражения $(a+b)^2$ с бесповторными переменными к повторным переменным. После группировки в правой части всех известных слагаемых, получим:

$\forall_{cde}(c - \text{число } \& \ d - \text{число } \& \ e - \text{число} \to d^2 + 2de = c - e^2 \leftrightarrow ((d = -e + \sqrt{c} \ \vee \ d = -(e + \sqrt{c})) \ \& \ 0 \leq c))$.

Роль неизвестной здесь играет переменная d. Чтобы получить из этой эквивалентности известную формулу для решения квадратного уравнения, далее можно использовать прием программирующего вывода, усматривающий подвыражение уравнения, не содержащее неизвестных, и обозначающий его новым параметром. На первом шаге, вводя вспомогательный параметр a для выражения в правой части уравнения и разрешая относительно c уравнение $c - e^2 = a$, получим:

$\forall_{ade}(a - \text{число } \& \ d - \text{число } \& \ e - \text{число} \to d^2 + 2de = a \leftrightarrow ((d = -e + \sqrt{a + e^2} \ \vee \ d = -(e + \sqrt{a + e^2})) \ \& \ 0 \leq a + e^2))$

Еще раз применяем этот же прием, обозначая посредством b подвыражение $2e$ и разрешая относительно e уравнение $2e = b$. Получаем:

$\forall_{abd}(a - \text{число } \& \ b - \text{число } \& \ d - \text{число} \to d^2 + bd = a \leftrightarrow ((d = -(b + \sqrt{b^2 + 4a})/2 \ \vee \ d = (-b + \sqrt{b^2 + 4a}/2)) \ \& \ 0 \leq b^2 + 4a))$

Опускаем описание приемов программирующего вывода, доводящих обобщение данной формулы до случая произвольного коэффициента перед квадратом неизвестной. Заметим, что на всех этапах обобщения теорема сопровождалась характеристикой, указывающей неизвестную d.

В заключение приведем прием вывода теорем, позволяющий системе "открывать" формулу Кардано.

13. Попытка использовать оператор усмотрения повторяющихся подвыражений для получения импликации, подбирающей корень уравнения.

Прием выводит теорему, позволяющую подобрать значение неизвестной, имеющей кратные вхождения в реализуемое условие. Он анализирует исходную теорему вида $\forall_x(A(x) \to B(x))$ и пытается таким образом преобразовать утверждение $B(x)$, чтобы какое-либо неконстантное подвыражение t встречалось в нем более одного раза. Далее выбирается новая переменная y, находится результат $C(x, y)$ замены всех вхождений t в преобразованное $B(x)$ на переменную y, и выводится следствие $\forall_x(A(x) \ \& \ y = t \to C(x, y))$. Роль неизвестной здесь играет переменная y, причем t - подбираемое значение. Это отражено в характеристике, сопровождающей теорему. Далее реализуется цепочка обобщений.

В качестве примера рассмотрим тождество для куба суммы:

$\forall_{ab}(a - \text{число } \& \ b - \text{число} \to (a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3)$.

Консеквент $(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$ нетрудно привести к виду

$(a+b)^3 = a^3 + 3ab(a+b) + b^3$,

используя для этого вспомогательный пакетный нормализатор, выделяющий "скрытые" повторные вхождения одного и того же выражения. Подвыражение $a+b$ здесь встречается дважды, и для вспомогательной переменной y получаем следствие:

$\forall_{aby}(a - \text{число } \&\ b - \text{число } \&\ y = a + b \to y^3 - 3aby = a^3 + b^3)$.

Здесь все неизвестные члены сгруппированы слева, а известные - справа. Далее реализуется цепочка обобщений, основанных на том же принципе, что и для квадратного уравнения: выбирается подвыражение без неизвестных, которое обозначается вспомогательным параметром. Прежде всего, для $-3ab$ вводится параметр c. Подставляемое вместо a выражение определяется из уравнения $-3ab = c$, и получаем первый шаг цепочки обобщений:

$\forall_{bcy}(b - \text{число } \&\ \neg(b=0)\ \&\ c - \text{число } \&\ y = -c/(3b) + b \to y^3 + cy = -c^3/(27b^3) + b^3)$.

На следующем шаге - вводим для выражения $-c^3/(27b^3) + b^3$ вспомогательный параметр a. Уравнение $-c^3/(27b^3) + b^3 = a$ легко решается, так как преобразуется к квадратному относительно b^3. В результате получаем формулу корня кубического уравнения, не имеющего члена с квадратом неизвестной:

$\forall_{abcy}(c - \text{число } \&\ \neg(c=0)\ \&\ a - \text{число } \&\ 0 \le 4c^3 + 27a^2\ \&\ y = -c/(3b) + b\ \&$
$b = \sqrt[3]{3\sqrt{3}a - \sqrt{4c^3 + 27a^2}}/(\sqrt[3]{2}\sqrt{3}) \to y^3 + cy = a)$

Для остальных шагов цепочки обобщений, связанных с добавлением коэффициента при y^3 и члена с y^2, создание приемов вывода не составляет труда.

Если вывод некоторой теоремы ячейки вывода из стартовых теорем ячейки разбит на отдельные шаги, то идея каждого такого шага обычно сравнительно простая. Для ее понимания можно ограничиться лишь примером результирующей теоремы и одной либо нескольких теорем, из которых она извлекается. Более сложной оказывается техническая проработка этой идеи. В 9 томе монографии "Компьютерное моделирование логических процессов" приводятся подробные описания алгоритмов, реализующих вывод теорем. Они представляют собой попросту переизложение соответствующих программ ЛОСа, с сохранением нумерации программных переменных. Каждый такой алгоритм сопровождается примером, обработка которого прослеживается в процессе выполнения алгоритма. Разумеется, чтение программы ЛОСа, хотя бы и переведенной в текстовые описания, не позволяет сразу понять принцип вывода. Однако, из примера и текста алгоритма, его все-таки можно восстановить. Ниже будут приведены упражнения как на самостоятельное создание приемов вывода, выводящих заданную теорему из других заданных теорем, так и на анализ процедур вывода, содержащихся в 9 томе. Пока что ограничимся несколькими примерами описания приемов вывода, извлеченными из этого тома.

14. Варьирование декомпозирующей эквивалентности с помощью тождества свертки.

В качестве примера рассмотрим вывод теоремы

$\forall_{adef}(a-\text{set}\ \&\ d-\text{set}\ \&\ e-\text{set}\ \&\ f-\text{set} \to d \subseteq f \cup (a \cap e) \leftrightarrow d \subseteq a \cup f\ \&\ d \subseteq e \cup f)$

из теоремы

$\forall_{bcd}(b-\text{set}\ \&\ c-\text{set}\ \&\ d-\text{set} \to d \subseteq b \cap c \leftrightarrow d \subseteq b\ \&\ d \subseteq c)$

и дополнительной теоремы

$\forall_{cef}(c-\text{set}\ \&\ e-\text{set}\ \&\ f-\text{set} \to (c \cup f) \cap (e \cup f) = f \cup (c \cap e))$

Текущая характеристика - "и(второйтерм)".

Переменной x10 присваивается заменяемый терм, переменной x11 - заменяющий, переменной x12 - набор антецедентов. Внутри заменяемой части выбирается вхождение x14 двуместной операции x15. В нашем примере - $b \cap c$. Проверяется, что она не является заголовком утверждения. Справочник поиска теорем "свертки" определяет по символу x15 указанную выше дополнительную теоремы. В качестве заменяемой ее части рассматривается та часть консеквента, которая имеет заголовком символ x15. В нашем примере - $(c \cup f) \cap (e \cup f)$. Проверяется, что дополнительная теорема не имеет характеристики "нормализация", указывающей противоположное направление замены. Затем процедура "тождвывод" определяет результат x21 преобразования вхождения x14 при помощи дополнительной теоремы. Он имеет вид:

$\forall_{adef}(a \cup f - \text{set}\ \&\ e \cup f - \text{set}\ \&\ d - \text{set}\ \&\ a - \text{set}\ \&\ e - \text{set}\ \&\ f - \text{set} \to$
$d \subseteq f \cup (a \cap e) \leftrightarrow d \subseteq a \cup f\ \&\ d \subseteq e \cup f)$

К теореме x21 последовательно применяются операторы "Спускоперандов", "демодификация" и "нормтеорема".

15. Попытка реализации антецедента сворачиваемой импликации с помощью дополнительной простой импликации.

В качестве примера рассмотрим вывод теоремы

$\forall_{abd}(a - \text{set}\ \&\ d - \text{set}\ \&\ a \subseteq \mathbb{R}\ \&\ d \subseteq a\ \&\ b - \text{число}\ \&\ \text{нижняягрань}(b, a) \to$
$\text{нижняягрань}(b, d))$

из теоремы

$\forall_{ab}(a - \text{set}\ \&\ a \subseteq \mathbb{R}\ \&\ b - \text{число} \to \text{нижняягрань}(b, a) \leftrightarrow \forall_c(c \in a \to b \leq c))$

и дополнительной теоремы:

$\forall_{abc}(a - \text{set}\ \&\ b - \text{set}\ \&\ a \subseteq b\ \&\ c \in a \to c \in b)$

Текущая характеристика - "кванторнаясвертка(x1)". Этой характеристикой снабжаются кванторные эквивалентности, выполняющие исключение квантора. x1 - направление замены.

Переменной x10 присваивается заменяемый терм, представляющий собой кванторную импликацию. Переменной x11 присваивается ее кванторная приставка.

Переменной x12 присваивается набор антецедентов импликации x10. В нем выбирается элементарное утверждение x13, заголовок x14 которого отличен от символа "не". В нашем примере x13 - "$c \in a$". По символу x14 справочник поиска теорем "транзитоперанд" определяет указанную выше дополнительную теорему. Переменной x17 присваивается результат переобозачения ее переменных на переменные, не входящие в исходную теорему. В нашем примере имеем:

$\forall_{def}(d - \text{set} \ \& \ e - \text{set} \ \& \ d \subseteq e \ \& \ f \in d \to f \in e)$

Переменной x18 присваивается консеквент теоремы x17., переменной x19 - объединение его параметров с переменными утверждения x13, отличными от переменных списка x11. В нашем примере - "e, f, a". Определяется подстановка S вместо переменных x19, унифицирующая для термов x13 и x18. Переменной x21 присваивается набор результатов применения S к антецедентам теоремы x17. Подсписок списка x21, образованный всеми утверждениями, содержащими переменные связывающей приставки x11, присваивается переменной x22. Переменной x23 присваивается набор результатов применения S к утверждениям списка x12, отличным от x13. Переменной x24 присваивается результат применения S к консеквенту импликации x10. В нашем примере x21 - "$d - \text{set}$", "$a - \text{set}$", "$d \subseteq a$", "$c \in d$", x22 - "$c \in d$", x23 - пустой список, x24 - "$b \leq c$". Формируется импликация x25 вида "длялюбого(x11 если x23 x22 то x24)". В нашем примере она имеет вид "$\forall_c(c \in d \to b \leq c)$". Переменной x26 присваивается набор результатов применения подстановки S к антецедентам исходной теоремы. Решается задача на преобразование для упрощения импликации x25 относительно посылок x25. На время ее решения блокировки снимаются. Проверяется, что результат x28 - элементарное утверждение. В нашем примере - "нижняяграань(b, d)". Определяется результат x29 применения подстановки S к заменяющей части исходной теоремы. Затем создается импликация, антецеденты которой получаются объединением списка x26 с разностью списков x21 и x22 и добавлением утверждения x29. Консеквентом является утверждение x28. Эта импликация обрабатывается оператором "нормтеорема".

16. Попытка развязки переменной атомарного выражения.

В качестве примера рассмотрим вывод теоремы

$\forall_{abefglmAK}(\text{коорд}(\text{вектор}(ab), K) = (l, m) \ \& \ \text{коорд}(\text{вектор}(Af), K) = (e, g) \ \&$
$a - \text{точка} \ \& \ b - \text{точка} \ \& \ f - \text{точка} \ \& \ A - \text{точка} \ \& \ \text{систкоорд}(K) \to$
$\text{коорд}(\text{вектор}(ab) + \text{вектор}(Af), K) = (e + l, g + m))$

из теоремы

$\forall_{abefglmABK}(\text{коорд}(\text{вектор}(fB), K) = (l, m) \ \& \ \text{коорд}(\text{вектор}(Af), K) = (e, g) \ \&$
$f - \text{точка} \ \& \ A - \text{точка} \ \& \ B - \text{точка} \ \& \ \text{систкоорд}(K) \to \text{коорд}(\text{вектор}(fB) +$
$\text{вектор}(Af), K) = (e + l, g + m))$

и дополнительной теоремы

$\forall_{ABC}(A - \text{точка} \ \& \ B - \text{точка} \ \& \ C - \text{точка} \to \exists_D(D - \text{точка} \ \& \ \text{вектор}(AB) =$
$\text{вектор}(CD)))$

От сложения двух векторов с совпадающими концами выполняется переход к сложению произвольных двух векторов. Текущая характеристика - "коорд". Этой характеристикой снабжаются кванторные тождества для координат объектов.

Переменной x8 присваивается набор антецедентов. В консеквенте находится атомарное выражение, тип значения которого отличен от "число", "множество", "слово". Этот тип присваивается переменной x10. В нашем примере - "Вектор". Переменной x11 присваивается пара частей равенства в консеквенте, переменной x12 - список атомарных подвыражений левой части, имеющих тип x10. В нашем примере - "Вектор(fB)", "Вектор(Af)". Проверяется, что список x12 непуст. Проверяется, что правая часть не содержит атомарных подвыражений типа x10. В списке x12 выбирается выражение x13; заголовок его присваивается переменной x14. В нашем примере - выражение "вектор(fB)". Проверяется, что число корневых операндов выражения x13 равно 2 и что каждый операнд - переменная. Переменной x15 присваивается список этих переменных. Проверяется, что его длина равна 2, т.е. переменные различные. В списке x15 выбирается переменная x16, имеющая в левой части более одного вхождения. В нашем случае - f. Переменной x17 присваивается оставшаяся переменная списка x15. В нашем случае - B. Проверяется, что она имеет единственное вхождение в левую часть. Справочник поиска теорем "незавгруппы" определяет по символу x14 указанную выше дополнительную теорему. Переменной x20 присваивается результат переобозначения ее переменных на не входящие в исходную теорему. В нашем примере получаем:

$$\forall_{abc}(a-\text{точка}\ \&\ b-\text{точка}\ \&\ c-\text{точка} \rightarrow \exists_d(d-\text{точка}\ \&\ \text{вектор}(ab) = \text{вектор}(cd)))$$

Переменной x21 присваивается набор антецедентов теоремы x20, переменной x22 - вхождение консеквента. Проверяется, что этот консеквент представляет собой квантор существования. Переменной x23 присваивается его связывающая приставка. Проверяется, что она состоит из единственной переменной x24. Переменной x25 присваивается набор конъюнктивных членов подкванторного утверждения. В нем выбирается равенство x26. В нашем примере - "вектор(ab) = вектор(cd)". Переменной x27 присваивается пара частей этого равенства. В ней выбирается утверждение x28, содержащее переменную x24. Для нашего примера - "вектор(cd)". Переменной x29 присваивается список параметров терма x28. Проверяется, что он имеет длину 2. Переменной x30 присваивается элемент этого списка, отличный от x24. В нашем случае - "c". Определяется подстановка S вместо переменных x29, унифицирующая термы x13 и x28. Переменной x32 присваивается терм, подставляемый ею вместо x30. В нашем примере - "f". Проверяется, что этот терм совпадает с переменной x16. Переменной 34 присваивается конкатенация списка антецедентов исходной теоремы и списка результатов применения подстановки S к антецедентам теоремы x20. Переменной x36 присваивается элемент списка x27,отличный от x28. В нашем случае - "вектор(ab)". Переменной x39 присваивается результат замены в консеквенте исходной теоремы всех вхождений выражения x13 на терм x36. В нашем примере - "коорд(вектор(ab) + вектор(Af), K) = $(e + l, g + m)$". Такая же замена предпринимается во всех утверждениях списка x34. Затем создается импликация с антецедентами x34 и консеквентом x39. Она обрабатывается нормализатором "нормтеорема".

17. Примерка кванторного определения свойства функции на константную и тождественную функции.

    В качестве примера рассмотрим вывод теоремы

    $\forall_{agi}(g - \text{число}\ \&\ \text{Число}(a)\ \&\ \text{типпредела}(i) \to \lim_{e \to a \setminus i} g = g)$

    из теоремы

    $\forall_{abfi}(f - \text{функция}\ \&\ \text{Число}(a)\ \&\ b - \text{число}\ \&\ \text{типпредела}(i)\ \&\ \text{Локопред}(f,a,i)\ \&$
    $\text{Dom}(f) \subseteq \mathbb{R}\ \&\ \text{Val}(f) \subseteq \mathbb{R} \to \text{предел}(f,i,a) = b \leftrightarrow \forall_c(c - \text{число}\ \&\ 0 < c \to$
    $\exists_d(0 < d\ \&\ d - \text{число}\ \&\ \forall_x(x - \text{число}\ \&\ x \in \text{Окрестность}(a,d,i) \to |f(x) - b| < c))))$

    Характеристика - "определение(предел$(f,i,a) = b$)".

    Напомним смысл используемых в теоремах обозначений. Условие "Число(a)" означает, что a - вещественное число либо один из символов "минусбеск", "плюсбеск". Условие "типпредела(i)" означает, что i - указатель типа предела, т.е. либо 0 (двусторонний предел), либо 1 (предел слева), либо 2 (предел справа). Условие "Локопред(f,a,i)" означает, что функция f определена в некоторой непустой проколотой окрестности точки a, причем i - указатель типа окрестности (см. "типпредела"). Выражение "Окрестн(a,d,i)" обозначает проколотую окрестность точки a, размеры которой определяются положительным числом d, причем i - указатель типа окрестности. Наконец, "предел(f,i,a)" обозначает предел функции f в точке a, причем i - указатель типа рассматриваемой окрестности.

    Перейдем к описанию приема. Переменной x8 присваивается определяемый терм. Проверяется, что теорема - эквивалентность. Переменной x12 присваивается определяющий терм. Проверяется, что он содержит квантор. Переменной x13 присваивается список антецедентов. В этом списке находится утверждение x14 с заголовком "функция". Переменной x15 присваивается операнд этого утверждения. Проверяется, что он представляет собой переменную, входящую в список параметров терма x8. В нашем примере x15 - переменная f. Проверяется, что все параметры утверждений x13 являются параметрами терма x8. В списке x13 находится утверждение x16, имеющее вид включения множества значений функции x15 в некоторое множество M. Выбирается переменная x17, не используемая в теореме. В нашем примере - переменная e. Переменной x18 присваивается результат добавления к списку x13 утверждения "принадлежит(x17 M)". В нашем примере - утверждения "$e \in \mathbb{R}$". Решается задача на исследование с посылками x18 и единственной целью (неизвестные x17). После этого в списке посылок задачи находится утверждение x21 вида "P(x17)", у которого P - обозначение типа объекта. В нашем примере - "число". Выбираются различные переменные x23 и x24, не входящие в исходную теорему и отличные от переменной x17. Поочередно рассматриваются три случая. В первом из них переменной x25 присваивается выражение "отображение(x17 принадлежит(x17 x24)x23)", во втором - "отображение(x17 x21 x23)", в третьем - "отображение(x17 x21 x17)". Первые два случая соответствуют константной функции с различием в задании области определения, третий - тождественной функции. В нашем примере берется второй случай, т.е. x25 имеет вид "$\lambda_e(g, e - \text{число})$".

Переменной x26 присваивается результат подстановки в терм x8 выражения x25 вместо переменной x15. В нашем примере - "$\lim_{e\to a\setminus i} g$". Переменной x27 присваивается результат такой же подстановки в терм x12. В нашем примере он имеет вид:

$\forall_c(c - \text{число} \ \& \ 0 < c \to \ \exists_d(0 < d \ \& \ d - \text{число} \ \& \ \forall_x(x - \text{число} \ \&$
$x \in \text{Окрестность}(a, d, i) \to |\lambda_e(g, e - \text{число})(x) - b| < c)))$

Переменной x28 присваивается список результатов подстановки x25 вместо x15 в утверждения списка x13. Переменной x29 присваивается список параметров терма x27. В нашем примере - a, b, g, i. Переменной x30 присваивается список утверждений набора x28, содержащих переменные списка x29. В нашем примере x30 состоит из утверждений:

"Число(a)", "$b - $ число", "типпредела(i)", "Локопред$(\lambda_e(g, e - \text{число}), a, i)$",
"Dom$(\lambda_e(g, e - \text{число})) \subseteq \mathbb{R}$", "Val$(\lambda_e(g, e - \text{число})) \subseteq \mathbb{R}$".

Переменной x31 присваивается остаток списка x28. Переменной x32 присваивается объединение списка x30 и набора конъюнктивных членов утверждения x27. Проверяется, что список x29 непуст. Решается задача на описание с посылками x31 и условиями x32. Цели ее - "полный", "прямойответ", "попыткаспуска", "неизвестные x29". Ответ присваивается переменной x34. В нашем примере он имеет вид "(a) & $b - $ число & типпредела(i) & $g = b$". Проверяется, что ответ отличен от символа "отказ" и не имеет связанных переменных. Создается импликация, антецедентами которой служат конъюнктивные члены утверждения x34, а консеквентом - терм x26. Она обрабатывается оператором "Нормтеорема", которому разрешается использовать все уже имеющиеся в списке вывода теоремы. Выбирается конъюнктивный член x37 результата такой обработки, который регистрируется в списке вывода.

18. Примерка кванторного определения свойства функции на элементарные операции раздела.

    В качестве примера рассмотрим вывод теоремы

    $\forall_{aghij}(\text{Dom}(h) = g \ \& \ \text{Dom}(j) = g \ \& \ g \subseteq \mathbb{R} \ \& \ \text{Val}(h) \subseteq \mathbb{R} \ \& \ \text{Val}(j) \subseteq \mathbb{R} \ \&$
    $h - \text{функция} \ \& \ j - \text{функция} \ \& \ \text{предел}(h, i, a) - \text{число} \ \& \ \text{предел}(j, i, a) - \text{число} \ \&$
    $\text{Число}(a) \ \& \ \text{типпредела}(i) \ \& \ \text{Локопред}(h, a, i) \ \& \ \text{Локопред}(j, a, i) \to$
    $\lim_{e\to a\setminus i}(h(e) + j(e)) = \text{предел}(h, i, a) + \text{предел}(j, i, a))$

    из теоремы

    $\forall_{abfi}(f - \text{функция} \ \& \ \text{Число}(a) \ \& \ b - \text{число} \ \& \ \text{типпредела}(i) \ \& \ \text{Локопред}(f, a, i) \ \&$
    $\text{Dom}(f) \subseteq \mathbb{R} \ \& \ \text{Val}(f) \subseteq \mathbb{R} \to \text{предел}(f, i, a) = b \leftrightarrow \forall_c(c - \text{число} \ \& \ 0 < c \to$
    $\exists_d(0 < d \ \& \ d - \text{число} \ \& \ \forall_x(x - \text{число} \ \& \ x \in \text{Окрестность}(a, d, i) \to |f(x) - b| < c))))$

    Характеристика - "определение(предел$(f, i, a) = b)$".

    Начало программы приема совпадает с началом программы предыдущего приема. Для удобства чтения повторим его. Переменной x8 присваивается определяемый терм. Проверяется, что теорема - эквивалентность. Переменной x12 присваивается определяющий терм. Проверяется, что он содержит квантор. Переменной x13 присваивается список антецедентов. В этом списке находится

утверждение x14 с заголовком "функция". Переменной x15 присваивается операнд этого утверждения. Проверяется, что он представляет собой переменную, входящую в список параметров терма x8. В нашем примере x15 - переменная f. Проверяется, что все параметры утверждений x13 являются параметрами терма x8. В списке x13 находится утверждение x16, имеющее вид включения множества значений функции x15 в некоторое множество M. Выбирается переменная x17, не используемая в теореме. В нашем примере - переменная e. Переменной x18 присваивается результат добавления к списку x13 утверждения "принадлежит(x17 M)". В нашем примере - утверждения "$e \in \mathbb{R}$". Решается задача на исследование с посылками x18 и единственной целью (неизвестные x17). После этого в списке посылок задачи находится утверждение x21 вида "P(x17)", у которого P - обозначение типа объекта. В нашем примере - "число".

Дальше начинаются различия. Перечисляются по возрастанию те разделы x23, к которым относится понятие P. В нашем примере x23 - "элементарнаяалгебра". Переменной x24 присваивается набор логических символов, относящихся к разделу x23 и его подразделам. В списке x24 выбирается символ x25, не являющийся предикатным символом, причем такой, что список типов значений выражений с заголовком x25 содержит символ P. В нашем примере x25 - "плюс". Проверяется, что арность x27 символа x25 отлична от 0 и от символа "натуральное" (последнее указывает на возможность произвольного натурального числа операндов и встречается крайне редко, например, у символа "набор"). В нашем примере x27 равно 2. Переменной x28 присваивается список переменных, не встречающихся в исходной теореме, длина которого равно x27+2. В нашем примере x28 состоит из переменных e, g, h, j. Переменным x29 и x30 присваиваются две первых переменных списка x28, переменной x31 - остальные переменные этого списка. В нашем примере x29 - e, x30 - g, x31 - h, j. Переменной x32 присваивается терм, получающийся соединением операцией x25 выражений "значение(X x29)" по всем переменным X списка x31. В нашем примере x32 имеет вид "$h(e) + j(e)$". Переменной x33 присваивается терм "отображение(x29 принадлежит(x29 x30)x32)". В нашем примере - "$\lambda_e(h(e) + j(e), e \in g)$". Переменной x34 присваивается список результатов подстановки терма x33 вместо переменной x15 в утверждения x13, переменной x35 - результат добавления к x34 утверждений "функция(X)", "равно(область(X)x30)" для всех переменных X из x31. Определяется о.д.з. терма, полученного навешиванием операции x25 на список переменных x31. Для каждого утверждения P из этой о.д.з. определяется результат P' подстановки в него вместо переменных X списка x31 термов "значение(X,x29)". Затем к списку x35 добавляется кванторная импликация "длялюбого(x29 если принадлежит(x29 x30) то P')". В нашем примере x35 состоит из следующих утверждений:

"$\lambda_e(h(e) + j(e), e \in g) -$ функция", "Число(a)", "$b -$ число", "типпредела(i)", "Локопред($\lambda_e(h(e) + j(e), e \in g), a, i$)", "Dom($\lambda_e(h(e) + j(e), e \in g)) \subseteq \mathbb{R}$", "Val($\lambda_e(h(e) + j(e), e \in g)) \subseteq \mathbb{R}$", "$h -$ функция", "$j -$ функция", "Dom(h) $= g$", "Dom(j) $= g$", "$\forall_e(e \in g \to j(e) -$ число)", "$\forall_e(e \in g \to h(e) -$ число)".

Переменной x36 присваивается результат подстановки терма x33 вместо переменной x15 в терм x12. В нашем примере он имеет вид:

$\forall_c(c - $ число $\& \ 0 < c \to \exists_d(0 < d \ \& \ d - $ число $\& \ \forall_x(x - $ число $\&$
$x \in$ Окрестность$(a, d, i) \to |\lambda_e(h(e) + j(e), e \in g)(x) - b| < c)))$

Переменной x37 присваивается результат обработки списка x35 процедурой "нормантецеденты" относительно параметров терма x36. В нашем примере он имеет следующий вид:

"Локопред(j, a, i)". "Число(a)", "b- число", "типпредела(i)", "Локопред(h, a, i)", "$g \subseteq \mathbb{R}$", "$h -$ функция", "$j -$ функция", "Dom$(h) = g$", "Dom$(j) = g$", "Val$(j) \subseteq \mathbb{R}$", "Val$(h) \subseteq \mathbb{R}$".

Решается задача на описание x38, посылками которой служат утверждения x37, а единственным условием - утверждение x36. Задача имеет цели "прямойответ", "неизвестные x31", "разделение x31". Ответ задачи присваивается переменной x39. В нашем примере он имеет вид "предел$(j, i, a) -$ число $\&$ предел$(h, i, a) -$ число $\&$ предел$(h, i, a) +$ предел$(j, i, a) = b$".

Собственно, основная работа выполняется задачей x38. Ее цель "разделение x31" указывает, что требуется преобразовать условия к такому виду, в котором никакие две переменные списка x31 (в нашем примере - h, j) не встречаются в одном и том же условии. Вкратце, последовательность действий при решении задачи такова. Сначала предпринимается исключение кванторной импликации в условии путем перехода к задаче с целью "независит". Затем отбрасывается квантор существования и его связывающая приставка добавляется к неизвестным. После этого снова удаляется квантор общности - путем перехода к задаче с целью "независит". В итоге список условий оказывается состоящим из утверждений "$| - h(c) - j(c) + b| < e$", "$d -$ число", "$0 < d$". При этом неизвестными служат переменные h, j, d, несущественной неизвестной - d, имеется цель "независит c, e", но переменной d разрешается зависеть от e. Сохраняется цель "разделение h, j". Далее срабатывает специальный прием, созданный для цели "разделение". Его теорема имеет следующий вид:

$\forall_{abcd}(\exists_{xyzv}(|b + x| < y \ \& \ |c + z| < v \ \& \ x + y = a \ \& \ y + v \le d \ \& \ x -$ число $\&$
$y -$ число $\& \ z -$ число $\& \ v -$ число$) \to |a + b + c| < d)$

Прием имеет заголовок "подборзначений". Указатель "контекст" определяет дополнительную идентификацию цели "разделение X". Переменная a идентифицируется со всеми слагаемыми, не содержащими переменных X, переменная b - со всем слагаемыми, содержащими некоторую выбираемую приемом переменную w списка X, переменная c - с остальными слагаемыми. Проверяется, что как b, так и c содержат переменные списка X. Прием сводит реализацию консеквента к реализации квантора существования - антецедента теоремы. По существу, используется неравенство для модуля суммы, разбиваемой с учетом цели "разделение". Для вывода теорем о пределе суммы и дроби созданы аналогичные приемы. Разумеется, для успешного применения приема вывода теорем эти заготовки, ориентированные на разделение переменных, должны быть созданы заранее. Впрочем, логика их возникновения из простейших свойств указанных операций вполне ясна и впоследствии может быть автоматизирована.

После применения указанного приема происходит исключение квантора существования. В результате появляются условия "$k + m \le e$", "$m -$ число",

"$l-$число", "$k-$число", "$f-$число", "$0 < -|-j(c)+l|+m$", "$0 < -|-h(c)+f|+k$", "$f + l = b$", "$d -$ число", "$0 < d$".

Для исключения неравенства "$0 < -| - j(c) + l| + m$"используется еще один прием, созданный для цели "разделение":

$\forall_{abcdfgi}(f -$ функция & Число(d) & $b -$ число & $c \in$ Окрестн(d, m, i) &
Локопред(f, d, i) & предел$(f, i, d) = a$ & $0 < b$ & $c \in$ Окрестн$(d, g(b), i)$ &
$\forall_x(x -$ число & $0 < x \to g(x) -$ число & $0 < g(x)) \to 0 < b - |a - f(c)|)$

Прием имеет заголовок "подборзначений". Он вводит новую переменную g, причем шестой, седьмой и восьмой антецеденты во вспомогательной задаче замещают консеквент, а девятый антецедент заносится в этой задаче в посылки. Хотя прием и относится к той же ячейке вывода, что и получаемая теорема, он непосредственно извлекается из определения предела.

Аналогичным образом исключается и второе неравенство, а далее применяются стандартные приемы исключения несущественных неизвестных и подбора значений.

Возвращаемся к рассмотрению работы приема вывод теорем. Переменной x40 присваивается элемент набора дизъюнктивных членов утверждения, получаемого после преобразования x39 к виду д.н.ф. В нашем примере x40 совпадает с x39. Проверяется, что среди конъюнктивных членов утверждения x40 нет равенства с переменной списка x31 в одной из своих частей. Переменной x41 присваивается результат подстановки терма x33 вместо переменной x15 в терм x8. В нашем примере - "$\lim_{e \to a \backslash i}(h(e) + j(e)) = b$". Переменной x42 присваивается объединение списка x37 с набором конъюнктивных членов утверждения x40. Переменной x43 присваивается результат обработки списка x42 оператором "нормантецеденты" относительно параметроы терма x41. Наконец, создается импликация с антецедентами x43 и консеквентом x41. Она обрабатывается оператором "Нормтеорема", которому разрешается использовать уже имеющиеся в списке вывода теоремы. После проверки того, что антецеденты не имеют связанных переменных, регистрируется в списке вывода.

Заметим, что приведенный прием применялся к определениям многих различных свойств функций. В общей сложности, с его участием выведено более сотни утверждений, зарегистрированных в базе теорем.

24.2 Примеры цепочек вывода теорем

Приведем несколько примеров цепочек вывода теорем. Каждая такая цепочка начинается с некоторой стартовой теоремы ячейки вывода и завершается теоремой той же ячейки. Все они реализуются системой автоматически, в одном цикле вывода для данной ячейки. Подзаголовки пунктов цепочки скопированы из оглавления приемов вывода теорем.

1. Вывод формулы корней квадратного уравнения из формулы для решения простейшего степенного уравнения.

Ячейка вывода расположена в разделе "Элементарная алгебра" - "Степени" - "Простейшие степенные уравнения" оглавления базы теорем.

(a) Стартовая теорема:

$\forall_{abc}(a - \text{число} \ \& \ c - \text{число} \ \& \ b - \text{rational} \ \& \ \text{числитель}(b) - \text{even} \ \&$
$\neg(b = 0) \to a^b = c \leftrightarrow 0 \le c \ \& \ (a = c^{1/b} \ \vee \ a = -c^{1/b}))$

(b) Попытка проварьировать эквивалентность, разрешающую утверждение с единственным вхождением неизвестной, при помощи тождества, создающего кратные вхождения неизвестной.

$\forall_{cde}(c - \text{число} \ \& \ d - \text{число} \ \& \ e - \text{число} \to 2de + d^2 = c - e^2 \leftrightarrow$
$(d = -e + \sqrt{c} \ \vee \ d = -(e + \sqrt{c})) \ \& \ 0 \le c)$

В качестве дополнительной использована теорема

$\forall_{ab}(a - \text{число} \ \& \ b - \text{число} \to (a + b)^2 = a^2 + 2ab + b^2$

(c) Обобщение эквивалентности для решения уравнения: выделение невырожденного известного подвыражения с уникальным параметром и замена его на новый параметр.

$\forall_{ade}(d - \text{число} \ \& \ e - \text{число} \ \& \ a - \text{число} \to 2de + d^2 = a \leftrightarrow$
$(d = -e + \sqrt{a + e^2} \ \vee \ d = -(e + \sqrt{a + e^2})) \ \& \ 0 \le a + e^2)$

(d) Обобщение эквивалентности для решения уравнения: выделение невырожденного известного подвыражения с уникальным параметром и замена его на новый параметр.

$\forall_{abd}(d - \text{число} \ \& \ a - \text{число} \ \& \ b - \text{число} \to bd + d^2 = a \leftrightarrow$
$(d = -(b + \sqrt{4a + b^2})/2 \ \vee \ d = (-b + \sqrt{4a + b^2})/2) \ \& \ 0 \le 4a + b^2)$

(e) Обобщение эквивалентности для разрешения относительно неизвестной: домножение обеих частей двуместного отношения на новый параметр и дистрибутивная развертка.

$\forall_{abcd}(\neg(c = 0) \ \& \ a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ d - \text{число} \to bcd + cd^2 =$
$ac \leftrightarrow (d = -(b + \sqrt{4a + b^2})/2 \ \vee \ d = (-b + \sqrt{4a + b^2})/2) \ \& \ 0 \le 4a + b^2)$

В качестве дополнительных использованы теоремы:

$\forall_{abc}(\neg(b = 0) \ \& \ a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \to ab = bc \leftrightarrow a = c)$

$\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \to ab + ac = a(b + c))$

(f) Обобщение эквивалентности для решения уравнения: выделение невырожденного известного подвыражения с уникальным параметром и замена его на новый параметр.

$\forall_{bcde}(\neg(c = 0) \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ d - \text{число} \ \& \ e - \text{число} \to bcd + cd^2 =$
$e \leftrightarrow (d = -(b + \sqrt{4e/c + b^2})/2 \ \vee \ d = (-b + \sqrt{4e/c + b^2})/2) \ \& \ 0 \le 4e/c + b^2)$

(g) Обобщение эквивалентности для решения уравнения: выделение невырожденного известного подвыражения с уникальным параметром и замена его на новый параметр.

$\forall_{acde}(\neg(c = 0) \ \& \ c - \text{число} \ \& \ d - \text{число} \ \& \ e - \text{число} \ \& \ a - \text{число} \to ad + cd^2 =$
$e \leftrightarrow (d = -(a + \sqrt{4ce + a^2})/2 \ \vee \ d = (-a + \sqrt{4ce + a^2})/2) \ \& \ 0 \le 4ce + a^2)$

2. Вывод соотношения для длин сторон треугольника и длины медианы из теоремы косинусов.

   Ячейка вывода расположена в разделе "Элементарная геометрия" - "Фигуры" - "Треугольник" - "Теорема косинусов".

   (a) Стартовая теорема:

   $\forall_{ABC}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ \neg(A = C)\ \&\ \neg(A = B) \to 2l(AB)l(AC)\cos(\angle(BAC)) = l(AB)^2 + l(AC)^2)$

   (b) Попытка использования двух соотношений для исключения тригонометрической функции угла.

   $\forall_{defg}(\neg(d = e)\ \&\ \neg(e = f)\ \&\ \neg(e = g)\ \&\ e \in \text{отрезок}(df)\ \&\ d - \text{точка}\ \&\ e - \text{точка}\ \&\ f - \text{точка}\ \&\ g - \text{точка} \to l(dg)^2l(ef) + l(fg)^2l(de) = (l(de)l(ef) + l(eg)^2)(l(de) + l(ef)))$

   Стартовая теорема используется и как основная, и как дополнительная.

   (c) Попытка отбросить избыточное отрицание равенства переменной либо константному выражению в антецедентах.

   $\forall_{defg}(f - \text{точка}\ \&\ d - \text{точка}\ \&\ \neg(e = f)\ \&\ \neg(e = g)\ \&\ e \in \text{отрезок}(df)\ \&\ e - \text{точка}\ \&\ g - \text{точка} \to l(dg)^2l(ef) + l(fg)^2l(de) = (l(de)l(ef) + l(eg)^2)(l(de) + l(ef)))$

   (d) Попытка отбросить избыточное отрицание равенства переменной либо константному выражению в антецедентах.

   $\forall_{defg}(f - \text{точка}\ \&\ d - \text{точка}\ \&\ \neg(e = g)\ \&\ e \in \text{отрезок}(df)\ \&\ e - \text{точка}\ \&\ g - \text{точка} \to l(dg)^2l(ef) + l(fg)^2l(de) = (l(de)l(ef) + l(eg)^2)(l(de) + l(ef)))$

   (e) Попытка отбросить избыточное отрицание равенства переменной либо константному выражению в антецедентах.

   $\forall_{defg}(f - \text{точка}\ \&\ d - \text{точка}\ \&\ e \in \text{отрезок}(df)\ \&\ e - \text{точка}\ \&\ g - \text{точка} \to l(dg)^2l(ef) + l(fg)^2l(de) = (l(de)l(ef) + l(eg)^2)(l(de) + l(ef)))$

   (f) Попытка отождествления двух числовых атомов для сокращения соотношения.

   $\forall_{defg}(\neg(d = e)\ \&\ l(de) = l(ef)\ \&\ e \in \text{отрезок}(df)\ \&\ d - \text{точка}\ \&\ e - \text{точка}\ \&\ f - \text{точка}\ \&\ g - \text{точка} \to -2l(ef)^2 - 2l(eg)^2 + l(dg)^2 + l(fg)^2 = 0)$

   (g) Попытка отбросить избыточное отрицание равенства переменной либо константному выражению в антецедентах.

   $\forall_{defg}(f - \text{точка}\ \&\ e - \text{точка}\ \&\ d - \text{точка}\ \&\ l(de) = l(ef)\ \&\ e \in \text{отрезок}(df)\ \&\ g - \text{точка} \to -2l(ef)^2 - 2l(eg)^2 + l(dg)^2 + l(fg)^2 = 0)$

3. Вывод формулы для расстояния от точки до прямой в трехмерном пространстве из условия перпендикулярности двух прямых.

   Ячейка вывода расположена в разделе "Аналитическая геометрия" - "Уравнение прямой в пространстве" - "Условие перпендикулярности двух прямых".

(a) Стартовая теорема:

$\forall_{ABCDKabcdefghkpqr}(a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{число}\ \&\ e - \text{число}\ \&\ f - \text{число}\ \&\ g - \text{число}\ \&\ h - \text{число}\ \&\ k - \text{число}\ \&\ p - \text{число}\ \&\ q - \text{число}\ \&\ r - \text{число}\ \&\ A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ D - \text{точка}\ \&\ \text{прямкоорд}(K)\ \&\ \neg(C = D)\ \&\ \neg(A = B)\ \&\ \text{коорд}(\text{прямая}(AB), K) = \text{set}_{xyz}(\text{пропорцнаборы}((x + a, y + b, z + c), (d, e, f))\ \&\ x - \text{число}\ \&\ y - \text{число}\ \&\ z - \text{число})\ \&\ \text{коорд}(\text{прямая}(CD), K) = \text{set}_{uvw}(\text{пропорцнаборы}((u + g, v + h, w + k), (p, q, r))\ \&\ u - \text{число}\ \&\ v - \text{число}\ \&\ w - \text{число}) \to \text{прямая}(AB) \perp \text{прямая}(CD) \leftrightarrow dp + eq + fr = 0)$

(b) Переход к импликации, выводящей соотношение для параметров уравнения.

$\forall_{abcdefghkpqrABCDK}(\neg(A = B)\ \&\ \neg(C = D)\ \&\ \text{коорд}(\text{прямая}(AB), K) = \text{set}_{xyz}(\text{пропорцнаборы}((x+a, y+b, z+c), (d, e, f))\ \&\ x - \text{число}\ \&\ y - \text{число}\ \&\ z - \text{число})\ \&\ \text{коорд}(\text{прямая}(CD), K) = \text{set}_{uvw}(\text{пропорцнаборы}((u + g, v + h, w + k), (p, q, r))\ \&\ u - \text{число}\ \&\ v - \text{число}\ \&\ w - \text{число})\ \&\ a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{число}\ \&\ e - \text{число}\ \&\ f - \text{число}\ \&\ g - \text{число}\ \&\ h - \text{число}\ \&\ k - \text{число}\ \&\ p - \text{число}\ \&\ q - \text{число}\ \&\ r - \text{число}\ \&\ A - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ D - \text{точка}\ \&\ \text{прямая}(AB) \perp \text{прямая}(CD)\ \&\ \text{прямкоорд}(K) \to dp + eq + fr = 0)$

(c) Преобразование параметрического описания для уравнения множества точек в явное задание этого уравнения и определение координат точки.

$\forall_{abcdefghijlmnptu}(\neg(a = l)\ \&\ \neg(m = n)\ \&\ t = h(b+c)+i(d+f)+j(e+q)\ \&\ u = h^2 + i^2 + j^2\ \&\ \text{коорд}(l, p) = (b, d, e)\ \&\ \text{коорд}(\text{прямая}(mn), p) = \text{set}_{qrs}(q - \text{число}\ \&\ r - \text{число}\ \&\ s - \text{число}\ \&\ \text{пропорцнаборы}((h, i, j), (c + q, f + r, g + s)))\ \&\ a \in \text{прямая}(mn)\ \&\ c - \text{число}\ \&\ f - \text{число}\ \&\ g - \text{число}\ \&\ h - \text{число}\ \&\ i - \text{число}\ \&\ j - \text{число}\ \&\ a - \text{точка}\ \&\ l - \text{точка}\ \&\ m - \text{точка}\ \&\ n - \text{точка}\ \&\ \text{прямая}(al) \perp \text{прямая}(mn)\ \&\ \text{прямкоорд}(p) \to \text{коорд}(a, p) = (-c + ht/u, -f + it/u, -g + jt/u))$

В качестве дополнительной используется теорема:

$\forall_{abcdehijklsBCK}(\neg(a = s)\ \&\ \neg(B = C)\ \&\ \text{коорд}(s, K) = (b, d, e)\ \&\ \text{коорд}(\text{прямая}(BC), K) = \text{set}_{xyz}(x - \text{число}\ \&\ y - \text{число}\ \&\ z - \text{число}\ \&\ \text{пропорцнаборы}(j, k, l), (c+x, h+y, i+z))\ \&\ a \in \text{прямая}(BC)\ \&\ c - \text{число}\ \&\ h - \text{число}\ \&\ i - \text{число}\ \&\ j - \text{число}\ \&\ k - \text{число}\ \&\ l - \text{число}\ \&\ a - \text{точка}\ \&\ s - \text{точка}\ \&\ B - \text{точка}\ \&\ C - \text{точка}\ \&\ \text{Трехмерн}(K) \to \exists_m(\text{коорд}(a, K) = (jm - c, km - h, lm - i)\ \&\ \text{коорд}(\text{прямая}(as), K) = \text{set}_{pqr}(p - \text{число}\ \&\ q - \text{число}\ \&\ r - \text{число}\ \&\ \text{пропорцнаборы}((p - b, q - d, r - e), (jm - b - c, km - d - h, lm - e - i)))\ \&\ m - \text{число}))$

После идентификации прямых исходной теоремы с прямыми дополнительной соотношение из консеквента исходной теоремы используется для определения параметра m. Далее из-под квантора существования извлекается соотношение для координат точки a.

(d) Попытка анализа контекста теоремы для получения соотношения, не содержащего точки, координаты которой определяются теоремой.

$\forall_{bcdefghijlmnp}(\neg(m = n)\ \&\ \neg(l \in \text{прямая}(mn))\ \&\ \text{коорд}(l, p) = (b, d, e)\ \&\ \text{коорд}(\text{прямая}(mn), p) = \text{set}_{qrs}(q - \text{число}\ \&\ r - \text{число}\ \&\ s - \text{число}\ \&$

пропорцнаборы$((h,i,j),(c+q,f+r,g+s)))$ & $c-$число & $f-$число & $g-$число & $h-$число & $i-$число & $j-$число & $l-$точка & $m-$точка & $n-$точка & прямкоорд$(p) \to (h(d+f)-i(b+c))^2+(i(e+g)-j(d+f))^2+(h(e+g)-j(b+c))^2=(h^2+i^2+j^2)$(расстдопрямой$(l,$прямая$(mn)))^2)$

Решается задача на исследование, посылки которой суть антецеденты и консеквент исходной теоремы. Задаче ставится цель на вывод следствий, не содержащих точки a. В частности, выводится соотношение для расстояния от точки l до прямой mn. Затем, при помощи задачи на описание, точка a исключается из антецедентов. Вместо содержавших ее антецедентов добавляется антецедент "$\neg(l \in$ прямая$(mn))$".

(e) Попытка отбросить несущественный антецедент.

$\forall_{bcdefghijlmnp}(\neg(m=n)$ & коорд$(l,p)=(b,d,e)$ & коорд(прямая$(mn),p)=$ $\text{set}_{qrs}(q-$число & $r-$число & $s-$число & пропорцнаборы$((h,i,j),(c+q,f+r,g+s)))$ & $c-$число & $f-$число & $g-$число & $h-$число & $i-$число & $j-$число & $l-$точка & $m-$точка & $n-$точка & прямкоорд$(p) \to (h(d+f)-i(b+c))^2+(i(e+g)-j(d+f))^2+(h(e+g)-j(b+c))^2=(h^2+i^2+j^2)$(расстдопрямой$(l,$прямая$(mn)))^2)$

4. Вывод формулы интегрирования по частям из определения первообразной.

Ячейка вывода расположена в разделе "Математический анализ" - "Интегрирование" - "Первообразные" - "Определение первообразной".

(a) Стартовая теорема:

$\forall_{fg}(f-$функция & $\text{Dom}(f) \subseteq \mathbb{R}$ & $\text{Val}(f) \subseteq \mathbb{R} \to$ первообразная$(f,g) \leftrightarrow g-$функция & $\text{Dom}(f)=\text{Dom}(g)$ & $\text{Val}(g) \subseteq \mathbb{R}$ & $\forall_x(x \in \text{Dom}(g) \to$ дифференцируема(g,x) & производная$(g,x)=f(x)))$

(b) Усмотрение явного выражения функции через параметры.

$\forall_g(\forall_x(x \in \text{Dom}(g) \to$ дифференцируема$(g,x))$ & $\text{Dom}(g) \subseteq \mathbb{R}$ & $\text{Val}(g) \subseteq \mathbb{R}$ & $g-$функция $\to$ первообразная$(\lambda_x($производная$(g,x), x \in \text{Dom}(g)), g))$

(c) Примерка тождеств, упрощающих выражение под описателем "отображение".

$\forall_{acde}(\text{Dom}(a)=\text{Dom}(e)$ & $\text{Dom}(e) \subseteq \mathbb{R}$ & $a-$функция & $e-$функция & первообразная(a,c) & первообразная$(e,d) \to$ первообразная$(\lambda_x(a(x)d(x)+c(x)e(x), x \in \text{Dom}(e)), \lambda_b(c(b)d(b), b \in \text{Dom}(e))))$

В качестве дополнительной используется теорема

$\forall_{aeg}(\text{Dom}(e)=\text{Dom}(g)$ & $\text{Dom}(g) \subseteq \mathbb{R}$ & $\text{Val}(e) \subseteq \mathbb{R}$ & $\text{Dom}(g) \subseteq \mathbb{R}$ & $a \in \text{Dom}(g)$ & $e-$функция & $g-$функция & дифференцируема(e,a) & дифференцируема$(g,a) \to$ производная$(\lambda_c(e(c)g(c), c \in \text{Dom}(g)), a)=e(a)$производная$(g,a)+g(a)$производная$(e,a))$

(d) Попытка исключить один из операндов для выделения новой двуместной операции.

$\forall_{acdei}(\text{Dom}(a)=\text{Dom}(e)$ & $\text{Dom}(e) \subseteq \mathbb{R}$ & $a-$функция & $e-$функция & первообразная(a,c) & первообразная(e,d) & первообразная$(\lambda_x(-c(x)e(x),$

$x \in \mathrm{Dom}(e)), i) \to$ первообразная$(\lambda_k(a(k)d(k), k \in \mathrm{Dom}(e)), \lambda_g(c(g)d(g) + i(g), g \in \mathrm{Dom}(e))))$

В качестве дополнительной используется теорема

$\forall_{acde}(\mathrm{Dom}(a) = \mathrm{Dom}(e)$ & $\mathrm{Dom}(e) \subseteq \mathbb{R}$ & $a -$ функция & $e -$ функция & первообразная(a, c) & первообразная$(e, d) \to$ первообразная$(\lambda_x(a(x)+e(x), x \in \mathrm{Dom}(e)), \lambda_b(c(b) + d(b), b \in \mathrm{Dom}(e))))$

(e) Переход к области определения исходных функций.

$\forall_{acdei}(\mathrm{Dom}(a) = \mathrm{Dom}(e)$ & $\mathrm{Dom}(e) \subseteq \mathbb{R}$ & $a -$ функция & $e -$ функция & первообразная(a, c) & первообразная(e, d) & первообразная$(\lambda_x(-c(x)e(x), x \in \mathrm{Dom}(e)), i) \to$ первообразная$(\lambda_k(a(k)d(k), k \in \mathrm{Dom}(a)), \lambda_g(c(g)d(g) + i(g), g \in \mathrm{Dom}(e))))$

(f) Попытка отбросить внешнюю одноместную операцию (антецедент).

$\forall_{acdeh}(\mathrm{Dom}(a) = \mathrm{Dom}(e)$ & $\mathrm{Dom}(e) \subseteq \mathbb{R}$ & $a -$ функция & $e -$ функция & первообразная(a, c) & первообразная(e, d) & первообразная$(\lambda_x(c(x)e(x), x \in \mathrm{Dom}(e)), h) \to$ первообразная$(\lambda_k(a(k)d(k), k \in \mathrm{Dom}(e)), \lambda_g(c(g)d(g) - h(g), g \in \mathrm{Dom}(e))))$

(g) Переход к области определения исходных функций.

$\forall_{acdeh}(\mathrm{Dom}(a) = \mathrm{Dom}(e)$ & $\mathrm{Dom}(e) \subseteq \mathbb{R}$ & $a -$ функция & $e -$ функция & первообразная(a, c) & первообразная(e, d) & первообразная$(\lambda_x(c(x)e(x), x \in \mathrm{Dom}(e)), h) \to$ первообразная$(\lambda_k(a(k)d(k), k \in \mathrm{Dom}(a)), \lambda_g(c(g)d(g) - h(g), g \in \mathrm{Dom}(e))))$

(h) Усмотрение явной выразимости параметров друг через друга.

$\forall_{abch}(\forall_f(f \in \mathrm{Dom}(b) \to$ дифференцируема$(b, f))$ & $\mathrm{Dom}(a) = \mathrm{Dom}(b)$ & $\mathrm{Dom}(b) \subseteq \mathbb{R}$ & $\mathrm{Val}(b) \subseteq \mathbb{R}$ & $a-$функция & $b-$функция & первообразная(a, c) & первообразная$(\lambda_x(c(x)$производная$(b, x), x \in \mathrm{Dom}(b)), h) \to$ превообразная$(\lambda_x(a(k)b(k), k \in \mathrm{Dom}(b)), \lambda_g(b(g)c(g) - h(g), g \in \mathrm{Dom}(b))))$

В качестве дополнительной используется теорема

$\forall_g(\forall_x(x \in \mathrm{Dom}(g) \to$ дифференцируема$(g, x))$ & $\mathrm{Dom}(g) \subseteq \mathbb{R}$ & $\mathrm{Val}(g) \subseteq \mathbb{R}$ & $g -$ функция $\to$ превообразная$(\lambda_x($производная$(g, x), x \in \mathrm{Dom}(g)), g))$

5. Вывод утверждения о перестановке столбцов определителя из определения определителя.

Ячейка вывода расположена в разделе "Линейная алгебра" - "Определители" - "Определение определителя".

(a) Стартовая теорема:

$\forall_{An}(n -$ натуральное & матр$(A, \mathbb{R}, n, n) \to \det(A) = \sum_{p,\text{перестановка}(p,\{1,\ldots,n\})}(-1)^{\text{четность}(p)} \prod_{i=1}^{n} A(i, p(i)))$

(b) Попытка замены варьируемой переменной при вычислении операции над конечным семейством.

$\forall_{acA}$(перестановка$(a, \{1, \ldots, c\})$ & матр$(A, \mathbb{R}, c, c) \rightarrow \det(A) = (-1)^{\text{четность}(a)}\det(\lambda_{df}(A(d, a(f)), d \in \{1, \ldots, c\}$ & $f \in \{1, \ldots, c\})))$

В качестве дополнительной используется теорема:

$\forall_{fhn}(n -$ натуральное & перестановка$(f, \{1, \ldots, n\})$ & перестановка$(h, \{1, \ldots, n\}) \rightarrow$ перестановка(произведение$(f, h), \{1, \ldots, n\}))$

(c) Попытка явного разрешения тождества относительно единственной численной операции над семейством.

$\forall_{acA}$(перестановка$(a, \{1, \ldots, c\})$ & матр$(A, \mathbb{R}, c, c) \rightarrow \det(\lambda_{df}(A(d, a(f)), d \in \{1, \ldots, c\}$ & $f \in \{1, \ldots, c\})) = (-1)^{\text{четность}(a)}\det(A))$

24.3 Упражнения

В приводимых ниже упражнениях требуется дать объяснение тому, как можно было бы получить заданную теорему, исходя из другой заданной теоремы, иногда с использованием явно указанных дополнительных теорем. Речь идет не о доказательстве, когда результирующая теорема известна заранее, а об "открытии" следствия, полезного относительно некоторых целевых характеристик. Соответственно, процедура вывода теорем начинает свою работу с рассмотрения вполне определенной характеристики исходной теоремы. Эта характеристика в упражнении тоже указывается.

Там, где используются дополнительные теоремы, нужно прежде всего объяснить способ их поиска в базе теорем. Обычно применяются справочники поиска теорем, приемы которых нужно создавать заблаговременно и которые перечисляют теоремы, обладающие некоторым специальным свойством. Такой справочник при решении упражнения можно предложить свой. В редких случаях, когда приемов справочника поиска теорем оказалось бы слишком много, вместо них применяется полный просмотр тех или иных разделов базы теорем, определяемых по символам, присутствующим в исходной теореме упражнения. Обычно это не сильно замедляет работу системы.

Далее, следует объяснить логику перехода от исходной и дополнительной теорем к результирующей теореме. Здесь можно использовать как обычные логические процедуры - унификацию двух или нескольких термов, подстановку в терм, замену подтерма и т.д., так и обращения к решателю для решения вспомогательных задач. Обычно вывод новой теоремы завершается обращением к процедуре "нормтеорема" либо "Нормтеорема", которые предпринимают общую стандартизацию результата. После этого нужно уточнить, какими характеристиками предполагается снабдить новую теорему. Либо такие характеристики указываются явно, либо их список, определяемый процедурой характеризатора, как-то ограничивается.

После создания нового приема вывода теорем обычно предпринимается прокрутка системы вывода по всей базе теорем, чтобы выявить точки, где новый прием порождал чрезмерно большое количество следствий либо блокировал получение ранее выводившихся теорем. При необходимости в программу приема добавляются ограничения, предотвращающие такие явления. Обычно вводится блокировка попытки

применения приема в зависимости от того, какими приемами были получены источник вывода и предшествующие ему в цепочке вывода теоремы.

Указания к упражнениям будут содержать лишь ссылки на те точки оглавления программ, в которых располагается необходимый прием вывода, а также ссылки на точки оглавления базы теорем, где расположена выводимая теорема. Разумеется, не предполагается, что целью упражнения является написание ЛОС-программы приема. Достаточно лишь определить общий план действий при получении новой теоремы из старой, который мог бы лечь в основу такой программы. Просмотр ЛОС-программ, реализующих приемы вывода теоремы, может помочь освоить технику программирования таких приемов.

Все программы приемов вывода теорем расположены в разделе "База теорем" - "Программирующий вывод" - "Приемы вывода теорем (справочник ПРОГРВЫВОД)". Пункты этого раздела соответствуют стартовым характеристикам теоремы, с которых начинается работа приемов. Поэтому ссылки на приемы будут начинаться с названия характеристики, после которой будет располагаться последовательность названий подпунктов меню оглавления. Например, "Вывод" - "Логические следствия теоремы" - "Контрапозиция для получения импликации, используемой в проверочном операторе".

После выхода на концевой пункт оглавления программ, соответствующий приему, можно посмотреть ЛОС-программу приема, нажав "курсор вправо". Это нажатие переводит в концевую часть программы. Программа относится к справочнику "прогрвывод", т.е. на момент обращения к ней определены следующие значения: x1 - текущая четверка (вес - исходная теорема - характеристики - блок вывода) списка вывода, x2 - исходная теорема, x3 - список ее характеристик, x4 - блок вывода, x5 - пара (список вывода - индикатор изменений), x6 - уровень сканирования, x7 - текущая характеристика исходной теоремы, по которой срабатывает прием.

Из того же концевого пункта оглавления программ можно посмотреть, какие теоремы были выведены с помощью данного приема. Для этого достаточно, вместо "курсор вправо", нажать "л". Появится список теорем, в котором можно будет найти ту теорему, которая должна выводиться в упражнении. Клавишами "курсор вверх - курсор вниз" можно выделить номер нужной теоремы и нажать "курсор вправо". Появится кадр, содержащий в верхней части результирующую теорему, а в нижней - исходную теорему и дополнительные теоремы. Между ними расположено название приема. Если нужно проследить дальнейшую предысторию вывода, снова используются клавиши курсора: "вправо" - вход в предыдущий шаг вывода, "влево" - возвращение. Чтобы посмотреть, как работала программа приема вывода, нажимается клавиша "ъ". Она переводит в отладчик ЛОСа, остановленный внутри процедуры "ктв". Нажимая "Page Up", попадаем в программу "регтеор", и после еще одного нажатия клавиши "Page Up" - в программу приема вывода. Здесь обычными средствами отладчика ЛОСа можно просматривать значения программных переменных, и таким образом анализировать работу приема вывода. Для возвращения в просмотр цепочки вывода теоремы нажимается "0" и "Enter".

В 9 томе монографии "Компьютерное моделирование логических процессов" приведены подробные описания действий ЛОС-программ приемов вывода теорем. По сути, это просто переводы ЛОС-програм на русский язык. Они размещены так же, как в оглавлении программ - по своим стартовым характеристикам.

Наконец, вместо просмотра программы приема вывода, можно перейти в базе теорем к просмотру той теоремы, вывод которой требуется объяснить в упражнении. После выхода на просмотр теоремы нужно убедиться, что горизонтальная линия под характеристиками теоремы - зеленая. Зеленый цвет означает, что система способна выводить данную теорему из стартовых теорем ячейки вывода. Впрочем, для упражнений это заведомо выполнено. Далее нажимается "д", которое переводит в просмотр первого кадра цепочки вывода теоремы. Эта цепочка - та же, что и при указанном выше просмотре вывода теоремы через концевой пункт оглавления программ. Нажатием "ъ", как и выше, можно попасть в кадр отладчика ЛОСа, завершающий применение приема.

Переходим к перечислению упражнений.

1. Вывести теорему

   $\forall_{bc}(b - \text{set} \ \& \ c - \text{set} \to b \setminus c = \text{set}_a(a \in b \ \& \ \neg(a \in c)))$

   из теоремы

   $\forall_{abc}(b - \text{set} \ \& \ c - \text{set} \to a \in b \setminus c \leftrightarrow a \in b \ \& \ \neg(a \in c))$

   Стартовая характеристика вывода - "принадлежит(второйтерм)" (определение принадлежности множеству).

2. Вывести теорему

   $\forall_{bef}(b - \text{set} \ \& \ e - \text{set} \ \& \ f - \text{set} \to \text{непересек}(b, e \cup f) \leftrightarrow \text{непересек}(b, e) \ \& \ \text{непересек}(b, f))$

   из теоремы

   $\forall_{ab}(a - \text{set} \ \& \ b - \text{set} \to \text{непересек}(a, b) \leftrightarrow \forall_c(c \in a \to \neg(c \in b)))$

   и дополнительной теоремы

   $\forall_{abc}(b - \text{set} \ \& \ c - \text{set} \to a \in b \cup c \leftrightarrow a \in b \ \vee \ a \in c)$

   Стартовая характеристика - "кванторнаясвертка(первыйтерм)".

3. Вывести теорему

   $\forall_{af}(a - \text{set} \ \& \ f - \text{функция} \to \text{образ}(f, \text{прообраз}(f, a)) \subseteq a)$

   из теоремы

   $\forall_{afx}(f - \text{функция} \ \& \ a - \text{set} \to x \in \text{прообраз}(f, a) \leftrightarrow x \in \text{Dom}(f) \ \& \ f(x) \in a)$

   Стартовая характеристика - "определение($x \in$ прообраз(f, a))".

4. Вывести теорему

   $\forall_{ac}(\neg(a = \emptyset) \ \& \ a - \text{set} \ \& \ a \subseteq \mathbb{R} \ \& \ \text{верхняягрань}(c, a) \to \sup(a) \leq c)$

   из теоремы

   $\forall_a(a - \text{set} \ \& \ \neg(a = \emptyset) \ \& \ a \subseteq \mathbb{R} \ \& \ \text{огрсверху}(a) \to \text{наименьший}(\sup(a), \text{set}_b(\text{верхняягрань}(b, a))))$

   Стартовая характеристика - "определение($\sup(a)$)".

5. Вывести теорему

   $\forall_{acd}(a - \text{set} \ \& \ c - \text{set} \ \& \ d - \text{set} \ \& \ a \subseteq c \cup d \ \& \ \text{конечное}(c) \ \& \ \text{конечное}(d) \rightarrow \text{card}(c) + \text{card}(d) = \text{card}(a) \leftrightarrow a = c \cup d \ \& \ \text{непересек}(c, d))$

   из теоремы

   $\forall_{ab}(a - \text{set} \ \& \ b - \text{set} \ \& \ a \subseteq b \ \& \ \text{конечное}(b) \rightarrow \text{card}(a) = \text{card}(b) \leftrightarrow a = b)$

   и дополнительной теоремы

   $\forall_{ab}(a - \text{set} \ \& \ b - \text{set} \ \& \ \text{конечное}(a) \ \& \ \text{конечное}(b) \rightarrow \text{card}(a \cup b) = \text{card}(a) + \text{card}(b) \leftrightarrow \text{непересек}(a, b))$

   Стартовая характеристика - "усиление(второйтерм)".

6. Вывести теорему

   $\forall_{defghi}(0 < d \ \& \ d - \text{число} \ \& \ e - \text{число} \ \& \ f - \text{число} \ \& \ g - \text{число} \ \& \ h - \text{число} \ \& \ i - \text{число} \rightarrow gd^{e+f} + hd^{e+i} = (gd^f + hd^i)d^e)$

   из теоремы

   $\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \rightarrow ab + ac = a(b + c))$

   и дополнительной теоремы

   $\forall_{abc}(a - \text{число} \ \& \ b - \text{число} \ \& \ c - \text{число} \ \& \ 0 < a \rightarrow a^b a^c = a^{b+c})$

   Стартовая характеристика - "свертка(второйтерм)".

7. Вывести теорему

   $\forall_{ade}(a - \text{число} \ \& \ d - \text{число} \ \& \ e - \text{число} \rightarrow d = e - a \leftrightarrow e = a + d)$

   из теоремы

   $\forall_{bcd}(b - \text{число} \ \& \ c - \text{число} \ \& \ d - \text{число} \rightarrow c = d \leftrightarrow b + c = b + d)$

   и дополнительной теоремы

   $\forall_a(a - \text{число} \rightarrow a - a = 0)$

   Стартовая характеристика - "общнорм(первыйтерм)".

8. Вывести теорему

   $\forall_{acdef}(\neg(a = 0) \ \& \ 0 < df \ \& \ a - \text{число} \ \& \ c - \text{число} \ \& \ d - \text{число} \ \& \ e - \text{число} \ \& \ f - \text{число} \rightarrow e(f/d)^c/a = e/(a(d/f)^c))$

   из теоремы

   $\forall_{cdef}(0 < df \ \& \ c - \text{число} \ \& \ d - \text{число} \ \& \ e - \text{число} \ \& \ f - \text{число} \rightarrow e(f/d)^c = e/(d/f)^c)$

   и дополнительной теоремы

$\forall_{ade}(\neg(a = 0)\ \&\ \neg(e = 0)\ \&\ a - \text{число}\ \&\ d - \text{число}\ \&\ e - \text{число} \rightarrow$
$d/(ae) = (d/e)/a)$

Стартовая характеристика - "нормализация(первыйтерм)".

9. Вывести теорему

   $\forall_{ab}(a - \text{число}\ \&\ b - \text{rational}\ \&\ \neg(\text{знаменатель}(b) - \text{even}) \rightarrow 0 < a^b \leftrightarrow$
   $0 < a\ \&\ \neg(\text{числитель}(b) - \text{even})\ \vee\ \text{числитель}(b) - \text{even}\ \&\ \neg(a = 0))$

   из теоремы

   $\forall_{ab}(a - \text{число}\ \&\ b - \text{rational}\ \&\ \neg(\text{знаменатель}(b) - \text{even})\ \&$
   $\neg(\text{числитель}(b) - \text{even}) \rightarrow 0 < a^b \leftrightarrow 0 < a)$

   и дополнительной теоремы

   $\forall_{ab}(a - \text{число}\ \&\ \text{числитель}(b) - \text{even}\ \&\ b - \text{rational} \rightarrow 0 < a^b \leftrightarrow \neg(a = 0))$

   Стартовая характеристика - "общнорм(второйтерм)".

10. Вывести теорему

    $\forall_{aef}(\neg(\cos e = 0)\ \&\ \neg(\cos f = 0)\ \&\ a - \text{число}\ \&\ e - \text{число}\ \&\ f - \text{число} \rightarrow$
    $a \operatorname{tg} e + a \operatorname{tg} f = a \sin(e + f)/(\cos e \cos f))$

    из теоремы

    $\forall_{bcf}(\neg(\cos f = 0)\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ f - \text{число} \rightarrow c \sin b + c \cos b \operatorname{tg} f =$
    $c \sin(b + f)/\cos f)$

    и дополнительной теоремы

    $\forall_{abx}(a - \text{число}\ \&\ b - \text{число}\ \&\ x - \text{число}\ \&\ \neg(\cos x = 0) \rightarrow (a \sin x + b \cos x)/\cos x =$
    $a \operatorname{tg} x + b)$

    Стартовая характеристика - "упрощение(второйтерм)".

11. Вывести теорему

    $\forall_{ab}(a - \text{число}\ \&\ 0 \leq a + 1\ \&\ 0 \leq -a + 1 \rightarrow b = \arcsin(a) \leftrightarrow a = \sin b\ \&$
    $b - \text{число}\ \&\ 0 \leq b + \pi/2\ \&\ 0 \leq -b + \pi/2)$

    из теоремы

    $\text{обрфункция}(\lambda_x(\sin x, x \in [-\pi/2, \pi/2])) = \lambda_x(\arcsin(x), x \in [-1, 1])$

    и дополнительной теоремы

    $\forall_{fab}(f - \text{функция}\ \&\ \text{взаимнооднозначно}(f)\ \&\ a \in \operatorname{Val}(f) \rightarrow$
    $\text{обрфункция}(f)(a) = b \leftrightarrow b \in \operatorname{Dom}(f)\ \&\ f(b) = a)$

    Стартовая характеристика - "опр(арксинус)". Такая характеристика означает, что теорема является определением функции, обозначенной символом "арксинус".

12. Вывести теорему

$\forall_{abcd}(\neg(a=b)\ \&\ \neg(a=c)\ \&\ \neg(a=d)\ \&\ \neg(b=d)\ \&\ \neg(c=d)\ \&$ биссектриса$(bacd)$ $\&\ a-$точка $\&\ b-$точка $\&\ c-$точка $\&\ d-$точка $\&$ прямая$(ab)\perp$ прямая$(bd)\ \&$ прямая$(ac)\perp$ прямая$(cd)\to l(cd)=l(bd))$

из теоремы

$\forall_{ABCD}(\neg(A=B)\ \&\ \neg(A=C)\ \&\ \neg(A=D)\ \&$ биссектриса$(BACD)\ \&$ $A-$точка $\&\ B-$точка $\&\ C-$точка $\&\ D-$точка $\to\angle(CAD)=\angle(BAD))$

и дополнительной теоремы

$\forall_{ABCDEF}(A-$точка $\&\ B-$точка $\&\ C-$точка $\&\ D-$точка $\&\ E-$точка $\&$ $F-$точка $\&\ \neg(D=F)\ \&\ \neg(A=C)\ \&\ \neg(E=F)\ \&\ \neg(D=E)\ \&$ $\neg(B=C)\ \&\ \neg(A=B)\ \&$ прямая$(AB)\perp$ прямая$(BC)\ \&$ прямая$(DE)\perp$ прямая$(EF)\ \&\ l(AC)=l(DF)\ \&\ \angle(ACB)=\angle(EFD)\to l(AB)=l(DE))$

Дополнительная теорема - признак равенства прямоугольных треугольников по гипотенузе и острому углу. Стартовая характеристика исходной теоремы - "равны".

13. Вывести теорему

$\forall_{aBC}(\neg(a=B)\ \&\ \neg(a=C)\ \&\ \neg(B=C)\ \&\ a-$точка $\&\ B-$точка $\&\ C-$точка $\to$ $\sin(\angle(BaC))l(aC)=\sin(\angle(aBC))l(BC))$

из теоремы

$\forall_{ABC}(A-$точка $\&\ B-$точка $\&\ C-$точка $\&\ \neg(B=C)\ \&\ \neg(A=C)\ \&$ $\neg(A=B)\ \&$ прямая$(AB)\perp$ прямая$(AC)\to l(AC)=\sin(\angle(ABC))l(BC))$

Стартовая характеристика - "числовойатом".

14. Вывести теорему

$\forall_{abcdeAB}(\neg(a=b)\ \&\ \neg(a=d)\ \&\ \neg(a=e)\ \&\ \neg(b=d)\ \&\ \neg(b=e)\ \&$ $\neg(A=B)\ \&\ \neg(c\in$ прямая$(ab))\ \&\ a\in$ отрезок$(cd)\ \&\ a\in$ окружность$(AB)\ \&$ $b\in$ отрезок$(ce)\ \&\ b\in$ окружность$(AB)\ \&\ d\in$ окружность$(AB)\ \&$ $e\in$ окружность$(AB)\ \&\ a-$точка $\&\ b-$точка $\&\ c-$точка $\&\ d-$точка $\&$ $e-$точка $\&\ A-$точка $\&\ B-$точка $\to\angle(bdc)=\angle(aec))$

из теоремы

$\forall_{ABCDEF}(A-$точка $\&\ B-$точка $\&\ C-$точка $\&\ D-$точка $\&\ E-$точка $\&$ $F-$точка $\&\ \neg(D=F)\ \&\ \neg(C=F)\ \&\ \neg(D=E)\ \&\ \neg(C=E)\ \&$ $\neg(C=D)\ \&\ \neg(A=B)\ \&\ C\in$ окружность$(AB)\ \&\ D\in$ окружность$(AB)\ \&$ $E\in$ окружность$(AB)\ \&\ F\in$ окружность$(AB)\ \&$ окрестность$(E,F,$ прямая$(CD))\to\angle(CED)=\angle(CFD))$

и дополнительной теоремы

$\forall_{ABCDE}(A-$точка $\&\ B-$точка $\&\ C-$точка $\&\ D-$точка $\&\ E-$точка $\&$ $\neg(A=B)\ \&\ A\in$ отрезок$(CD)\ \&\ B\in$ отрезок$(CE)\ \&$ $\neg(C\in$ прямая$(AB))\to$ однасторона$(D,E,$ прямая$(AB)))$

Выводимая теорема связана с двумя секущими, проведенными из общей точки. Стартовая характеристика - "числовойатом".

15. Вывести теорему

$\forall_{efglmABK}$(коорд(вектор(fB), K) = (l, m) & коорд(вектор(Af), K) = (e, g) & f – точка & A – точка & B – точка & систкоорд(K) → коорд(вектор(AB), K) = ($e + l, g + m$))

из теоремы

$\forall_{ABKabcd}$(систкоорд(K) & A – точка & B – точка & коорд(A, K) = (a, b) & коорд(B, K) = (c, d) → коорд(вектор(AB), K) = ($c - a, d - b$))

и дополнительной теоремы

$\forall_{ABC}$(A – точка & B – точка & C – точка → вектор(AB) + вектор(BC) = вектор(AC))

Стартовая характеристика - "определение(коорд(вектор(AB),K))".

16. Вывести теорему

$\forall_{deghijK}$(¬($d = e$) & коорд(d, K) = (g, h) & коорд(e, K) = (i, j) & d – точка & e – точка & систкоорд(K) → коорд(прямая(de), K) = set_{xy}($hi + x(j - h) + y(g - i) - gj = 0$ & x – число & y – число))

из теоремы

$\forall_{ABCKP}$(систкоорд(K) & $K = (A, B, C)$ & $P \subseteq$ плокость(ABC) → Прямая(P) ↔ $\exists_{abc}$(a – число & b – число & c – число & ¬($a^2 + b^2 = 0$) & коорд(P, K) = set_{xy}(x – число & y – число & $ax + by + c = 0$)))

и дополнительной теоремы

$\forall_{ABP}$(Прямая(P) & A – точка & B – точка & $A \in P$ & $B \in P$ & ¬($A = B$) → P = прямая(AB))

Стартовая характеристика - "уравнкрив".

17. Вывести теорему

$\forall_{abcdef}$(¬($d^2 + e^2 + f^2 = 0$) & a – число & b – число & c – число & d – число & e – число & f – число → set_{xyz}(x – число & y – число & z – число & пропорциональны((d, e, f), ($x - a, y - b, z - c$))) = set_{xyz}($\exists_g$($x = a + dg$ & $y = b + eg$ & $z = c + fg$ & g – число)))

из теоремы

$\forall_{KP}$(P – set & систкоорд(K) & Трехмерн(K) & $P \subseteq$ Точки → Прямая(P) ↔ $\exists_{abcdef}$(a – число & b – число & c – число & d – число & e – число & f – число & ¬($d^2 + e^2 + f^2 = 0$) & коорд(P, K) = set_{xyz}(x – число & y – число & z – число & пропорцнаборы(($x - a, y - b, z - c$), (d, e, f)))))

и дополнительной теоремы

$\forall_{abcdef}(a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{число}\ \&\ e - \text{число}\ \&\ f - \text{число} \rightarrow$
$\text{пропорцнаборы}((a,b,c),(d,e,f)) \leftrightarrow \exists_{uv}(u - \text{число}\ \&\ v - \text{число}\ \&$
$\neg(u^2+v^2=0)\ \&\ au+dv=0\ \&\ bu+ev=0\ \&\ cu+fv=0))$

Стартовая характеристика - "уравнкрив".

18. Вывести теорему

$\forall_{bcdefghijlmnp}(\neg(m=n)\ \&\ \neg(l \in \text{прямая}(mn))\ \&\ \text{коорд}(l,p)=(b,d,e)\ \&$
$\text{коорд}(\text{прямая}(mn),p)=\text{set}_{qrs}(q - \text{число}\ \&\ r - \text{число}\ \&\ s - \text{число}\ \&$
$\text{пропорцнаборы}((h,i,j),(c+q,f+r,g+s)))\ \&\ c - \text{число}\ \&$
$f - \text{число}\ \&\ g - \text{число}\ \&\ h - \text{число}\ \&\ i - \text{число}\ \&\ j - \text{число}\ \&\ l - \text{точка}\ \&$
$m - \text{точка}\ \&\ n - \text{точка}\ \&\ \text{прямкоорд}(p) \rightarrow (h(d+f)-i(b+c))^2+(i(e+g)-$
$j(d+f))^2+(h(e+g)-j(b+c))^2=(h^2+i^2+j^2)(\text{расстдопрямой}(l,\text{прямая}(mn)))^2)$

из теоремы

$\forall_{abcdefghijlmnptu}(\neg(a=l)\ \&\ \neg(m=n)\ \&\ t=h(b+c)+i(d+f)+j(e+g)\ \&$
$u=h^2+i^2+j^2\ \&\ \text{коорд}(l,p)=(b,d,e)\ \&\ \text{коорд}(\text{прямая}(mn),p)=$
$\text{set}_{qrs}(q - \text{число}\ \&\ r - \text{число}\ \&\ s - \text{число}\ \&\ \text{пропорцнаборы}((h,i,j),(c+q,f+r,$
$g+s)))\ \&\ a \in \text{прямая}(mn)\ \&\ c - \text{число}\ \&\ f - \text{число}\ \&\ g - \text{число}\ \&\ h - \text{число}\ \&$
$i - \text{число}\ \&\ j - \text{число}\ \&\ a - \text{точка}\ \&\ l - \text{точка}\ \&\ m - \text{точка}\ \&\ n - \text{точка}\ \&\ \text{прямая}(al) \perp$
$\text{прямая}(mn)\ \&\ \text{прямкоорд}(p) \rightarrow \text{коорд}(a,p)=(-c+ht/u,-f+it/u,-g+jt/u))$

Стартовая характеристика - "систкоорд".

19. Вывести теорему

$\forall_{abABK}(\neg(A=B)\ \&\ \text{коорд}(A,K)=(a,b)\ \&\ A - \text{точка}\ \&\ B - \text{точка}\ \&$
$\text{прямкоорд}(K) \rightarrow \exists_c(\text{коорд}(\text{окружность}(AB),K)=\text{set}_{xy}(c-2ax-2by+$
$x^2+y^2=0\ \&\ x - \text{число}\ \&\ y - \text{число})\ \&\ c - \text{число}))$

из теоремы

$\forall_{abABK}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ \text{прямкоорд}(K)\ \&\ \neg(A=B)\ \&$
$\text{коорд}(A,K)=(a,b) \rightarrow \text{коорд}(\text{окружность}(AB),K)=\text{set}_{xy}((x-a)^2+$
$(y-b)^2=l(AB)^2\ \&\ x - \text{число}\ \&\ y - \text{число}))$

Стартовая характеристика - "уравнмножество".

20. Вывести теорему

$\forall_a(a - \text{число} \rightarrow \lim(\lambda_c(a, c - \text{натуральное}))=a)$

из теоремы

$\forall_{ab}(a - \text{число}\ \&\ \text{последовательность}(b,\mathbb{R}) \rightarrow \lim(b)=a \leftrightarrow \forall_e(e - \text{число}\ \&$
$0<e \rightarrow \exists_n(n - \text{натуральное}\ \&\ \forall_m(m - \text{натуральное}\ \&\ n \le m \rightarrow$
$|b(m)-a|<e))))$

Стартовая характеристика - "определение(равно(пределпослед(b) a))".

21. Вывести теорему

$\forall_{aeg}(\mathrm{Dom}(e) = \mathrm{Dom}(g)\ \&\ \mathrm{Dom}(g) \subseteq \mathbb{R}\ \&\ \mathrm{Val}(e) \subseteq \mathbb{R}\ \&\ \mathrm{Val}(g) \subseteq \mathbb{R}\ \&$
$a \in \mathrm{Dom}(g)\ \&\ e -$ функция $\&\ g -$ функция $\&$ дифференцируема$(e, a)\ \&$
дифференцируема$(g, a) \to$ производная$(\lambda_c(e(c) + g(c), c \in \mathrm{Dom}(g)), a) =$
производная$(e, a) +$ производная$(g, a))$

из теоремы

$\forall_{abf}(f -$ функция $\&\ \mathrm{Dom}(f) \subseteq \mathbb{R}\ \&\ \mathrm{Val}(f) \subseteq \mathbb{R}\ \&\ a \in \mathrm{Dom}(f)\ \&$
типпредела$(b)\ \&$ локопред$(f, a, b)\ \&$ дифференцируема$(f, a) \to$
производная$(f, a) = \lim_{x \to a \setminus b}(f(x) - f(a))/(x - a))$

Стартовая характеристика - "определение(производная(f, a))".

22. Вывести теорему

$\forall_{abch}(\forall_f(f \in \mathrm{Dom}(b) \to$ дифференцируема$(b, f))\ \&\ \mathrm{Dom}(a) = \mathrm{Dom}(b)\ \&$
$\mathrm{Dom}(b) \subseteq \mathbb{R}\ \&\ \mathrm{Val}(b) \subseteq \mathbb{R}\ \&\ a -$ функция $\&\ b -$ функция $\&$ первообразная(a, c)
$\&$ первообразная$(\lambda_x(c(x)$производная$(b, x), x \in \mathrm{Dom}(b)), h) \to$
первообразная$(\lambda_k(a(k)b(k), k \in \mathrm{Dom}(b)), \lambda_g(b(g)c(g) - h(g), g \in \mathrm{Dom}(b))))$

из теоремы

$\forall_{acdeh}(\mathrm{Dom}(a) = \mathrm{Dom}(e)\ \&\ \mathrm{Dom}(e) \subseteq \mathbb{R}\ \&\ a -$ функция $\&\ e -$ функция $\&$
первообразная$(a, c)\ \&$ первообразная$(e, d)\ \&$ первообразная$(\lambda_x(c(x)e(x),$
$x \in \mathrm{Dom}(e)), h) \to$ первообразная$(\lambda_k(a(k)d(k), k \in \mathrm{Dom}(a)), \lambda_g(c(g)d(g) - h(g),$
$g \in \mathrm{Dom}(e))))$

и дополнительной теоремы

$\forall_g(\forall_x(x \in \mathrm{Dom}(g) \to$ дифференцируема$(g, x))\ \&\ \mathrm{Dom}(g) \subseteq \mathbb{R}\ \&\ \mathrm{Val}(g) \subseteq \mathbb{R}\ \&$
$g -$ функция $\to$ первообразная$(\lambda_x($производная$(g, x), x \in \mathrm{Dom}(g)), g))$

Стартовая характеристика - "функперех(1)". Напомним, что такая характеристика указывает на частный случай определения функциональной характеристики. При этом 1 - номер операнда консеквента, на котором расположена характеризуемая функция. Предлагаемый переход представляет собой последнее звено цепочки вывода формулы интегрирования по частям из определения первообразной.

23. Вывести теорему

$\forall_{acde}(\mathrm{Dom}(d) = c\ \&\ \mathrm{Dom}(e) = c\ \&\ a -$ set $\&\ a \subseteq c\ \&\ c \subseteq \mathbb{R}\ \&$
$\mathrm{Val}(d) \subseteq [0, \infty)\ \&\ \mathrm{Val}(e) \subseteq (0, \infty)\ \&\ d -$ функция $\&\ e -$ функция $\&$
невозрастает$(e, a)\ \&$ возрастает$(d, a) \to$ возрастает$(\lambda_b(d(b)/e(b), b \in c), a))$

из теоремы

$\forall_{fga}(a -$ set $\&\ f -$ функция $\&\ a \subseteq \mathrm{Dom}(f)\ \&\ \mathrm{Dom}(f) \subseteq \mathbb{R}\ \&\ \mathrm{Val}(f) \subseteq \mathbb{R} \to$
возрастает$(f, a) \leftrightarrow \forall_{xy}(x \in a\ \&\ y \in a\ \&\ 0 < x - y \to 0 < f(x) - f(y)))$

Стартовая характеристика - "определение(возрастает(f, a))".

24. Вывести теорему

$\forall_{acA}(\text{перестановка}(a,\{1,\ldots,c\})\ \&\ \text{матр}(A,\mathbb{R},c,c) \to \det(A) = (-1)^{\text{четность}(a)}\det(\lambda_{df}(A(d,a(f)), d \in \{1,\ldots,c\}\ \&\ f \in \{1,\ldots,c\})))$

из теоремы

$\forall_{An}(n - \text{натуральное}\ \&\ \text{матр}(A,\mathbb{R},n,n) \to \det(A) = \sum_{p,\text{перестановка}(p,\{1,\ldots,n\})}(-1)^{\text{четность}(p)}\prod_{i=1}^{n} A(i,p(i)))$

и дополнительной теоремы

$\forall_{fhn}(n - \text{натуральное}\ \&\ \text{перестановка}(f,\{1,\ldots,n\})\ \&\ \text{перестановка}(h,\{1,\ldots,n\}) \to \text{перестановка}(\text{произведение}(f,h),\{1,\ldots,n\}))$

Стартовая характеристика - "описатель(первыйтерм)".

Затем, из выведенной теоремы, вывести теорему

$\forall_{acA}(\text{перестановка}(a,\{1,\ldots,c\})\ \&\ \text{матр}(A,\mathbb{R},c,c) \to \det(\lambda_{df}(A(d,a(f)), d \in \{1,\ldots,c\}\ \&\ f \in \{1,\ldots,c\})) = (-1)^{\text{четность}(a)}\det(A))$

Стартовая характеристика та же - "описатель(второйтерм)".

Обычно теорема, зарегистрированная в ячейке вывода, получается из стартовых теорем ячейки не одним, а несколькими шагами. Далее предлагаются упражнения, в которых нужно восстановить всю цепочку вывода и наметить приемы вывода для каждого перехода, включая способы поиска дополнительных теорем.

25. Предложить цепочку шагов для вывода теоремы

$\forall_{abcdABK}(a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{число}\ \&\ A - \text{точка}\ \&\ B - \text{точка}\ \&\ \neg(A = B)\ \&\ \text{прямкоорд}(K)\ \&\ \text{коорд}(\text{окружность}(AB),K) = \text{set}_{xy}(d{+}ax^2{+}ay^2{+}bx{+}cy = 0\ \&\ x{-}\text{число}\ \&\ y{-}\text{число}) \to 4a^2l(AB)^2 = b^2{+}c^2{-}4ad)$

из теоремы

$\forall_{abABK}(A - \text{точка}\ \&\ B - \text{точка}\ \&\ \text{прямкоорд}(K)\ \&\ \text{коорд}(A,K) = (a,b) \to \text{коорд}(\text{окружность}(AB),K) = \text{set}_{xy}((x-a)^2 + (y-b)^2 = l(AB)^2\ \&\ x - \text{число}\ \&\ y - \text{число}))$

26. Предложить цепочку шагов для вывода теоремы

$\forall abcde(\neg(a = 0)\ \&\ \neg(b - 1 = 0)\ \&\ \neg(d - 1 = 0)\ \&\ \neg(e - 1 = 0)\ \&\ 0 < b\ \&\ 0 < d\ \&\ 0 < e\ \&\ a - \text{число}\ \&\ b - \text{число}\ \&\ c - \text{число}\ \&\ d - \text{число}\ \&\ e - \text{число} \to d^{c/(a\log_b e)} = b^{c/(a\log_d e)})$

из теоремы

$\forall_{ab}(a - \text{число}\ \&\ b - \text{число}\ \&\ 0 < a\ \&\ \neg(a - 1 = 0)\ \&\ 0 < b \to a^{\log_a b} = b)$

27. Предложить цепочку шагов для вывода теоремы

    $\forall_{abdef}(\neg(a=b)\ \&\ \neg(e=f)\ \&\ \neg(d=e)\ \&\ l(de)=l(ef)\ \&\ d\in$ окружность$(ab)\ \&$ $e\in$ окружность$(ab)\ \&\ f\in$ окружность$(ab)\ \&\ a-$ точка $\&\ b-$ точка $\&$ $d-$ точка $\&\ e-$ точка $\&\ f-$ точка $\&$ разныестороны$(d,f,$ прямая$(ae))\to$ биссектриса$(defa))$

    из теоремы

    $\forall_{ABCDEF}(A-$ точка $\&\ B-$ точка $\&\ C-$ точка $\&\ D-$ точка $\&\ E-$ точка $\&$ $F-$ точка $\&\ \neg(A=B)\ \&\ C\in$ окружность$(AB)\ \&\ D\in$ окружность$(AB)\ \&$ $E\in$ окружность$(AB)\ \&\ F\in$ окружность$(AB)\to l(CD)=l(EF)\leftrightarrow\angle(CAD)=$ $\angle(EAF))$

28. Предложить цепочку шагов для вывода теоремы

    $\forall_{af}(\neg(\cos(f(a))=0)\ \&\ \mathrm{Dom}(f)\subseteq\mathbb{R}\ \&\ \mathrm{Val}(f)\subseteq\mathbb{R}\ \&\ a\in\mathrm{Dom}(f)\ \&$ $f-$ функция $\&$ дифференцируема$(f,a)\to$ производная$(\lambda_x(\mathrm{tg}\,f(x),$ $x\in\mathrm{Dom}(f)),a)=$ производная$(f,a)/(\cos f(a))^2)$

    из теоремы

    $\forall_{fga}(f-$функция $\&\ g-$функция $\&\ \mathrm{Dom}(f)\subseteq\mathbb{R}\ \&\ \mathrm{Dom}(g)\subseteq\mathbb{R}\ \&\ \mathrm{Val}(f)\subseteq\mathbb{R}\ \&$ $\mathrm{Val}(g)\subseteq\mathbb{R}\ \&\ a\in\mathrm{Dom}(f)\ \&$ дифференцируема$(f,a)\ \&$ дифференцируема$(g,$ $f(a))\to$ дифференцируема$(\lambda_x(g(f(x)),x\in\mathrm{Dom}(f)),a)\ \&$ производная$(\lambda_x(g(f(x)),x\in\mathrm{Dom}(f)),a)=$ производная$(g,f(a))\cdot$ производная$(f,a))$

29. Предложить цепочку шагов для вывода теоремы

    $\forall_{abce}(\neg($знаменатель$(b)-$ even$)\ \&\ \neg($числитель$(b)-$ even$)\ \&\ b-$ rational $\&$ первообразная$(\lambda_j(a(j)(-j^2+1)^{(b-1)/2}/e(j),j-$ число$),c)\to$ первообразная$(\lambda_x(a(\sin x)(\cos x)^b/e(\sin x),x-$ число$),\lambda_d(c(\sin d),d-$ число$)))$

    из теоремы

    $\forall_{ace}(\forall_f(f\in\mathrm{Dom}(e)\to$ дифференцируема$(e,f))\ \&\ \mathrm{Dom}(a)\subseteq\mathbb{R}\ \&$ $\mathrm{Dom}(e)\subseteq\mathbb{R}\ \&\ \mathrm{Val}(e)\subseteq\mathbb{R}\ \&\ a-$функция $\&\ e-$функция $\&$ первообразная$(a,c)\to$ первообразная$(\lambda_x(a(e(x))$производная$(e,x),x\in\mathrm{Dom}(e)),\lambda_d(c(e(d)),$ $d\in\mathrm{Dom}(e))))$

Указания

1. Используется прием "Принадлежит" - "Логические следствия теоремы" - "Вывод из определения принадлежности равенства для определяемогомножества". Путь к теореме в базе теорем - "Теория множеств" - "Разность множеств" - "Определение принадлежности разности множеств" - "Свертка описателя "класс".

2. Используется прием "Кванторнаясвертка" - "Использование дополнительной эквивалентности для преобразования заменяемой части" - "Применение дизъюнктивной декомпозирующей эквивалентности к антецеденту заменяемой импликации и последующая свертка подимпликаций". Путь к теореме в базе теорем - "Теория множеств" - "Непересекающиеся множества" - "Определение непересекающихся множеств" - "Непересечение с объединением".

3. Используется прием "Определение" - "Логические следствия теоремы" - "Извлечение из эквивалентности с конъюнкцией в заменяющей части импликации для усмотрения конъюнктивного члена определяющей части". Путь к теореме ва базе теорем - "Теория множеств" - "Простейшие свойства функций" - "Прообраз множества" - "Определение принадлежности прообразу множества" - "Образ прообраза".

4. Используется прием "Определение" - "Использование задачи на исследование для вывода следствий" - "Вывод не содержащих описателей следствий из консеквента, содержащего описатель "класс" ". Путь к теореме в базе теорем - "Теория множеств" - "Числовые множества" - "Точная верхняя грань" - "Определение точной верхней грани" - "Любая верхняя грань не меньше точной верхней грани".

5. Используется прием "Усиление" - "Использование дополнительной эквивалентности для упрощения заменяемой части" - "Использование эквивалентности, выражающей условие декомпозируемости числового атома". Путь к теореме в базе теорем - "Теория множеств" - "Мощности множеств"/ - "Мощность разности вложенных конечных множеств" - "Сумма мощностей двух множеств равна мощности третьего".

6. Используется прием "Свертка" - "Применение дополнительного тождества для упрощения заменяемого терма" - "Использование нормализующего тождества типа свертки для варьирования тождества свертки". Путь к теореме в базе теорем - "Элементарная алгебра" - "Умножение" - "Дистрибутивность" - "Приведение подобных членов для степеней".

7. Используется прием "Общнорм" - "Применение дополнительного тождества для упрощения заменяемого терма" - "Попытка опрокинуть эквивалентность общей стандартизации путем сильного упрощения заменяемого терма". Путь к теореме в базе теорем - "Элементарная алгебра" - "Плюс" - "Сложение численных равенств" - "Уравнение $A + X = B$".

8. Используется прием "Нормализация" - "Обобщение теоремы" - "Обобщение упрощающего тождества путем ввода бесповторного дополнительного параметра операнда". Путь к теореме в базе теорем - "Элементарная алгебра" - "Степени" - "Дробь в основании степени" - "Деление на степень дроби".

9. Используется прием "Общнорм" - "Склейка двух теорем" - "Попытка склеить две эквивалентности общей стандартизации, отличающиеся заменяющими частями". Путь к теореме в базе теорем - "Элементарная алгебра" - "Неравенства" - "Меньше" - "Положительность степени" - "Обобщения".

10. Используется прием "Упрощение" - "Использование дополнительного тождества для упрощения заменяемой части" - "Попытка навесить внешнюю операцию для устранения повторного вхождения переменной". Путь к теореме в базе теорем - "Элементарная алгебра" - "Тригонометрия" - "Синус" - "Синус суммы" - "Сумма тангенсов".

11. Используется прием "Опр" - "Использование тождества для упрощения заменяемой части дополнительной теоремы" - "Использование дополнительной теоремы, связанной с самым сложным понятием заменяющей части". Путь к тео-

реме в базе теорем - "Элементарная алгебра" - "Тригонометрия" - "Арксинус" - "Определение арксинуса" - "Простейшее уравнение с арксинусом". Фактически торема, которую требуется вывести в данном упражнении, является лишь промежуточным звеном цепочки вывода теоремы указанного раздела базы теорем.

12. Используется прием "Равны" - "Использование текущего тождества для варьирования дополнительной теоремы" - "Реализация антецедента дополнительной теоремы, выводящей равенство двух числовых атомов". Путь к теореме в базе теорем - "Элементарная геометрия" - "Биссектриса" - "Определение биссектрисы" - "Равноудаленность от сторон угла точки, лежащей на биссектрисе".

13. Используется прием "Числовойатом" - "Использование задачи на исследование для вывода следствий теоремы" - "Попытка варьирования теоремы путем дополнительного построения, позволяющего исключить старую переменную". Путь к теореме в базе теорем - "Элементарная геометрия" - "Фигуры" - "Треугольник" - "Тригонометрические соотношения в прямоугольном треугольнике" - "Синус" - "Теорема синусов". Заметим, что для копии теоремы синусов создана также отдельная ячейка вывода, в которой эта копия является стартовой теоремой.

14. Используется прием "Числовойатом" - "Использование дополнительной теоремы для реализации сложного антецедента" - "Попытка сведения более чем двуместного отношения в антецедентах к более простым отношениям". Путь к теореме в базе теорем - "Элементарная геометрия" - "Фигуры" - "Окружность на плоскости" - "Вписанные углы, опирающиеся на общую хорду" - "Две секущие, проведенные из общей точки". Заметим, что в этом разделе хранится не сама выводимая в упражнении теорема, а результат дальнейшей ее обработки - двукратного отбрасывания избыточных отрицания равенств в антецедентах.

15. Используется прием "Определение" - "Использование задачи на описание для получения новой теоремы" - "Попытка рассмотреть операцию, значением которой служит атомарный объект, координаты которого определяются - с подстановкой определяющего операцию терма в результирующее тождество". Путь к теореме в базе теорем - "Аналитическая геометрия" - "Координаты" - "Координаты на плоскости" - "Координаты векторов" - "Определение координат вектора" - "Определение координат суммы векторов".

16. Используется прием "Уравнкрив" - "Использование дополнительной теоремы" - "Использование дополнительной теоремы, выражающей множество точек через отдельные точки". Путь к теореме в базе теорем - "Аналитическая геометрия" - "Уравнение прямой на плоскости" - "Общий вид уравнения прямой на плоскости" - "Уравнение прямой, проходящей через две точки".

17. Используется прием "Уравнкрив" - "Использование дополнительной теоремы" - "Использование определения отношения в задании класса для получения его параметрического описания". Путь к теореме в базе теорем - "Аналитическая геометрия" - "Уравнение прямой в пространстве" - "Общий вид канонического уравнения прямой в пространстве" - "Переход от параметрического уравнения прямой к каноническому".

18. Используется прием "Систкоорд" - "Использование задачи на исследование для вывода следствий теоремы" - "Попытка анализа контекста теоремы для получения соотношения, не содержащего точки, координаты которой определяются теоремой". Путь к теореме в базе теорем - "Аналитическая геометрия" - "Уравнение прямой в пространстве" - "Условие перпендикулярности двух прямых" - "Расстояние от точки до прямой".

19. Используется прием "Уравнмножество" - "Логические следствия теоремы" - "Ввод вспомогательных параметров для числовых атомов, встречающихся в уравнении множества объектов". Путь к теореме в базе теорем - "Аналитическая геометрия" - "Линии второго порядка" - "Окружность" - "Уравнение окружности на плоскости" - "Уравнение окружности с заданным центром".

20. Используется прием "Определение" - "Реализация определяющей части" - "Примерка кванторного определения свойства последовательности на константную и тождественную последовательности". Путь к теореме в базе теорем - "Математический анализ" - "Пределы функций одной переменной" - "Предел последовательности" - "Определение предела последовательности" - "Подбор константной последовательности, имеющей заданный предел".

21. Используется прием "Определение" - "Реализация определяющей части" - "Примерка использующего описатели определения операции над функцией на элементарные операции раздела". Путь к теореме в базе теорем - "Математический анализ" - "Дифференцируемость и вычисление производных" - "Производные функции одной переменной" - "Определение производной" - "Производная суммы".

22. Используется прием "Функперех" - "Использование дополнительной теоремы для усиления основной" - "Усмотрение явное выразимости параметров друг через друга". Путь к теореме в базе теорем - "Математический анализ" - "Интегрирование" - "Первообразные" - "Определение первообразной" - "Интегрирование по частям".

23. Используется прием "Определение" - "Реализация определяющей части" - "Примерка кванторного определения свойства функции на элементарные операции раздела". Путь к теореме в базе теорем - "Математический анализ" - "Монотонность" - "Возрастание" - "Определение возрастания числовой функции на множестве" - "Дробь".

24. Первый вывод использует прием "Описатель" - "Вывод следствий с помощью задач на описание" - "Попытка замены варьируемой переменной при вычислении операции над конечным семейством". Второй вывод использует прием "Описатель" - "Обобщение теоремы" - "Попытка явного разрешения тождества относительно единственной численной операции над семейством". Путь к итоговой теореме в базе теорем - "Линейная алгебра" - "Определители" - "Определение определителя" - "Перестановка столбцов".

25. Путь к итоговой теореме в базе теорем - "Аналитическая геометрия" - "Линии второго порядка" - "Окружность"/ - "Уравнение окружности на плоскости"/ - "Выражение радиуса окружности через коэффициенты уравнения". Выбирается та теорема списка, которая совпадает с указанной в упражнении, нажимается

"д", и при помощи клавиш курсора просматривается вся цепочка вывода. Переход к следующей теореме (из нее выводится текущая) - "курсор вправо". Для запуска повтора текущего перехода цепочки нажимается "ъ". Для возвращения в просмотр цепочки - "0" и "Enter".

26. Путь к итоговой теореме в базе теорем - "Элементарная алгебра" - "Логарифмы" - "Определение логарифма" - "Изменение основания степени за счет преобразования логарифма в показателе степени".

27. Путь к итоговой теореме в базе теорем - "Элементарная геометрия" - "Фигуры" - "Окружность на плоскости" - "Центральные углы, опирающиеся на равные хорды" - "Две равные хорды, проведенные из общей точки".

28. Путь к итоговой теореме в базе теорем - "Математический анализ" - "Дифференцируемость и вычисление производных" - "Производные функции одной переменной" - "Производная сложной функции" - "Производная тангенса".

29. Путь к итоговой теореме в базе теорем - "Математический анализ" - "Интегрирование" - "Первообразные" - "Замена переменной интегрирования" - "Простейшие тригонометрические замены".

Глава 25

Примеры приемов, созданных генератором приемов

Описанная процедура генератора приемов была протестирована на различных разделах и позволила создать свыше 2000 новых приемов, которые были добавлены к основной базе приемов решателя. В действительности процедура создала гораздо большее количество приемов, но многие из них оказались ориентированы на крайне редкие ситуации, и при ручном отборе были проигнорированы, хотя и не сильно замедляли систему. Таким образом, пока аппарат генератора приемов должен рассматриваться лишь как средство полуавтоматического развития решателя, позволяющее выйти на новый уровень обучения сравнительно с ГЕНОЛОГом. Работа над его развитием продолжается.

Основным источником ценных новых приемов является логический вывод в базе теорем. Для такого вывода создается процедура, представляющая собой, по существу, отдельный решатель - решатель "верхнего уровня". Если прием "обычного" решателя объясняет, каким образом преобразовать текущую задачу, предположительно в направлении достижения ее ответа, то прием "теоремного" решателя объясняет, как можно было бы рассуждать, чтобы из рассматриваемой теоремы вывести полезные ее следствия. Фактически, предпринимаются попытки объяснить, каким образом эти новые теоремы могли бы быть "открыты" при анализе текущей теоремы. Подробнее о процедуре логического вывода в базе теорем, используемой системой, будет рассказано в девятом томе монографии. Предварительно, в восьмом томе, будут приведены общие сведения об организации базы теорем и создании спецификаций приемов по теоремам.

Здесь же ограничимся иллюстрацией работы генератора приемов, приведя серию созданных им в различных разделах новых приемов. Заметим, что приемы эти, в основном, представляют собой аналоги приемов, ранее созданных вручную, но не востребованные в тех задачах, по которым решатель обучался. Наибольшую ценность представляют самые простые из таких аналогов, так как вероятность столкнуться с задачей, где они могли бы сработать, достаточно высока. Впрочем, система логического вывода смогла предложить и несколько более сложных приемов, продемонстрировав потенциальные возможности данного подхода.

Приводимые ниже примеры выбраны из 2000 новых приемов достаточно случайным образом. Во многих разделах (линейная алгебра, математический анализ, теория вероятностей, общая алгебра) аппарат вывода теорем пока развит слабо. Здесь были

созданы лишь очень простые приемы, но даже они оказались полезными. Опыт показал, что наиболее частая причина отказов решателя - отсутствие именно таких простых приемов.

Чтобы получить более полный список автоматически созданных приемов, нужно зайти в корневое меню представляющего интерес оглавления базы приемов и нажать "a". Пролистывание приемов осуществляется клавишами "курсор вверх" "курсор вниз".

Приемы по алгебре множеств и комбинаторике

1. Параметрическое описание условия невключения.

   $\forall_{ab}(\neg(a \subseteq b) \leftrightarrow \exists_c(\neg(c \in b)\ \&\ c \in a))$

   Преобразуется содержащее неизвестные условие задачи на описание, не имеющей существенных неизвестных. Заметим, что ранее имелся аналогичный прием, применявшийся в задачах на поиск примера.

2. Упрощение дизъюнкции.

   $\forall_{ade}(d \subseteq e \rightarrow a \subseteq d\ \vee\ a \subseteq e \leftrightarrow a \subseteq e)$

3. Исключение несущественной неизвестной.

   $\forall_b(b - \text{set} \rightarrow \exists_a(a - \text{set}\ \&\ b \subseteq a))$

   Прием имеет заголовок "связка".

4. Проверка включения симметрической разности.

   $\forall_{ade}(a \subseteq d \cup e\ \&\ d \subseteq a \cup e \rightarrow a \triangle d \subseteq e)$

   Прием проверочного оператора "усмсодержится".

5. Включение прообраза объединения.

   $\forall_{adef}(\text{прообраз}(f, a) \subseteq a\ \&\ \text{прообраз}(f, d) \subseteq e \rightarrow \text{прообраз}(f, a \cup d) \subseteq e)$

   Прием проверочного оператора "усмсодержится".

6. Невключение образов при взаимно-однозначном отображении.

   $\forall_{abi}(\text{взаимнооднозначно}(i)\ \&\ \neg(a \subseteq b) \rightarrow= neg(\text{образ}(i, a) \subseteq \text{образ}(i, b)))$

   Прием проверочного оператора "усмнесодержится".

7. Сокращенная переформулировка двух не содержащих неизвестных условий непересечения в задаче на описание, имеющей цель "свертка".

   $\forall_{cdef}(\text{непересек}(c, e \cap (d \cup f)) \leftrightarrow \text{непересек}(c, d \cap e)\ \&\ \text{непересек}(e, c \cap f))$

8. Переход от параметрического задания класса к непосредственному.

   $\forall_{bd}(\text{set}_a(\exists_c(a = b \cup c\ \&\ c - \text{set})\ \&\ d(a)) = \text{set}_a(a - \text{set}\ \&\ b \subseteq a\ \&\ d(a)))$

9. Прием нормализатора "нормобъединение", склеивающий два примыкающих полуинтервала в интервал.

   $\forall_{bcf}(0 < c - b \ \& \ 0 < f - c \to (b, c] \cup [c, f) = (b, f))$

10. Декомпозиция включения.

    $\forall_{adef}(d \subseteq e \cup (f \setminus a) \leftrightarrow d \subseteq e \cup f \ \& \ a \cap d \subseteq e)$

11. Прием проверочного оператора "усмнепересек", усматривающий непересечение с прямым произведением.

    $\forall_{abcd}(\text{непересек}(b, a \times c) \ \& \ \text{непересек}(b, a \times d) \to \text{непересек}(b, a \times (c \cup d)))$

12. Прием проверочного оператора "усмненепересек", усматривающий пересечение прямых произведений.

    $\forall_{abcd}(\neg(\text{непересек}(a, c)) \ \& \ \neg(\text{непересек}(b, d)) \to \neg(\text{непересек}(a \times b, c \times d)))$

13. Прием проверочного оператора "усмнепересек", усматривающий непересечение с прообразом объединения.

    $\forall_{abcd}(\text{непересек}(b, \text{прообраз}(d, a)) \ \& \ \text{непересек}(b, \text{прообраз}(d, c)) \to \text{непересек}(b, \text{прообраз}(d, a \cup c)))$

14. Декомпозиция включения прямого произведения, имеющего своим сомножителем объединение.

    $\forall_{adef}((a \cup f) \times d \subseteq e \leftrightarrow a \times d \subseteq e \ \& \ f \times d \subseteq e)$

15. Кванторная расшифровка условия непересечения с объединением семейства множеств.

    $\forall_{be}(\text{непересек}(b, \bigcup(e)) \leftrightarrow \forall_{cf}(c \in b \ \& \ f \in \text{Dom}(e) \to \neg(c \in e(f))))$

16. Прием нормализатора "нормобласть", указывающий область определения тождественного отображения.

    $\forall_A(\text{Dom}(\text{тождфунк}(A)) = A)$

17. Декомпозиция включения образа объединения.

    $\forall_{adef}(\text{образ}(f, a \cup d) \subseteq e \leftrightarrow \text{образ}(f, a) \subseteq e \ \& \ \text{образ}(f, d) \subseteq e)$

18. Образ промежутка для элементарных функций.

    $\forall_{abcdef}(0 < -e + 1 \ \& \ 0 \le b - a \to \text{образ}(\lambda_g(\log_e g, f(g)), [a, b]) = [\log_e b, \log_e a])$

    $\forall_{abcdfg}(0 < a - 1 \ \& \ 0 < -g + 1 \ \& \ 0 \le b - a \to \text{образ}(\lambda_e(\log_e g, f(e)), [a, b]) = [\log_a g, \log_b g])$

    $\forall_{abcdf}(0 < a \ \& \ 0 < -b + \pi \ \& \ 0 \le b - a \to \text{образ}(\lambda_e(\cos e, f(e)), [a, b]) = [\cos b, \cos a])$

$\forall_{abcdfg}(0 < g \ \& \ 0 \leq b - a \to \text{образ}(\lambda_e(e^g, f(e)), [a, b]) = [a^g, b^g])$

Везде промежуток берется с концами произвольных типов. Приведена лишь малая часть автоматически созданных приемов данного раздела.

19. Сокращенная переформулировка двух условий в задаче на свертку.

    $\forall_{afx}(x \in \text{слой}(f, a) \leftrightarrow x \in \text{Dom}(f) \ \& \ a = f(x))$

    Замена выполняется справа налево.

20. Расшифровка условия принадлежности прообразу элемента.

    $\forall_{afx}(x \in \text{Dom}(f) \to x \in \text{слой}(f, a) \leftrightarrow a = f(x))$

    Прием выполняет общую стандартизацию утверждения и сопровождается стандартными для данного типа приемов фильтрами.

21. Декомпозиция условия включения прообраза объединения.

    $\forall_{adef}(\text{прообраз}(f, a \cup d) \subseteq e \leftrightarrow \text{прообраз}(f, a) \subseteq e \ \& \ \text{прообраз}(f, d) \subseteq e)$

22. Прообраз разности.

    $\forall_{abf}(\text{прообраз}(f, a \setminus b) = \text{прообраз}(f, a) \setminus \text{прообраз}(f, b))$

23. Расшифровка условия принадлежности интервалу.

    $\forall_{abc}(b - \text{число} \ \& \ c - \text{число} \to a \in (b, c) \leftrightarrow a - \text{число} \ \& \ 0 < a - b \ \& \ 0 < c - a)$

    Прием имеет тип "конъюнктивная декомпозиция элементарного утверждения" и сопровождается стандартными для данного типа фильтрами.

24. Склейка примыкающих интервала и промежутка.

    $\forall_{abce}(0 < b - e \ \& \ 0 \leq c - b \to [b, c] \cup (e, b) = (e, c])$

    Конец c может иметь произвольный тип; конец b принадлежит промежутку.

25. Склейка двух условий задачи на описание, явно разрешенных относительно неизвестной.

    $\forall_{bcd}(b - \text{число} \ \& \ c - \text{число} \to d \subseteq [b, c] \leftrightarrow \text{верхняягрань}(c, d) \ \& \ \text{нижняягрань}(b, d))$

    Замена выполняется справа налево. Выражения b, c не содержат неизвестных, переменная d является неизвестной.

26. Декомпозиция условия "наибольший" для последующей расшифровки условия принадлежности неизвестному множеству.

    $\forall_{ab}(\text{наибольший}(a, b) \leftrightarrow a \in b \ \& \ \text{верхняягрань}(a, b))$

    Преобразуется условие задачи на описание. Выражение b содержит неизвестные и не является переменной.

27. Исключение несущественной неизвестной

    $\forall_e(e - \text{число} \to \exists_a(a - \text{set} \ \& \ a \subseteq \mathbb{R} \ \& \ \text{наибольший}(e, a)))$

    Прием имеет заголовок "связка".

28. Ограниченность сверху сумм элементов ограниченных сверху множеств (прием проверочного оператора "усмогрсверху")

    $\forall_{bd}(\text{огрсверху}(b) \ \& \ \text{огрсверху}(d) \to \text{огрсверху}(\text{set}_x(\exists_{yz}(x = y + z \ \& \ y \in b \ \& \ z \in d))))$

    Антецеденты обрабатываются проверочными операторами.

29. Мощность множества элементов конечного отрезка целых чисел, делящихся на заданное натуральное число.

    $\forall_{acm}(m - \text{натуральное} \to \text{card}(\text{set}_n(n \in \{c, \ldots, a\} \ \& \ m|n)) = \max(1 + [-c/m] + [a/m], 0))$

30. Число множеств, заключенных "между" двумя множествами.

    $\forall_{bf}(\text{конечное}(f \setminus b) \to \text{card}(\text{set}_a(a - \text{set} \ \& \ a \subseteq f \ \& \ b \subseteq a)) = 2^{\text{card}(f \setminus b)}$ при $b \subseteq f,$ иначе $0)$

31. Мощность множества сумм с фиксированным слагаемым.

    $\forall_{ce}(e \subseteq \mathbb{R} \to \text{card}(e) = \text{card}(\text{set}_a(a + c \in e \ \& \ a - \text{число})))$

    Прием применяется справа налево.

32. Исключение несущественной неизвестной.

    $\forall_{cd}(\exists_a(a - \text{set} \ \& \ c \subseteq a \cup d \ \& \ \text{конечное}(a)) \leftrightarrow \text{конечное}(c \setminus d))$

    Прием имеет заголовок "связка".

33. Усмотрение бесконечного надмножества бесконечного множества (прием проверочного оператора "усмнеконечное")

    $\forall_{ab}(b \subseteq a \ \& \ \neg(\text{конечное}(b)) \to \neg(\text{конечное}(a)))$

Приемы по элементарной алгебре

1. Усмотрение ненулевого значения (прием проверочного оператора "усмне0")

   $\forall_{ac}(\neg(c = 0) \ \& \ \text{числитель}(c) - \text{even} \ \& \ 0 \leq a \ \& \ \neg(a - 1 = 0) \to \neg(-1 + a^c = 0))$

2. Общая стандартизация с исключением квантора.

   $\forall_{bf}(b - \text{число} \to \exists_c(b = cf \ \& \ c - \text{число}) \leftrightarrow \neg(b = 0) \ \& \ \neg(f = 0) \ \vee \ b = 0)$

3. Приемы нормализатора общей стандартизации "нормумножение" (приведена лишь часть автоматически созданных приемов)

   $\forall_{ac}(\sin a < 0 \ \& \ 0 \le \cos a \to (-\sin a)^c(-\operatorname{ctg} a)^c = (\cos a)^c)$

   $\forall_{bc}(\neg(\text{знаменатель}(c) - \text{even}) \ \& \ c - \text{rational} \to (-\operatorname{tg} b)^c(\cos b)^c = (-\sin b)^c)$

   $\forall_{ab}(\cos(a-b)\operatorname{tg}(b-a) = \sin(b-a))$

   $\forall_{cdef}(0 < de \to (d/e)^c(e/d)^f = (d/e)^{c-f})$

   $\forall_{cefgh}(\neg(\text{знаменатель}(e) - \text{even}) \ \& \ 0 < \log_f g \ \& \ 0 < log_f h \ \& \ e - \text{rational} \to (\log_f g)^c(\log_f h)^e(\log_h g)^e = (\log_f g)^{c+e})$

4. Приемы нормализатора разложения на множители.

   $\forall_{bc}(\sin b \operatorname{tg} c + \cos b = \cos(b-c)/\cos c)$

   $\forall_{bcd}(d \le 0 \to \min(-bd, -cd) = -d\min(b,c))$

   $\forall_{ae}(e^5 - a^5 - ae^4 = (a^2 + e^2 - ae)(e^3 - a^3 - ea^2))$

5. Сокращение дробных выражений в нормализаторе общей стандартизации "нормдробь".

   $\forall_{abcfgh}(\neg(\text{знаменатель}(h) - \text{even}) \ \& \ 0 < cg \ \& \ h - \text{rational} \to fc^hg^h/(a(cg)^b) = f(cg)^{h-b}/a)$

6. Сокращение степеней, основания которых отличаются знаком.

   $\forall_{acdefg}(\neg(\text{знаменатель}(d) - \text{even}) \ \& \ \neg(\text{числитель}(d) - \text{even}) \ \& \ 0 < f - g \ \& \ d - \text{rational} \to e(g-f)^d/(a(f-g)^c) = -e(f-g)^{d-c}/a)$

7. Группировка внутрь условного выражения.

   $\forall_{abd}(b - \text{число} \to (b \text{ при } a, \text{ иначе } 0)/d = (b/d \text{ при } a, \text{ иначе } 0))$

   Прием применяется слева направо.

8. Использование равенства из посылок для исключения неизвестных.

   $\forall_{cdefhij}(\neg(j = 0) \ \& \ \neg(\text{знаменатель}(h) - \text{even}) \ \& \ dj = fi \to ed^h/ci^h = ef^h/cj^h)$

   Последний антецедент идентифицируется с посылкой задачи на доказательство либо на иследование. Выражения d, i содержат неизвестные, а выражения f, j - не содержат.

9. Приемы нормализатора общей стандартизации "нормдробь" (приведена лишь часть автоматически созданных приемов).

   $\forall_{acdefghij}(0 < cgij \ \& \ 0 \le dhij \to e(ch/dg)^f/a = e(hi/dj)^f/(a(gi/cj)^f))$

   Замена выполняется справа налево.

$\forall_{abcf}(f\sin(2b/c)/(a\sin(b/c)\cos(b/c)) = 2f/a)$

$\forall_{abcf}(\neg(\text{знаменатель}(b)-\text{even})\ \&\ b-\text{rational} \to f\cdot(\sin c)^b/(a(\operatorname{tg} c)^b) = f\cdot(\cos c)^b/a)$

$\forall_{acdefg}(\neg(\text{знаменатель}(c)-\text{even})\ \&\ \neg(\text{знаменатель}(d)-\text{even})\ \&\ c-\text{rational}\ \&$
$d-\text{rational} \to e|f|^c/(a|g|^d) = e|f^c/g^d|/a)$

10. Преобразование степени произведения в произведение степеней с изменением знаков оснований степени.

    $\forall_{cde}(d \le 0\ \&\ e \le 0 \to (-d)^c(-e)^c = (de)^c)$

    Прием применяется справа налево.

11. Отбрасывание знаков основания в произведении степеней.

    $\forall_{cde}(\neg(\text{знаменатель}(c)-\text{even})\ \&\ c-\text{rational} \to (-d)^c(-e)^c = d^c e^c)$

12. Произведение степеней с основаниями, отличающимися знаками.

    $\forall_{ace}(\text{числитель}(e)-\text{even}\ \&\ e-\text{rational}\ \&\ a \le 0 \to a^e(-a)^c = (-a)^{c+e})$

13. Произведение степеней с взаимно обратными основаниями.

    $\forall_{cdef}(0 < de \to (d/e)^c(e/d)^f = (d/e)^{c-f})$

14. Общая стандартизация группы посылок.

    $\forall_{bce}(\neg(b=0)\ \&\ \text{числитель}(b)-\text{even}\ \&\ b-\text{rational} \to -e^b + c^b = 0\ \&\ ce \le 0 \leftrightarrow$
    $c = -e)$

15. Разрешение равенства относительно переменной, связанной внешним квантором.

    $\forall_{abd}(\neg(\text{знаменатель}(b)-\text{even})\ \&\ \neg(\text{числитель}(b)-\text{even})\ \&\ d-\text{число}\ \&$
    $b-\text{rational} \to d = a^b \leftrightarrow a = d^{1/b})$

    Замена выполняется слева направо. Переменная a связана внешним квантором, причем преобразуемое вхождение не является консеквентом кванторной импликации. Переменная a не встречается в выражениях b, d.

16. Исключение несущественной неизвестной.

    $\forall_{bc}(c-\text{число} \to \exists_a(c = a^b\ \&\ 0 < a\ \&\ a-\text{число}) \leftrightarrow \neg(b=0)\ \&\ \neg(c-1=0)\ \&$
    $0 < c\ \vee\ c = 1)$

    Прием имеет заголовок "связка".

17. Решение степенного неравенства с отрицательным показателем степени.

$\forall_{abcd}(\neg(a=0)\ \&\ \neg(\text{знаменатель}(b)-\text{even})\ \&\ b<0\ \&\ b-\text{rational}\ \&\ d=c^{1/b}\rightarrow c<a^b\leftrightarrow\neg(\text{числитель}(b)-\text{even})\ \&\ ((a<d\ \&\ 0<c\ \vee\ c\leq 0)\ \&\ 0<a\ \vee\ a<d\ \&\ c<0)\ \vee\ (a<d\ \&\ -d<a\ \&\ 0<c\ \vee\ c\leq 0)\ \&\ \text{числитель}(b)-\text{even})$

Выражение a содержит неизвестные; выражения b, c - не содержат. Последний антецедент выделен указателем "идентификатор". Он присваивает переменной d результат упрощения выражения $c^{1/b}$.

18. Стандартизация степенного неравенства.

$\forall_{ac}(\neg(\text{знаменатель}(c)-\text{even})\ \&\ \neg(\text{числитель}(c)-\text{even})\ \&\ 0<c\ \&\ c-\text{rational}\rightarrow 0<-a^c+1\leftrightarrow 0<-a+1)$

19. Сумма двойных радикалов.

$\forall_{ade}(d\leq 0\ \&\ 0\leq e\rightarrow a\sqrt{2\sqrt{e+d^2}-2\sqrt{e}}=-a\sqrt{d+\sqrt{e+d^2}}+a\sqrt{-d+\sqrt{e+d^2}})$

Прием применяется справа налево.

20. Переход к дизъюнкции в посылке задачи на доказательство или задачи на исследование, имеющей цель "противоречие".

$\forall_{bc}(0\leq b\ \&\ 0\leq c\rightarrow\neg(b+c=0)\leftrightarrow\neg(c=0)\ \vee\ \neg(b=0))$

21. Общая стандартизация с исключением квантора.

$\forall_{bd}(0<b\rightarrow\exists_a(a^b<d\ \&\ a-\text{число}\ \&\ 0\leq a)\leftrightarrow 0<d)$

22. Явное разрешение утверждения относительно переменной, связанной внешним описателем.

$\forall_{bcd}(0<b\ \&\ 0<d\rightarrow b\leq c/d\leftrightarrow d\leq c/b)$

Замена выполняется слева направо. Преобразуемое утверждение расположено под описателем, в связывающую приставку которого входит переменная d. Выражения b, c не имеют переменных, связанных внешними кванторами и описателями.

23. Прием проверочного оператора.

$\forall_{axy}(0<a-1\ \&\ 0\leq y-x\rightarrow\log_a x-\log_a y\leq 0)$

24. Вынесение степени за знак модуля.

$\forall_{cde}(\neg(\text{знаменатель}(e)-\text{even})\ \&\ e-\text{rational}\rightarrow|c^e/d^e|=|c/d|^e)$

25. Сокращение дроби с модулем.

$\forall_{acde}(\neg(\text{знаменатель}(e) - \text{even}) \ \& \ 0 < ac^e \ \& \ e - \text{rational} \to d|c|^e/(ac^e) = d/|a|)$

$\forall_{acde}(\neg(\text{знаменатель}(e) - \text{even}) \ \& \ ac^e < 0 \ \& \ e - \text{rational} \to d|c|^e/(ac^e) = -d/|a|)$

26. Общая стандартизация с исключением квантора.

$\forall_b(\exists_a(a - \text{число} \ \& \ |a| \le b) \leftrightarrow 0 \le b)$

27. Группировка дизъюнктивных членов относительно неизвестных.

$\forall_{ab}(0 < b \ \& \ b - \text{число} \to a = b \ \vee \ b = -a \leftrightarrow b = |a|)$

Прием применяется к дизъюнктивному условию задачи на описание. Выражение b содержит неизвестные, a - не содержит.

28. Свертка группы явно разрешенных относительно неизвестных условий в одно, тоже явно разрешенное.

$\forall_{ab}((a = b \ \vee \ b = -a) \ \& \ b \le 0 \leftrightarrow b = -|a|)$

Прием применяется к двум условиям задачи на описание. Переменная b - неизвестная; выражение a не содержит неизвестных.

29. Исключение сигнума.

$\forall_a(-1 + \operatorname{sg}(a) = 0 \leftrightarrow 0 < a)$

30. Декомпозиция сложной операции, использующая посылку для идентификации подмножества операндов ассоциаивно-коммутативной операции.

$\forall_{ade}(e < 0 \to \log_a(de) = \log_a(-d) + \log_a(-e))$

Антецедент идентифицируется с посылкой, указывающей знак подпроизведения e произведения de, раположенного под логарифмом.

31. Исключение логарифма.

$\forall_{cde}(\text{числитель}(e) - \text{even} \ \& \ e - \text{rational} \to \log_{c^e d^e} |cd| = 1/e)$

$\forall_{cde}(\neg(\text{знаменатель}(e) - \text{even}) \ \& \ \neg(\text{числитель}(e) - \text{even}) \ \& \ e - \text{rational} \to e = \log_{-cd}(-c^e d^e))$

$\forall_{abc}(a = -a \log_{b/c}(c) + a \log_{b/c}(b))$

32. Свертка суммы двух логарифмов с вынесением наружу показателя степени.

$\forall_{defg}(\neg(\text{знаменатель}(g) - \text{even}) \ \& \ 0 < de \ \& \ g - \text{rational} \to g \log_f(de) = \log_f(d^g) + \log_f(e^g))$

Прием применяется справа налево.

33. Упрощение выражения под описателем относительно варьируемой переменной.

    $\forall_{abcd}(d\log_a b + d\log_a c = d\log_a(bc))$

    Сумма логарифмов расположена под описателем, в связывающую приставку которого входит переменная x1, причем выражения b, c не содержат переменных этой приставки.

34. Сокращение в сумме двух логарифов.

    $\forall_{abcde}(b\log_a(d/(ce)) + b\log_a|e| = b\log_a|d/c|)$

35. Формула приведения.

    $\forall_{abc}(c|a \;\&\; a - \text{целое} \;\&\; c - \text{целое} \to \sin(b + a\pi/c) = (-1)^{(}a/c)\sin b)$

36. Исключения модуля под синусом содержащим неизвестные.

    $\forall_{abc}(\sin(c|a|/b) = \sin(ac/b)\cdot \mathrm{sg}(a))$

37. Свертка константного выражения.

    $\forall_{ab}(a\sqrt{3}\sin b + a\cos b = 2a\sin(b + \pi/6))$

    Прием применяется слева направо. Оба слагаемые константные.

38. Сокращение синусов с модулем.

    $\forall_{abcfgh}(\neg(\text{знаменатель}(c) - \text{even}) \;\&\; \neg(\text{знаменатель}(g) - \text{even}) \;\&\; c - \text{rational} \;\&\; g - \text{rational} \to f\cdot(\mathrm{sg}(b^c))^g/a = f\cdot\sin(h|b|^c)^g/(a(\sin(hb^c))^g))$

    Прием применяется справа налево.

39. Вынесение сигнума из-под синуса.

    $\forall_{axy}(\sin(x\mathrm{sg}(y)/a) = \sin(x/a)\mathrm{sg}(y))$

40. Общая стандартизация

    $\forall_{ac}(2a\sin(\pi/6 - c) + a\sqrt{3}\sin c = a\cos c)$

41. Исключение тангенса.

    $\forall_{bcf}(c\sin b + c\cos b\,\mathrm{tg}\, f = c\sin(b + f)/\cos f)$

42. Свертка конъюнкции.

    $\forall_c(\neg(\sin(2c) = 0) \leftrightarrow \neg(\sin c = 0) \;\&\; \neg(\cos c = 0))$

    Прием применяется справа налево. Заметим, что речь идет о преобразовании именно конъюнкции, а не двух посылок либо условий.

43. Усмотрение неравенства с помощью проверочных операторов.

$\forall_d(\sin d < 0 \ \&\ \cos d < 0 \to 0 < \sin(2d))$

Антецеденты обрабатываются проверочными операторами. Прием имеет заголовок "второйтерм", т.е. заменяет неравенство на константу "истина".

44. Общая стандартизация группы посылок.

$\forall_c(\neg(\sin c = 0) \ \&\ \sin(2c) = 0 \leftrightarrow \cos c = 0)$

45. Отбрасывание дизъюнктивного члена.

$\forall_c(\sin c = 0 \ \vee\ \sin(2c) = 0 \leftrightarrow \sin(2c) = 0)$

46. Шаг сведения условия задачи на описание к кратным вхождениям единственного неизвестного подтерма.

$\forall_{ab}(a\sin b + a\cos b = a\sqrt{2}\sin(b + \pi/4))$

Прием применяется к условию задачи на описание. Выражение b содержит неизвестные, причем если имеются вхождения неизвестных в это условие, не расположенные в двух заменяемых слагаемых, то все они заключены внутри вхождений выражения $\sin(b + \pi/4)$.

47. Прием нормализатора "нормкосинус".

$\forall_a(\cos(\arccos(a)) = a)$

48. Общая стандартизация.

$\forall_{abc}(0 < \cos b \ \&\ 0 \le a - c \ \&\ \sin b \le 0 \to (-\operatorname{tg} b)^c(\cos b)^a = (-\sin b)^c(\cos b)^{a-c})$

49. Сложение тангенсов.

$\forall_{aef}(a\operatorname{tg} e + a\operatorname{tg} f = a\sin(e + f)/(\cos e\cos f))$

50. Сокращенная переформулировка при завершающем редактировании.

$\forall_{abcfg}(\neg(\text{знаменатель}(b) - \text{even}) \ \&\ b - \text{rational} \to f \cdot (\operatorname{ctg}(c + g))^b/a = f \cdot (-\operatorname{tg} c\operatorname{tg} g + 1)/(a(\operatorname{tg} c + \operatorname{tg} g)^b))$

Прием применяется справа налево на этапе редактирования ответа задачи на описание либо на преобразование.

51. Исключение квадрата тангенса.

$\forall_{ad}(d - d(\operatorname{tg} a)^2 = d\cos(2a)/(\cos a)^2)$

52. Вынесение степени минус единицы из-под тангенса.

$\forall_{abc}(b - \text{целое} \to \operatorname{tg} a(-1)^b/c = (-1)^b\operatorname{tg}(a/c))$

53. Занесение минуса под тангенс разности при завершающем редактировании ответа.

$\forall_{bcde}(c\,\mathrm{tg}(e-d)/b = -c\,\mathrm{tg}(d-e)/b)$

54. Арксинус косинуса.

$\forall_b(b \le 0\ \&\ 0 \le b+\pi \to \arcsin(\cos(b)) = b+\pi/2)$

55. Разрешение равенства с арккосинусом относительно неконстантного выражения.

$\forall_{ab}(\arccos(a) = b \leftrightarrow b - \text{число}\ \&\ a = \cos b\ \&\ 0 \le b\ \&\ 0 \le \pi - b)$

Выражение a неконстантное, а b - константное.

56. Равенство нулю разности арктангенсов.

$\forall_{ac}(-\operatorname{arctg} c + \operatorname{arctg} a = 0 \leftrightarrow a - c = 0)$

57. Котангенс арктангенса.

$\forall_b(\mathrm{ctg}(\operatorname{arctg} b) = 1/b)$

58. Объединение двух условий задачи на описание.

$\forall_{mn}(m - \text{целое}\ \&\ n - \text{целое} \to \neg(m = n)\ \&\ n - 1 < m \leftrightarrow n + 1 \le m)$

$\forall_{ab}(a - \text{целое}\ \&\ b - \text{целое} \to b - 1 < a\ \&\ a \le b \leftrightarrow a = b)$

59. Прием проверочного оператора "усмнецелое".

$\forall_{mn}(m - \text{целое}\ \&\ \neg(n - \text{целое}) \to \neg(m + n - \text{целое}))$

60. Исключение несущественной неизвестной.

$\forall_c(\exists_n(n - \text{натуральное}\ \&\ n \le c\ \&\ n - \text{число}) \leftrightarrow 1 \le c)$

Прием имеет заголовок "связка".

61. Прием проверочного оператора "усмнатуральное".

$\forall_a(a - \text{натуральное} \to [a] - \text{натуральное})$

62. Общая стандартизация утверждения.

$\forall_{abc}(\neg(c = 0)\ \&\ c - \text{rational} \to bc/a - \text{rational} \leftrightarrow b/a - \text{rational})$

$\forall_b(|b| - \text{rational} \leftrightarrow b - \text{rational})$

Замена выполняется слева направо.

63. Прием проверочного оператора "усмрациональное".

$\forall_{bc}(b - \text{rational} \ \& \ c - \text{rational} \leftrightarrow \min(b, c) - \text{rational})$

64. Прием проверочного оператора "усмнерациональное".

$\forall_{bc}(c - \text{rational} \ \& \ \neg(b - \text{rational}) \rightarrow \neg(b/c - \text{rational}))$

65. Делимость на модуль.

$\forall_{ab}(|b| \mid a \leftrightarrow b \mid a)$

66. Делимость на частное.

$\forall_{bde}(e - \text{целое} \rightarrow (d/e) \mid b \leftrightarrow d \mid (be))$

Прием выполняет общую стандартизацию; замена происходит слева направо.

67. Кванторная свертка.

$\forall_{mn}(\neg(n = 0) \ \& \ m - \text{целое} \ \& \ n - \text{целое} \rightarrow m \mid n \leftrightarrow \exists_k(\neg(k = 0) \ \& \ n/k = m \ \& \ k - \text{целое}))$

Замена выполняется справа налево. Прием сопровождается обычными для приемов кванторной свертки фильтрами.

68. Переход от параметрического задания класса к непосредственному.

$\forall_{am}(m - \text{целое} \rightarrow \text{set}_n(\exists_k(n = km \ \& \ k - \text{целое}) \ \& \ a(n)) = \text{set}_n(n - \text{целое} \ \& \ m \mid n \ \& \ a(n)))$

69. Приемы проверочного оператора "усмделит".

$\forall_{km}(k - \text{целое} \rightarrow m \mid km)$

$\forall_{ade}(ae \mid d \rightarrow a \mid d/e)$

70. Прием проверочного оператора "усмнеделит".

$\forall_{ab}(\neg(b \mid a) \rightarrow \neg(b \mid -a))$

71. Усмотрение нулевого вычета.

$\forall_{abc}(a - \text{натуральное} \ \& \ b - \text{натуральное} \ \& \ b \mid c \ \& \ c - \text{целое} \rightarrow (ac)(\text{mod } ab) = 0)$

72. Кванторная свертка.

$\forall_{bmn}(n - \text{натуральное} \ \& \ b - \text{целое} \ \& \ m - \text{целое} \rightarrow b(\text{mod } n) = m(\text{mod } n) \rightarrow \exists_a(b = m + an \ \& \ a - \text{целое}))$

Замена выполняется справа налево. Прием сопровождается обычными для приемов кванторной свертки фильтрами.

73. Сокращенная переформулировка группы условий в задаче на свертку.

    $\forall_{abmn}(a(\mathrm{mod}\ \mathrm{нок}(b,n)) = 0 \leftrightarrow a(\mathrm{mod}\ b) = 0\ \&\ a(\mathrm{mod}\ n) = 0)$

    Замена выполняется справа налево.

74. Попытка использования параметрического описания при получении частичного ответа.

    $\forall_{kmn}(k \in \{0,\ldots,n-1\} \to k = m(\mathrm{mod}\ n) \leftrightarrow \exists_p(m = k + np\ \&\ p - \mathrm{целое}))$

    Прием применяется к условию задачи на описание, имеющей цель "пример" либо цель "параметризация". Переменная m - неизвестная, не входящая в k, n.

75. Прием нормализатора "нормвычет".

    $\forall_a(\neg(a - \mathrm{even}) \to a(\mathrm{mod}\ 2) = 1)$

76. Сумма гиперболических косинусов.

    $\forall_{ce}(0 \le e-c+1 \to \sum_{f=c}^{e} \mathrm{ch}(f) = (-\mathrm{ch}\, c - \mathrm{ch}\, e + \mathrm{ch}(c-1) + \mathrm{ch}(e+1))e/(-e+1)^2)$

Приемы по элементарной геометрии

1. Прием проверочного оператора "усмточка".

   $\forall_{aAB}(a \in \mathrm{луч}(AB) \to a - \mathrm{точка})$

2. Усмотрение различия диагоналей квадрата проверочным оператором "разныепрямые".

   $\forall_{ABCD}(\mathrm{квадрат}(ABCD) \to \mathrm{разныепрямые}(\mathrm{прямая}(AC), \mathrm{прямая}(BD)))$

3. Усмотрение равенства длин пересекающихся отрезков из равенства длин их разностей.

   $\forall_{BCEF}(\mathrm{актив}(l(BC))\ \&\ \mathrm{актив}(l(EF))\ \&\ B \in \mathrm{отрезок}(EF)\ \&\ E \in \mathrm{отрезок}(BC) \to l(BC) = l(EF))$

4. Исключение описателя "класс".

   $\forall_{AB}(\mathrm{отрезок}(AB) = \mathrm{set}_C(C - \mathrm{точка}\ \&\ \mathrm{точкалуча}(A,B,C)\ \&\ l(AC) \le l(AB)))$

5. Из точки на биссектрисе проведены под равными углами отрезки до пересечения со сторонами угла. Тогда расстояния от вершины угла до точек пересечения равны.

   $\forall_{abcd}(\mathrm{разныепрямые}(\mathrm{прямая}(ad), \mathrm{прямая}(ab))\ \&\ \angle(adb) = \angle(adc)\ \&\ \mathrm{биссектриса}(bacd) \to l(ab) = l(ac))$

   Прием имеет заголовок "вывод". Расстояния $l(ab)$, $l(ac)$ уже рассматриваются в задаче.

6. Если две точки находятся по разные стороны от прямой, а третья - по одну сторону с одной из данных двух точек, то она же находится по разные стороны с другой из этих точек.

   $\forall_{ABCDE}$(разныестороны$(C, D,$ прямая$(AB))$ & однасторона$(C, E,$ прямая$(AB))$ & разныепрямые(прямая(CA), прямая$(AB))$ & разныеточки$(C, A) \to$ разныестороны$(E, D,$ прямая$(AB)))$

7. Соотношения пропорциональности для отрезков, отсекаемых параллельными прямыми.

   $\forall_{ABCDEab}(B \in$ прямая(AD) & прямая$(DE) \parallel$ прямая(BC) & $C \in$ прямая(AE) & разныепрямые(прямая(AE), прямая$(BC))$ & $al(CE) = bl(BD) \to$ $al(AC) = bl(AB))$

   Прием имеет заголовок "вывод". Последний антецедент, устанавливающий пропорциональность двух расстояний с известными коэффициентами a, b, выделен указателем "равно".

   $\forall_{ABCEab}$(актив$(l(aA))$ & актив$(l(bE))$ & актив$(l(BC))$ & актив$(l(ab))$ & актив$(l(AB))$ & разныепрямые(прямая(aA), прямая$(AE))$ & $B \in$ прямая(aA) & $C \in$ прямая(AE) & прямая$(ab) \parallel$ прямая(BC) & прямая$(aA) \parallel$ прямая(bE) & разныестороны$(a, b,$ прямая$(AE)) \to (l(aA) + l(bE))l(BC) = l(ab)l(AB))$

   Прием имеет заголовок "вывод". Одно из участвующих в соотношении расстояний имеет тип "неизв", остальные - известны.

8. Усмотрение параллельности прямых из соотношений пропорциональности.

   $\forall_{ABCDE}$(разныеточки(D, E) & разныеточки(B, C) & $A \in$ отрезок(CE) & $A \in$ отрезок(BD) & $l(AC)l(AD) = l(AB)l(AE)$ & актив(прямая$(DE)) \to$ прямая$(BC) \parallel$ прямая$(DE))$

   $\forall_{BCDEab}$(разныеточки(b, B) & разныеточки(a, D) & $l(aD)l(BC) = l(bB)l(DE)$ & $C \in$ прямая(bB) & $E \in$ прямая(aD) & прямая$(ab) \parallel$ прямая(BD) & однасторона$(B, D,$ прямая$(CE))$ & однасторона$(C, E,$ прямая$(BD)) \to$ прямая$(ab) \parallel$ прямая$(CE))$

   Приемы имеют заголовок "вывод".

9. Усмотрение перпендикулярности прямых из равенства для разности углов.

   $\forall_{ABCD}(-\angle(ABC)+\angle(BCD) = \pi/2$ & $C \in$ отрезок(AD) & актив(прямая$(AD)) \to$ прямая$(AB) \perp$ прямая$(AD))$

   Прием имеет заголовок "вывод". Первый и третий антецеденты идентифицируются с посылками.

10. Варианты применения теоремы Менелая.

    $\forall_{ABCDEFabcd}$(разныеточки(B,D) & разныепрямые(прямая(AC), прямая(BC)) & $al(AF) = bl(DF)$ & $cl(BD) = dl(CD)$ & $D \in$ отрезок(BC) & $E \in$ отрезок(AC) & $F \in$ прямая(AD) & $F \in$ прямая$(BE) \to bdl(CE) = (ac + ad)l(AE)$)

    Прием имеет заголовок "вывод". Расстояния AF, DF, BD, CD, а также хотя бы одно из расстояний CE, AE уже рассматриваются в задаче. Известные коэффициенты пропорциональности a, b, c, d определяются при помощи синтезаторов.

    $\forall_{ABCDEFcdef}$(разныепрямые(прямая(AC), прямая(BC)) & $cl(CE) = dl(AE)$ & $el(BF) = fl(EF)$ & $D \in$ отрезок(BC) & $E \in$ отрезок(AC) & $F \in$ прямая(AD) & $F \in$ прямая$(BE) \to (ce + cf + de)l(DF) = dfl(AF)$)

    Прием имеет заголовок "вывод". Расстояния CE, AE, BF, EF, а также хотя бы одно из расстояний DF, AF уже рассматриваются в задаче. Известные коэффициенты пропорциональности c, d, e, f определяются при помощи синтезаторов.

11. Переход от неравенства для углов к неравенству для расстояний.

    $\forall_{ABC}(0 \le -l(AB) + l(BC) \leftrightarrow 0 \le -\angle(ACB) + \angle(BAC))$

12. Соотношения, выводимые при помощи теоремы синусов.

    $\forall_{ABCa}$(актив$(\angle(BAC))$ & актив$(l(AC))$ & актив$(\angle(BaC))$ & актив$(l(aC))$ & $\angle(aBC) = \angle(ABC) \to \sin(\angle(BAC))l(AC) = \sin(\angle(BaC))l(aC)$)

    $\forall_{defg}$(актив$(\angle(egf))$ & актив$(l(de))$ & актив$(l(fg))$ & актив$(\angle(dge))$ & актив$(l(dg))$ & актив$(l(ef))$ & $e \in$ отрезок$(df) \to \sin(\angle(egf))l(de)l(fg) = \sin(\angle(dge))l(dg)l(ef)$)

    $\forall_{defg}$(актив$(\angle(efg))$ & актив$(l(fg))$ & актив$(\angle(edg))$ & актив$(l(dg))$ & $l(de) = l(ef)$ & прямая$(df) \parallel$ прямая$(eg) \to \sin(\angle(efg))l(fg) = \sin(\angle(edg))l(dg)$)

    $\forall_{defg}$(актив$(\angle(fdg))$ & актив$(l(dg))$ & актив$(\angle(feg))$ & актив$(l(ef))$ & разныепрямые(прямая(df), прямая(eg)) & прямая$(df) \parallel$ прямая(eg) & разныестороны$(d, e,$ прямая$(fg)) \to \sin(\angle(fdg))l(dg) = \sin(\angle(feg))l(ef)$)

    $\forall_{defg}$(актив$(\angle(dgf))$ & актив$(l(dg))$ & актив$(l(eg))$ & актив$(\angle(efg))$ & актив$(l(df))$ & актив$(l(ef))$ & разныепрямые(прямая(df), прямая(eg)) & прямая$(df) \parallel$ прямая(eg) & разныестороны$(d, e,$ прямая$(fg)) \to$ $\sin(\angle(dgf))l(dg)l(eg) = \sin(\angle(efg))l(df)l(ef)$)

    Приемы имеют заголовок "вывод". Используются, если один из числовых атомов не известен, а остальные известны.

13. Два треугольника с вертикальными углами.

    $\forall_{defgh}$(актив$(l(de))$ & актив$(l(dg))$ & актив$(l(df))$ & актив$(l(dh))$ & актив$(l(gh))$ & актив$(l(ef))$ & $d \in$ отрезок(eh) & $d \in$ отрезок$(fg) \to$ $(l(de)l(dg) - l(df)l(dh))(l(df)l(dg) - l(de)l(dh)) =$ $l(gh)^2l(de)l(df) - l(ef)^2l(dg)l(dh)$)

Прием имеет заголовок "вывод". Одно из участвующих в соотношении расстояний имеет тип "неизв", остальные - известны.

14. Усмотрение точки пересечения двух биссектрис отрезку третьей биссектрисы.

$\forall_{ABCDEFG}$(разныепрямые(прямая(AB), прямая(AC)) & $D \in$ прямая(AB) & $E \in$ прямая(BC) & $F \in$ прямая(AC) & $G \in$ прямая(BF) & $G \in$ прямая(CD) & биссектриса$(ABCF)$ & биссектриса$(ACBD)$ & биссектриса$(BACE) \rightarrow$ $G \in$ отрезок(AE))

Прием имеет заголовок "вывод".

15. Исключение несущественной неизвестной (усмотрение существования точки пересечения трех биссектрис).

$\forall_{ABCDEFG}(\triangle(ABC)$ & биссектриса$(BACE)$ & $E \in$ прямая(BC) & биссектриса$(ABCF)$ & $F \in$ прямая(AC) & биссектриса$(ACBD)$ & $D \in$ прямая$(AB) \rightarrow \exists_G(G -$ точка & $G \in$ отрезок(AE) & $G \in$ отрезок(BF) & $G \in$ отрезок$(CD)))$

Прием имеет заголовок "связка".

16. Соотношения для двух треугольников со смежной стороной, один из которых - прямоугольный.

$\forall_{defg}$(актив$(\angle(efg))$ & актив$(l(fg))$ & актив$(l(dg))$ & $l(de) = l(ef)$ & прямая$(de) \perp$ прямая(dg) & прямая$(df) \parallel$ прямая$(eg) \rightarrow$ $\sin(\angle(efg))l(fg) = l(dg))$

$\forall_{defg}$(актив$(\angle(dfg))$ & актив$(l(de))$ & актив$(l(fg))$ & актив$(l(dg))$ & актив$(l(eg))$ & $e \in$ отрезок(df) & прямая$(dg) \perp$ прямая$(eg) \rightarrow \sin(\angle(dfg))l(de)l(fg) = l(dg)l(eg))$

$\forall_{defgh}$(актив$(l(gh))$ & актив$(l(dg))$ & актив$(l(dh))$ & актив$(l(df))$ & актив$(l(de))$ & $d \in$ отрезок(eh) & $d \in$ отрезок(fg) & прямая$(ef) \perp$ прямая$(eh) \rightarrow (-l(gh)^2 + l(dg)^2 + l(dh)^2)l(df) = 2l(de)l(dg)l(dh))$

Приемы имеют заголовок "вывод".

17. Равные отрезки, отложенные на боковых сторонах равнобедренного треугольника от его основания.

$\forall_{abcCF}$(актив$(l(cC))$ & актив$(l(aF))$ & $l(ab) = l(bc)$ & $l(aC) = l(cF)$ & точкалуча(a, b, C) & точкалуча$(c, b, F) \rightarrow l(cC) = l(aF))$

Прием имеет заголовок "вывод".

18. Равенство двух прямоугольных треугольников по гипотенузе и острому углу.

$\forall_{abcABC}(l(bc) = l(BC)$ & $\angle(abc) = \angle(ABC)$ & прямая$(ab) \perp$ прямая(ac) & прямая$(AB) \perp$ прямая$(AC) \rightarrow l(AC) = l(ac))$

Прием имеет заголовок "вывод". Расстояния $l(AC), l(ac)$ уже рассматриваются в задаче.

19. Признаки равенства треугольников, имеющих общую сторону либо общий угол.

    Хотя имеются общие признаки равенства треугольников, но их идентификация трудоемка, так как априори неясно, какие именно треугольники нужно рассматривать. Если же треугольники имеют общую сторону либо общий угол, то идентификация упрощается, и для этих случаев были созданы отдельные приемы. Часть таких приемов создавалась вручную; генератор приемов существенно пополнил их запасы. Здесь мы приводим лишь несколько таких приемов.

    $\forall_{BDEF}(\text{актив}(\angle(DBF))\ \&\ \text{актив}(\angle(DEF))\ \&\ l(BD) = l(DE)\ \&\ \angle(BDF) = \angle(EDF) \to \angle(DBF) = \angle(DEF))$

    $\forall_{BCEF}(\text{актив}(l(BC))\ \&\ \text{актив}(l(EF))\ \&\ l(BF) = l(CE)\ \&\ \angle(BEC) = \angle(EBF) \to l(BC) = l(EF))$

    $\forall_{ACEF}(\text{актив}(l(CE))\ \&\ \text{актив}(l(EF))\ \&\ l(AC) = l(AF)\ \&\ \angle(CAE) = \angle(EAF) \to l(CE) = l(EF))$

    $\forall_{abcde}(\text{актив}(l(bd))\ \&\ \text{актив}(l(ce))\ \&\ l(ab) = l(ae)\ \&\ l(ae) = l(ad)\ \&$
    $\text{точкалуча}(a, b, c)\ \&\ \text{точкалуча}(a, d, e) \to l(bd) = l(ce))$

20. Высоты параллелограмма относятся как его стороны.

    $\forall_{ABCDEFabcd}(a\ in\text{прямая}(CD)\ \&\ b\ in\text{прямая}(AB)\ \&\ E\ in\text{прямая}(BC)\ \&$
    $F \in \text{прямая}(AD)\ \&\ \text{прямая}(ab) \perp \text{прямая}(AB)\ \&\ \text{прямая}(AD) \perp \text{прямая}(EF)\ \&$
    $\text{прямая}(AB) \parallel \text{прямая}(CD)\ \&\ \text{прямая}(AD) \parallel \text{прямая}(BC)\ \&\ cl(EF) = dl(ab) \to cl(AB) = dl(AD))$

    Прием имеет заголовок "вывод". Последний антецедент выделен указателем "равно".

21. Усмотрение ромба в параллелограмме с перпендикулярными диагоналями.

    $\forall_{ABCD}(\text{параллелограмм}(ABCD)\ \&\ \text{прямая}(AC) \perp \text{прямая}(BD) \to \text{ромб}(ABCD))$

22. Исключение несущественной неизвестной.

    $\forall_{ABCD}(\text{ромб}(ABCD) \to \exists_E(E - \text{точка}\ \&\ E \in \text{отрезок}(BD)\ \&\ E\ in\text{отрезок}(AC)))$

    Прием имеет заголовок "связка".

23. Усмотрение четырехугольника в прямоугольнике.

    $\forall_{ABCD}(\text{прямоугольник}(ABCD) \to \text{четырехугольник}(ABCD))$

    Прием имеет заголовок "второйтерм".

24. Усмотрение квадрата в прямоугольнике с равными сторонами.

    $\forall_{ABCD}(-l(BC) + l(AB) = 0\ \&\ \text{прямоугольник}(ABCD) \to \text{квадрат}(ABCD))$

    Прием имеет заголовок "вывод". Первый антецедент выделен указателем "идентификатор".

25. Середина диагонали квадрата принадлежит другой диагонали.

$\forall_{ABCDE}(-l(CE)+l(AE)=0$ & $E \in$ прямая(AC) & квадрат$(ABCD) \rightarrow$
$E \in$ отрезок$(BD))$

Прием имеет заголовок "вывод". Первый антецедент выделен указателем "идентификатор".

26. Расшифровка условия "трапеция$(ABCD)$" задачи на доказательство.

$\forall_{ABCD}($трапеция$(ABCD) \leftrightarrow$ четырехугольник$(ABCD)$ &
прямая$(BC) \parallel$ прямая(AD) & $\neg($прямая$(AB) \parallel$ прямая$(CD))$ &
$\angle(BAD) \leq \pi/2$ & $\angle(ADC) \leq \pi/2)$

27. Если хорды равны, то равны и углы между хордой и радиусом.

$\forall_{abcdef}(l(cd)=l(ef)$ & $c \in$ окружность(ab) & $d \in$ окружность(ab) &
$e \in$ окружность(ab) & $f \in$ окружность$(ab) \rightarrow \angle(acd)=\angle(aef))$

Прием имеет заголовок "вывод". Первые два антецедента идентифицируются непосредственно, остальные - выделены указателем "усм".

28. Усмотрение принадлежности окружности точки, возникающей при рассмотрении биссектрисы центрального угла.

$\forall_{cefgh}(\angle(cef)=\angle(ehf)$ & $e \in$ окружность(fg) & $h \in$ окружность(fg) &
биссектриса$(hfce) \rightarrow l(cf)=l(fg))$

29. Принадлежность хорде проекции точки дуги.

$\forall_{ABCDEF}($окружность(AB) описана около фигура(CDE) & прямая$(CD) \perp$
прямая(EF) & разныестороны$(E, A,$ прямая$(CD))$ & Fпрямая$(CD) \rightarrow$
$F \in$ отрезок$(CD))$

30. Прямоугольный треугольник, дополняющий описанный треугольник до трапеции.

$\forall_{defgDE}($актив$(l(dg))$ & актив$(l(DE))$ & актив$(l(ef))$ & актив$(l(fg))$ &
разныепрямые(прямая$(df),$ прямая$(eg))$ & прямая$(df) \perp$ прямая(dg) &
прямая$(df) \parallel$ прямая(eg) & окружность(DE) описана около фигура(efg) &
разныестороны$(d, e,$ прямая$(fg)) \rightarrow 2l(dg)l(DE)=l(ef)l(fg))$

$\forall_{defgDE}($актив$(l(dg))$ & актив$(l(DE))$ & актив$(l(df))$ & актив$(l(ef))$ &
разныепрямые(прямая$(df),$ прямая$(eg))$ & прямая$(dg) \perp$ прямая(fg) &
прямая$(df) \parallel$ прямая(eg) & окружность(DE) описана около фигура(efg) &
разныестороны$(d, e,$ прямая$(fg)) \rightarrow 2l(dg)l(DE)=l(df)l(ef))$

31. Усмотрение окружности, описанной около четырехугольника.

$\forall_{ABCDE}($разныеточки(A, E) & $l(AE)=l(DE)$ & $l(BE)=l(DE)$ &
$l(CE)=l(DE)$ & четырехугольник$(ABCD)$ & актив(окружность$(EA)) \rightarrow$
окружность(EA) описана около фигура$(ABCD))$

32. Расшифровка по определению проверяемого условия.

    $\forall_{ABCDEa}(A - $точка & $B - $точка & $C - $точка & $\triangle(ABC)$ &
    $a =$ окружность$(DE) \to$ окружность(DE) вписана в фигура$(ABC) \leftrightarrow$
    отрезок(AB) $-$ касательная к(a) & отрезок(BC) $-$ касательная к(a) &
    отрезок(AC) $-$ касательная к$(a))$

33. Проекция центра вписанной окружности лежит на стороне треугольника.

    $\forall_{ABCDEF}(F \in$ прямая(AC) & $B - $точка & прямая$(AC) \perp$ прямая(DF) &
    окружность(DE) вписана в фигура$(ABC) \to F \in$ отрезок$(AC))$

34. Два треугольника с примыкающими основаниями и боковыми сторонами.

    $\forall_{adefg}($актив$(\angle(dfg))$ & актив$(l(de))$ & актив$(l(fg))$ & актив$(\angle(dae))$ &
    актив$(l(ad))$ & актив$(l(eg))$ & $e \in$ отрезок(df) & $g \in$ отрезок$(ae) \to$
    $\sin(\angle(dfg))l(de)l(fg) = \sin(\angle(dae))l(ad)l(eg))$

35. Перпендикуляр, проведенный к касательной в точке касания, проходит через центр окружности.

    $\forall_{ABCDEef}(E \in$ прямая(ef) & $E \in$ прямая(CD) & $E \in$ окружность(AB) &
    прямая$(ef) \perp$ прямая(CD) & прямая(CD) $-$касательная к окружность$(AB) \to$
    $A \in$ прямая$(ef))$

Приемы по аналитической геометрии

1. Вычитание векторов с общим концом.

   $\forall_{abC}($вектор$(aC) -$ вектор$(ab) =$ вектор$(bC))$

2. Преобразование условия ортогональности векторов в условие перпендикулярности прямых.

   $\forall_{ABCD}(\neg(A = B)$ & $\neg(C = D) \to$ вектор$(AB) \perp$ вектор$(CD) \leftrightarrow$
   прямая$(AB) \perp$ прямая$(CD))$

3. Исключение квантора.

   $\forall_{ax}(\neg(a =$ вектор0$)$ & Вектор$(x) \to$ коллинеарны$(a, x) \leftrightarrow$
   $\exists_y(x = ya$ & $y -$ число$))$

   Замена выполняется справа налево.

4. Решение простейшего векторного уравнения.

   $\forall_{acd}($Вектор$(d) \to d = ac \leftrightarrow \neg(a = 0)$ & $c = (1/a)d$ $\vee$ $a = 0$ & $d =$ вектор0$)$

   Прием имеет заголовок "второйтерм" и применяется к условию задачи на описание либо посылке задачи на исследование. Выражение c содержит неизвестные, а выражения a, d - не содержат.

5. Вывод соотношения пропорциональности для длин векторов из соотношения пропорциональности для векторов.

   $\forall_{abcd}(ab = cd \to |a|\text{длина}(b) = |c|\text{длина}(d))$

   Прием имеет заголовок "вывод". b, d - векторы; a, c - числа.

6. Усмотрение равенства нулю скалярного произведения ортогональных векторов.

   $\forall_{ab}(a \perp b \to \text{скалумнож}(a, b) = 0)$

   Прием имеет заголовок "второйтерм".

7. Скалярное произведение коллинеарных векторов.

   $\forall_{cde}(\neg(d = 0)\ \&\ \text{однонаправлены}(c, de) \to \text{скалумнож}(c, e) = \text{длина}(c)\text{длина}(e) \cdot \text{sg}(d))$

   Прием имеет заголовок "вывод". Указатель "контрольвывода" инициирует его срабатывание при усмотрении выражения "скалумнож(c, e)". Второй антецедент идентифицируется с посылкой.

8. Сведение угла между векторами к обычному углу.

   $\forall abcC(\text{разныеточки}(a, b)\ \&\ \text{разныеточки}(c, C)\ \&\ a \in \text{отрезок}(bc) \to \text{уголмежду}(\text{вектор}(ab), \text{вектор}(cC)) = \angle(bcC))$

   Прием имеет заголовок "вывод". Указатель "контрольвывода" инициирует его срабатывание при усмотрении выражения "уголмежду(вектор(ab), вектор(cC))".

9. Исключение несущественной неизвестной.

   $\forall_{be}(\text{однонаправлены}(b, e) \to \exists_a(\text{Вектор}(a)\ \&\ \text{однонаправлены}(a, b)\ \&\ \text{однонаправлены}(a, e)))$

   Прием имеет заголовок "связка".

10. Координаты суммы векторов.

    $\forall_{abceglmK}(\text{коорд}(c, K) = (l, m)\ \&\ \text{коорд}(a, K) = (e, g)\ \&\ b = c + a \to \text{коорд}(b, K) = (e + l, g + m))$

    Прием имеет заголовок "вывод". Первый и последний антецеденты идентифицируются с посылками. Выражения e, l, g, m не содержат невырожденных числовых атомов.

11. Отбрасывание минуса перед вектором.

    $\forall_{abcK}(b\text{—число}\ \&\ c\text{—число}\ \&\ \text{Вектор}(a) \to \text{коорд}(-a, K) = (b, c) \leftrightarrow \text{коорд}(a, K) = (-b, -c))$

    Прием имеет заголовок "второйтерм".

12. Вывод соотношения для координат вектора, параллельного прямой, проходящей через заданные точки.

    $\forall_{defgjklpqB}(\text{коорд}(B,l) = (p,q)\ \&\ B \parallel \text{прямая}(jk)\ \&\ \text{Вектор}(B)\ \&\ \text{коорд}(j,l) = (d,e)\ \&\ \text{коорд}(k,l) = (f,g) \to (g-e)p + (d-f)q = 0)$

    Прием имеет заголовок "вывод". Все антецеденты, кроме третьего, идентифицируются с посылками.

13. Усмотрение отрицательного скалярного произведени, если угол между векторами тупой.

    $\forall_{ABCKefgh}(\text{разныеточки}(A,B)\ \&\ \text{разныеточки}(A,C)\ \&\ \text{коорд}(\text{вектор}(AB),K) = (f,g)\ \&\ \text{коорд}(\text{вектор}(AC),K) = (e,h)\ \&\ \pi/2 < \angle(BAC)\ \&\ \text{прямкоорд}(K) \to ef + gh < 0)$

    Прием имеет заголовок "вывод". Три последних антецедента идентифицируются с посылками.

14. Определение координат суммы векторов.

    $\forall_{acgiqrszK}(\text{коорд}(c,K) = (r,s,z)\ \&\ \text{коорд}(a,K) = (g,i,q) \to \text{коорд}(c+a,K) = (g+r, i+s, q+z))$

    Прием имеет заголовок "второйтерм". Оба антецедента выделены указателем "идентификатор".

15. Связь между координатами пропорциональных векторов.

    $\forall_{abcdghjrK}(\text{коорд}(a,K) = (h,j,r)\ \&\ \text{коорд}(ga,K) = (b,c,d) \to b = gh)$

    Прием имеет заголовок "вывод". Антецеденты идентифицируются с посылками.

16. Соотношение для координат вершин прямоугольного треугольника.

    $\forall_{ABCabcdefimnp}(\text{коорд}(A,i) = (a,b,c)\ \&\ \text{коорд}(B,i) = (d,e,f)\ \&\ \text{коорд}(C,i) = (m,n,p)\ \&\ \text{прямая}(AB) \perp \text{прямая}(AC)\ \&\ \text{прямкоорд}(i) \to (d-a)(m-a) + (e-b)(n-b) + (f-c)(p-c) = 0)$

    Прием имеет заголовок "вывод". Все антецеденты, кроме четвертого, выделенного указателем "усм", идентифицируются с посылками.

17. Концы вертикально направленного вектора.

    $\forall_{dej}(d - \text{точка}\ \&\ e - \text{точка}\ \&\ \text{прямкоорд}(j)\ \&\ \text{вертикнапр}(\text{вектор}(de),j) \to \text{крд}(d,j,3) = \text{крд}(e,j,3) \leftrightarrow d = e)$

    Прием имеет заголовок "второйтерм". Третий антецедент идентифицируется непосредственно, остальные - обрабатываются проверочными операторами.

    $\forall_{bcK}(\text{вертикнапр}(\text{вектор}(bc),K)\ \&\ \text{Трехмерн}(K) \to \text{крд}(b,K,1) = \text{крд}(c,K,1))$

    Прием имеет заголовок "вывод".

18. Переформулировка посылки через координаты.

$\forall_{bcK}(\text{Трехмерн}(K) \to \text{влево}(\text{вектор}(bc), K) \leftrightarrow \text{крд}(b, K, 2) = \text{крд}(c, K, 2)$ & $\text{крд}(b, K, 3) = \text{крд}(c, K, 3)$ & $0 \le -\text{крд}(c, K, 1) + \text{крд}(b, K, 1))$

Прием имеет заголовок "второйтерм".

19. Отбрасывание множителя перед вектором.

$\forall_{bce}(b < 0$ & $\text{Трехмерн}(e) \to \text{вправо}(bc, e) \leftrightarrow \text{влево}(c, e))$

Прием имеет заголовок "второйтерм".

20. Условие на коэфффициенты уравнений параллельных прямых.

$\forall_{ABKabcdef}(\text{коорд}(A, K) = \text{set}_{xy}(c + ax + by = 0$ & $x - \text{число}$ & $y - \text{число})$ & $\text{коорд}(B, K) = \text{set}_{uv}(f + du + ev = 0$ & $u - \text{число}$ & $v - \text{число})$ & $\text{Прямая}(A)$ & $\text{Прямая}(B)$ & $A \parallel B \to ae - bd = 0)$

Прием имеет заголовок "вывод". Заметим, что при вводе приемов по аналитической геометрии вручную прямые задавались только выражением "прямая(...)". Система вывода теорем получла альтернативные их версии, где прямая обозначалсь переменной. Приемы, основанные на таких версиях, в итоге заменили многие из первоначально созданных приемов. Этим и объясняется то, что многие приводимые далее приемы для прямых попали в разряд "автоматически созданных", хотя их аналоги имелись до этого.

21. Переформулировка условия задачи через координаты.

$\forall_{ABKabcdepqr}(\text{коорд}(e, K) = \text{set}_{xy}(r + px + qy = 0$ & $x - \text{число}$ & $y - \text{число})$ & $\text{коорд}(A, K) = (a, b)$ & $\text{коорд}(B, K) = (c, d)$ & $\text{Прямая}(e) \to$ $\text{однасторона}(A, B, e) \leftrightarrow 0 \le (r + ap + bq)(r + cp + dq))$

Прием имеет заголовок "второйтерм" и применяется к подутверждениюю условия задачи. Первые три антецедента идентифицируются с утверждениями из контекста, последний - обрабатывается проверочными операторами.

22. Вывод неравенства на координаты точек, раположенных по одну сторону от прямой.

$\forall_{ABKabcdepqr}(\text{коорд}(e, K) = \text{set}_{xy}(r + px + qy = 0$ & $x - \text{число}$ & $y - \text{число})$ & $\text{коорд}(A, K) = (a, b)$ & $\text{коорд}(B, K) = (c, d)$ & $\text{Прямая}(e)$ & $\text{однасторона}(A, B, e) \to 0 \le (r + ap + bq)(r + cp + dq))$

Прием имеет заголовок "вывод". Второй и третий антецеденты выделены указателем "идентификатор", остальные - идентифицируются с посылками.

23. Переход от параметрического задания уравнения прямой в пространстве к стандартному.

$\forall_{abcdef}(\neg(d^2 + e^2 + f^2 = 0) \to \text{set}_{xyz}(x - \text{число}$ & $y - \text{число}$ & $z - \text{число}$ & $\text{пропорцнаборы}((d, e, f), (x - a, y - b, z - c))) = \text{set}_{xyz}(\exists_g(x = a + dg$ & $y = b + eg$ & $z = c + fg$ & $g - \text{число})))$

$\forall_{abcdf}(\neg(d^2+f^2=0) \to \text{set}_{xyz}(x - \text{число}\ \&\ y - \text{число}\ \&\ z - \text{число}\ \&$ пропорцнаборы$((d,0,f),(x-a,y-b,z-c))) = \text{set}_{xyz}(\exists_g(x = a+dg\ \&$ $z = c+fg\ \&\ g - \text{число})\ \&\ y = b))$

Замена выполняется справа налево.

24. Соотношения для коэффициентов уравнений параллельных прямых.

$\forall_{Kabcdefghijkpqr}(\text{коорд}(i,K) = \text{set}_{xyz}(x - \text{число}\ \&\ y - \text{число}\ \&\ z - \text{число}\ \&$ пропорцнаборы$((d,e,f),(a+x,b+y,c+z)))\ \&\ \text{коорд}(j,K) =$ $\text{set}_{uvw}(u - \text{число}\ \&\ v - \text{число}\ \&\ w - \text{число}\ \&$ пропорцнаборы$((p,q,r)$ $,(g+u,h+v,k+w)))\ \&\ \text{Прямая}(i)\ \&\ \text{Прямая}(j)\ \&\ i \parallel j \to dq - ep = 0\ \&$ $dr - fp = 0\ \&\ er - fq = 0)$

Прием имеет заголовок "вывод".

25. Условие параллельности прямой и плоскости.

$\forall_{CDEKabcdefgpqrs}(\text{коорд}(g,K) = \text{set}_{xyz}(x - \text{число}\ \&\ y - \text{число}\ \&\ z - \text{число}\ \&$ пропорцнаборы$((d,e,f),(a+x,b+y,c+z)))\ \&\ \text{коорд}(\text{плоскость}(CDE),K)$ $= \text{set}_{uvw}(s+pu+qv+rw = 0\ \&\ u - \text{число}\ \&\ v - \text{число}\ \&\ w - \text{число})\ \&$ $\text{Прямая}(g) \to g \parallel \text{плоскость}(CDE) \leftrightarrow dp+eq+fr)$

Прием имеет заголовок "второйтерм" и применяется к условию задачи.

26. Расшифровка содержащего неизвестные условия задачи на описание.

$\forall_E(\text{эллипс}(E) \leftrightarrow \exists_{Kab}(\text{прямкоорд}(K)\ \&\ a - \text{число}\ \&\ b - \text{число}\ \&\ 0 < b\ \&$ $0 \le a - b\ \&\ \text{коорд}(E,K) = \text{set}_{xy}(x^2/a^2 + y^2/b^2 = 1\ \&\ x - \text{число}\ \&\ y - \text{число})))$

Приемы по линейной алгебре

1. Прием проверочного оператора, усматривающего перестановку.

$\forall_{Af}(\text{конечное}(A)\ \&\ \text{Dom}(f) = \{1,\ldots,\text{card}(A)\}\ \&\ \text{Val}(f) = A \to$ $\text{перестановка}(f,A))$

2. Вынесение наружу численного коэффициента при умножении матриц.

$\forall_{bcdkmn}(c - \text{число}\ \&\ \text{матр}(b,a,n,k)\ \&\ \text{матр}(d,e,m,f)\ \&\ \mathbb{R} = a\ \&\ \mathbb{R} = e\ \&$ $n = f \to (cd)b = c(db))$

Прием имеет заголовок "второйтерм". Рассматриваются две различных операции умножения - операция умножения числовой функции на число "числкоэфф" и операция умножения матриц "умножматр". Второй и третий антецеденты обрабатываются пакетным синтезатором определения параметров матрицы.

3. Умножение на единичную матрицу.

$\forall_{aem}(\text{матр}(a,b,m,c)\ \&\ \mathbb{R} = b\ \&\ e = c \to a \cdot \text{единичнматр}(e) = a)$

Прием имеет заголовок "второйтерм".

4. Умножение на скалярную матрицу.

   $\forall_{bcfk}(c - \text{число}\ \&\ \text{матр}(b,a,d,k)\ \&\ \mathbb{R} = a\ \&\ f = d \to \text{скалярнматр}(c,f) \cdot b = cb)$

   Прием имеет заголовок "второйтерм".

5. Прием проверочного оператора, усматривающего матрицу.

   $\forall_{abkln}(\text{матр}(a,\mathbb{R},l,n)\ \&\ \text{матр}(b,\mathbb{R},n,k) \to \text{матр}(ab,\mathbb{R},l,k))$

Приемы по математическому анализу

1. Предел последовательности.

   Обычно пределы последовательностей вычисляются решателем путем обращения к нормализатору "нормпредел", определяющему пределы функций вещественной переменной. Если это не срабатывает, предусмотрены дополнительные приемы. В частности, приемы, декомпозирующие вычисление предела для случаев суммы, умножения, и т.п. Генератор приемов предложил несколько таких приемов. Приведем часть из них.

   $\forall_{abfg}(\lim(\lambda_c(f(c), c - \text{натуральное})) = a\ \&\ \lim(\lambda_c(g(c), c - \text{натуральное})) = b\ \&\ a - \text{число}\ \&\ b - \text{число} \to \lim(\lambda_c(f(c)g(c), c - \text{натуральное})) = ab)$

   $\forall_{abfg}(\lim(\lambda_c(f(c), c - \text{натуральное})) = a\ \&\ \lim(\lambda_c(g(c), c - \text{натуральное})) = b\ \&\ \neg(b = 0)\ \&\ a - \text{число}\ \&\ b - \text{число} \to \lim(\lambda_c(f(c)/g(c), c - \text{натуральное})) = a/b)$

   $\forall_{af}(\lim(\lambda_c(f(c), c - \text{натуральное})) = a\ \&\ a - \text{число} \to \lim(\lambda_c(|f(c)|, c - \text{натуральное})) = |a|)$

   Приемы имеют заголовок "второйтерм".

2. Усмотрение сходимости последовательностей

   Несколько приемов были созданы для проверочного оператора "усмсходится".

   $\forall_{adg}(\text{сходится}(g)\ \&\ \text{последовательность}(g,\mathbb{R}) \to \text{сходится}(\lambda_n(ag(n)/d, n - \text{натуральное})))$

   $\forall_{fg}(\text{сходится}(f)\ \&\ \text{сходится}(g)\ \&\ \text{последовательность}(f,\mathbb{R})\ \&\ \text{последовательность}(g,\mathbb{R}) \to \text{сходится}(\lambda_n(f(n)g(n), n - \text{натуральное})))$

3. Вывод определения условия сохранения знака в окрестности точки.

   $\forall_{af}(a - \text{число}\ \&\ \text{сохрзнака}(f,a) \to \exists_b(b - \text{число}\ \&\ 0 < b\ \&\ \text{Окрестн}(a,b,0) \subseteq \text{Dom}(f)\ \&\ \forall_{xy}(x \in (a-b,a)\ \&\ y \in (a,a+b) \to 0 < f(x)f(y))))$

   Прием применяется в задачах на доказательство либо на исследование.

4. Приемы проверочного оператора "переменазнака".

   $\forall_{ade}(\text{переменазнака}(\lambda_c(e(c), d(c)), a) \to \text{переменазнака}(\lambda_c(\arcsin e(c), d(c)), a))$

   $\forall_{adeg}(\neg(e = 0)\ \&\ \text{переменазнака}(\lambda_c(g(c), d(c)), a) \to$
   $\text{переменазнака}(\lambda_c(eg(c), d(c)), a))$

   $\forall_{adeg}(\text{переменазнака}(\lambda_c(g(c), d(c)), a) \to \text{переменазнака}(\lambda_c(g(c/e), d(c)), a))$

5. Вывод определения ограниченности функции на множестве.

   $\forall_{fA}(\text{ограничена}(f, A) \to \exists_b(b - \text{число}\ \&\ \forall_x(x \in A \to |f(x)| < b)))$

   Прием применяется в задачах на доказательство либо на исследование.

6. Попытка использования параметрического описания при подборе примера.

   $\forall_{fA}(\text{ограничена}(f, A) \leftrightarrow \exists_b(b - \text{число}\ \&\ \forall_x(x \in A \to |f(x)| < b)))$

   Прием имеет заголовок "параметризация". Он применяется к содержащему неизвестные условию задачи на описание, имеющей цель "пример" либо "параметризация".

7. Приемы проверочного оператора "усмограничена". Приводятся лишь несколько приемов.

   $\forall_{cdA}(\text{ограничена}(\lambda_a(d(a), c(a)), A) \to \text{ограничена}(\lambda_a(-d(a), c(a)), A))$

   $\forall_{cdeA}(\text{ограничена}(\lambda_a(d(a), c(a)), A)\ \&\ \text{ограничена}(\lambda_a(e(a), c(a)), A) \to$
   $\text{ограничена}(\lambda_a(d(a) + e(a), c(a)), A))$

   $\forall_{cdeA}(\neg(\text{точкаприкосн}(0, \text{Val}(\lambda_a(e(a), c(a))))))\ \&\ \text{ограничена}(\lambda_a(d(a), c(a)), A) \to$
   $\text{ограничена}(\lambda_a(d(a)/e(a), c(a)), A))$

   $\forall_{cdA}(\text{ограничена}(\lambda_a(\arccos d(a), c(a)), A))$

8. Приемы проверочного оператора "усмубывает".

   $\forall_{acde}(0 \le d(b)\ \&\ 0 < e(b)\ \&\ \text{невозрастает}(\lambda_b(e(b), c(b)), a)\ \&$
   $\text{убывает}(\lambda_b(d(b), c(b)), a) \to \text{убывает}(\lambda_b(d(b)e(b), c(b)), a))$

   Антецеденты обрабатываются проверочными операторами, причем первым двум операторам передаются дополнительные посылки "$c(b)$", "$b \in a$".

   $\forall_{acde}(d < 0\ \&\ \text{возрастает}(\lambda_b(e(b), c(b)), a) \to \text{убывает}(\lambda_b(e(b)/d, c(b)), a))$

   $\forall_{acde}(0 < d - 1\ \&\ 0 < e(b)\ \&\ \text{убывает}(\lambda_b(e(b), c(b)), a) \to$
   $\text{убывает}(\lambda_b(\log_d e(b), c(b)), a))$

   Антецеденты обрабатываются проверочными операторами, причем второму оператору передаются дополнительные посылки "$c(b)$", "$b \in a$".

   $\forall_{acd}(0 < d(b)\ \&\ \text{возрастает}(\lambda_b(d(b), c(b)), a) \to \text{убывает}(\lambda_b(\operatorname{cth} d(b), c(b)), a))$

   Антецеденты обрабатываются проверочными операторами, причем первому оператору передаются дополнительные посылки "$c(b)$", "$b \in a$".

9. Приемы проверочного оператора "усмчетнаяфункция".

   $\forall_{bcd}(0 < c(a)$ & четнаяфункция$(\lambda_a(c(a), b(a)))$ & четнаяфункция$(\lambda_a(d(a), b(a))) \to$ четнаяфункция$(\lambda_a(c(a)^{d(a)}, b(a)))$

   Антецеденты обрабатываются проверочными операторами, причем первому оператору передается дополнительная посылка "$b(a)$".

   $\forall_{bc}($нечетнаяфункция$(\lambda_a(c(a)^{d(a)}, b(a))) \to$ четнаяфункция$(\lambda_a(\operatorname{ch} c(a), b(a))))$

Приемы по комплексным числам

1. Попытка параметризации при подборе примера.

   $\forall_z(z -$ комплексное $\leftrightarrow \exists_{xy}(x -$ число & $y -$ число & $z = x + iy))$

   Прием имеет заголовок "параметризация" и применяется к условию задачи на описание, имеющей цель "пример" либо "параметризация". Переменная z - неизвестная.

2. Вынесение вещественного множителя из-под вещественной части.

   $\forall_{abf}(a -$ число $\to \operatorname{Re}(af/b) = a\operatorname{Re}(f/b))$

3. Вещественная часть сопряженного числа.

   $\forall_c(\operatorname{Re}($сопряженное$(c)) = \operatorname{Re}(c))$

4. Вещественная часть произведения.

   $\forall_{abcde}(c = ad - be$ & $a -$ число & $b -$ число & $d -$ число & $e -$ число $\to$
   $\operatorname{Re}((a + bi)(d + ei)) = c)$

   Прием относится к нормализатору общей стандартизации "нормвещественнаячасть". Переменные a, b, d, e идентифицируются с десятичными константами. Первый антецедент выделен указателем "программа".

5. Мнимая часть дробного выражения.

   $\forall_{acf}(\neg(a = 0)$ & $a -$ комплексное & $c -$ комплексное & $f -$ комплексное $\to$
   $\operatorname{Im}(cf/a) = \operatorname{Re}(f)\operatorname{Im}(c/a) + \operatorname{Re}(c/a)\operatorname{Im}(f))$

   Прием имеет заголовок "вывод" и применяется в задачах на доказательство либо на исследование. Его срабатывание инициируется усмотрением выражения "$\operatorname{Im}(cf/a)$".

6. Свертка условий в равенство двух выражений.

   $\forall_z(e = z \leftrightarrow \operatorname{Re}(e) = \operatorname{Re}(z)$ & $\operatorname{Im}(e) = \operatorname{Im}(z)$ & $z -$ комплексное$)$

   Прием применяется к трем посылкам задачи. Замена вполняется справа налево. Так как изначально из контекста должны были усматриваться условия на о.д.з. этих посылок, то после замены остается указание на то, что e комплексное.

7. Сумма квадратов вещественной и мнимой части.

   $\forall_{az}(a(|z|)^2 = a(\mathrm{Re}(z))^2 = a(\mathrm{Im}(z))^2)$

   Прием применяется справа налево.

8. Сопряженное к сопряженному.

   $\forall_c(\text{сопряженное}(\text{сопряженное}(c)) = c)$

   Прием относится к нормализатору "нормсопряженное".

9. Равенство сопряженных чисел.

   $\forall_{ac}(-\text{сопряженное}(a) + \text{сопряженное}(c) = 0 \leftrightarrow c - a = 0)$

   Прием имеет заголовок "второйтерм".

10. Исключение символов вещественной и мнимой части.

    $\forall_{az}(a\mathrm{Re}(z) + ai\mathrm{Im}(z) = az)$

    Прием относится к нормализатору "нормПлюс".

11. Упрощение квантора за счет перехода к новым переменным.

    $\forall_{abc}(\exists_{df}(b(a+d,f)\ \&\ d - \text{комплексное}\ \&\ c(f)) \leftrightarrow \exists_{df}(b(d,f)\ \&$
    $d - \text{комплексное}\ \&\ c(f)))$

    Прием имеет заголовок "второйтерм". Переменные b, c функциональные.

12. Исключение внешнего минуса.

    $\forall_{bcde}(c(d-e)/b = -(c(e-d)/b))$

    Прием применяется к условию задачи на преобразование, имеющей цели "упростить" либо "длина". Заменяемое вхождение - корневое. Замена происходит справа налево.

13. Лексикографическая стандартизация.

    $\forall_{cdefn}(n - \text{целое} \to (e-d)^n(f-c)^n = (c-f)^n(d-e)^n)$

    Замена выполняется слева направо. Предварительно проверяется, что новое выражение лексикографически предшествует старому.

14. Исключение несущественной неизвестной.

    $\forall_{ab}(b - \text{комплексное} \to \exists_z(b = az\ \&\ z - \text{комплексное}) \leftrightarrow \neg(a = 0)\ \&\ \neg(b = 0)\ \vee\ b = 0)$

    Прием имеет заголовок "связка".

15. Упрощение выражения под описателем относительно варьируемой переменной.

    $\forall_{abcdef}(ab/(df) + ac/(ef) = a(b/d + c/e)/f)$

    Замена выполняется слева направо. Выражение a группирует в себе все множители, связываемые внешним описателем "отображение" либо "класс". Выражения b, c, d, e, f этим описателем не связаны.

16. Разложение на множители суммы квадратов.

    $\forall_{ac}(a^2 + c^2 = (a - ci)(a + ci))$

    Прием относится к нормализатору разложения на комплексные множители "видУмножение".

    Остальные автоматически созданные приемы для комплексных чисел, в основном, представляют собой аналоги приемов, имевшихся для вещественных чисел. Их примеры мы не приводим.

Глава 26

Дополнение (2022 - 2025)

Первые главы данной книги были написаны несколько лет назад, почти одновременно с девятым томом монографии "Компьютерное моделирование логических процессов". За это время продолжалась работа по развитию логической системы и накопился материал, который предполагается включить в десятый том. Однако, пока этот том не вышел, в настоящем учебном пособии имеет смысл кратко рассказать о новых возможностях системы, чтобы облегчить работу с ее последней версией.

Классификация приемов вывода теорем

Описание приемов вывода теорем содержится в 9 томе монографии. Там пересказаны ЛОС-программы приемов, и работа каждой такой программы проиллюстрирована на примере. Этот пересказ осуществлялся в терминах программных переменных ЛОСа. Он получился многословным и трудно воспринимаемым, хотя и полным. Это создало потребность в определенной "расшифровке" действий приемов - кратком описании смысла, достаточном для точного понимания происходящего. Так как приемов вывода теорем накопилось много и их список постоянно пополняется (в том числе за время после написания 9 тома), необходимо было как-то классифицировать данные приемы. Напомним, что классификация приемов ГЕНОЛОГа позволила создать логический ассемблер и перейти к автоматическому созданию приемов. Получится ли аналогичным образом найти способ автоматического создания приемов вывода теорем, или классификация этих приемов позволит лишь довести их коллекцию до необходимой полноты - покажут будущие исследования. Пока удалось создать классификацию таких приемов и предпринять указанную выше их "расшифровку". Заметим, что, по сравнению с девятым томом монографии, к описанию добавилось множество новых приемов.

Классификация приемов вывода теорем реализована в виде древовидного оглавления, концевые пункты которого суть тексты расшифровки отдельных приемов. Чтобы перейти в данное оглавление, нужно сначала из корневого меню логической системы перейти в оглавление программ (клавиша "л"), затем нажать клавишу "т". После текста каждого концевого пункта этого оглавления помещается терм "программа(SN)". Он ссылается на контрольную точку "прием(N)" программы логического символа S, реализующей прием вывода теорем, соответствующий данному концевому пункту.

Можно посмотреть как ЛОС-программу приема вывода теорем, так и примеры использования данного приема при выводе теорем. В обоих случаях, прежде всего,

находясь на концевом пункте оглавления, следует нажать "курсор вправо". Это нажатие переводит в концевой пункт оглавления программ, соответствующий приему. Напомним, что к разделу оглавления программ, содержащему ссылки на приемы вывода теорем, можно перейти от корня оглавления вдоль пути "База теорем" - "Программирующий вывод" - "Приемы вывода теорем (справочник ПРОГРВЫВОД)".

Далее, как обычно, нажатие "курсор вправо" переводит в ЛОС-программу приема; нажатие "л" - в просмотр списка теорем, при выводе которых использовался прием. Чтобы для текущей выбранной теоремы списка посмотреть историю ее вывода, используются клавиши "курсор вправо - влево". Заметим, что часто рассматриваемый прием вывода использовался не в конце этой истории, а внутри нее. В каждом кадре, возникающем при просмотре, в верхней части прорисована результирующая теорема, в нижней - одна или несколько теорем, из которых она была выведена. Под верхней теоремой располагается список ее характеристик, затем - ссылка на прием вывода - S, u, N, где u - уровень срабатывания приема; S, N - те же, что и выше. В той же строчке, что ссылка на прием, повторяется название приема из оглавления приемов. Если на этом кадре нажать "ъ", то реализуется повторный запуск программы приема, с остановкой при обращении к процедуре "регтеор", регистрирующей полученную теорему. Чтобы вернуться в исходный кадр, нажимаются "0" и Enter. Для возвращения в список теорем, при выводе которых использован прием, наживается Esc. Далее, для возвращения в оглавление программ, нажимается "курсор вправо". Наконец, для возвращения в тот пункт классификации приемов, из которого перешли в оглавление программ, достаточно нажать "т".

Пока классификация приемов приемов вывода теорем и их расшифровка имеются лишь в программе логической системы. Однако, предполагается уделить им отдельную главу в очередном томе монографии.

Автоматическое создание приемов справочников поиска теорем и уточнение решающих правил приемов вывода теорем

Напомним, что база теорем логической системы разбита на так называемые ячейки вывода. Ячейка вывода представляет собой подраздел, у которого все пункты концевые. Первый пункт содержит стартовые теоремы вывода, остальные пункты - теоремы либо группы теорем, которые должны быть выведены из стартовых. Если теорема уже выводится системой, нижняя горизонтальная черта ее отображения на экране имеет зеленый цвет, иначе - черный. Система вывода теорем запускается при просмотре любой из стартовых теорем ячейки нажатием "л". Чтобы сохранить в архиве базы теорем все траектории вывода, вместо этого нажимается "Л". По окончании цикла вывода итоговый список теорем (список вывода) отображается на экране.

Прием вывода теорем получает в качестве входного данного некоторую теорему T, список ее характеристик, а также предысторию получения данной теоремы в текущем цикле вывода. В некоторых случаях новая теорема получается из теоремы T без привлечения дополнительных теорем, например, при помощи решения каких-либо вспомогательных задач. Однако, чаще для вывода необходима дополнительная теорема D, иногда - несколько таких теорем. Для поиска дополнительной теоремы используется три способа. Во-первых, просмотр раздела базы теорем, связанного с некоторым понятием. Во-вторых, дополнительная теорема может извлекаться из

списка теорем, полученных в текущем цикле вывода. Наконец (и это самый типичный вариант), для поиска дополнительной теоремы может использоваться так называемый справочник поиска теорем. Имеется несколько сотен таких справочников. Посмотреть их можно в разделе "База теорем" - "Спецификатор" - "Процедуры справочников "приемы" и "протокол" " - "Справочники" - "Поиск теорем" оглавления программ. По названию N справочника можно получить описание его действий в справочной информации о символе N (например, через F3 из просмотра ЛОС-программы символа N).

Справочнику поиска теорем передается в качестве входного данного некоторый логический символ S, связанны с теоремой T. По этому символу он выдает список кандидатов на роль дополнительной теоремы D. Каждая отдельная версия теоремы D предлагается отдельным приемом справочника, причем такой прием может быть создан по теореме D автоматически. Собственно, каждый раз, когда возникал новый справочник поиска теорем, сразу же создавались программа компилятора ГЕНОЛОГа для создания программы приема справочника по его спецификации, а также вставка в программу спецификатора, создающая спецификацию приема по теореме, если эта теорема удовлетворяет критериям справочника.

Таким образом, все предпосылки для авоматического создания приемов справочника поиска теорем изначально имелись. Однако, добавление новых приемов для уже созданного справочника поиска теорем происходило очень осторожно. Как правило, создавался лишь единственный прием, обеспечивающий вывод текущей прорабатываемой теоремы из базы теорем. Даже при таком подходе некоторые из справочников накопили сравнительно большое число своих приемов.

Попытки создать сразу все приемы для всех справочников поиска теорем предпринимались, но сразу столкнулись с определенными трудностями. Некоторые справочники, даже при создании множества их новых приемов, ничего не добавляли к итоговым выводам в базе теорем. Другие, наоборот, даже при добавлении сравнительно простых и естественных приемов, давали сильнейший всплеск активности и переполняли все разумные лимиты при выводе. Причина этого оказалась простой и типичной даже для обычного решателя: прием, созданный на единичном примере, требует длительного доучивания на множестве других примеров. Иначе его действия будут немотивированными и разрушат процесс решения задачи. Понимание этого позволило рассматривать новые, автоматически созданные приемы справочников поиска теорем, как необходимый обучающий материал для доработки приемов вывода теорем. Был предпринят цикл автоматического создания приемов для всех имеющихся в системе справочников поиска теорем. Всплески активности анализировались, и предпринималось необходимое уточнение решающих правил приемов вывода теорем. Таким образом, удалось преодолеть "одноразовый" характер почти всех справочников. Во многих случаях уточнялись даже не решающие правила приема, а критерии отбора теорем для справочника.

Создать новый прием справочника поиска теорем можно, во-первых, при просмотре некоторой теоремы в базе теорем. Нажимается "г", клавишами "курсор вверх - вниз" выбирается нужная спецификация приема, и нажимается "б". Результат находится в разделе "Генератор приемов" буфера базы приемов.

Во-вторых, можно создать все приемы заданного справочника поиска теорем по разделу базы теорем. Нужный раздел выделяется в оглавлении базы теорем, нажимается "ъ", в окне текстового редактора вводится название справочника (либо сохраня-

ется то, которое в нем уже прорисовано), и нажимается Enter. Результаты создаются в буфере базы приемов. Если нужно сравнить новые приемы данного справочника со старыми, можно из просмотра любого нового приема нажать Ctr-ъ. Снизу появляется список теорем, по которым ранее создавались приемы данного справочника.

Перенести новые приемы справочника поиска теорем из буфера базы приемов в ее основные разделы можно нажатием Ctr-г. При этом знаком "+" будут помечены те пункты буфера, для которых перенесение фактически состоялось. Перед таким перенесением стоит предпринять полную прокрутку вывода теорем и определить те ячейки вывода, в которых количество полученных теорем резко возросло, либо часть теорем оказалась утеряна, либо вообще вывод был прерван. Это позволит предпринять необходимые коррекции приемов вывода теорем.

В-третьих, можно создать вообще все приемы всех справочников поиска теорем по заданному разделу базы приемов. Для этого вместо "ъ" нажимается Ctr-ъ. Во избежание переполнений, лучше этим пользоваться только для концевых пунктов оглавления базы теорем.

Автоматическое пополнение базы теорем

При запуске цикла вывода теорем в ячейке логического вывода на экране прорисовывается итоговый список теорем. Эти теоремы пока не зарегистрированы ни в каких файлах системы. Те из них, которые уже имеются в базе теорем (не обязательно в текущей ячейке), помечены символом "+". Помимо них, система обычно находит множество новых.

Однако, не все новые теоремы представляют интерес. Обычно они лишь дают материал для доработки приемов вывода теорем, уточняющей их целевую направленность: вводятся дополнительные ограничения на количественные и качественные параметры теорем, создаваемых приемом. Предпринималась попытка выявить такие ограничения, в зависимости от характеристики теоремы, путем полного просмотра базы теорем. Эти ограничения введены в качестве общих фильтров при регистрации теоремы в списке вывода.

Хотя указанные фильтры и позволили отсечь появляение непомерно громоздких следствий, остаток малополезных теорем остается достаточно большим. По-видимому, эту проблему до некоторой степени решает примерка приемов, созданных по теореме: сохраняются только приемы, не сводящиеся к другим, и таким образом сохраняются и их теоремы. Однако, для примерки нужно уметь создавать хорошие тестовые примеры, и всегда есть риск не найти того примера, в котором прием окажется по-настоящему необходимым. В результате доводка будет отсекать ценные теоремы. Поэтому, пока остается актуальной задача отсечения ценных теорем от малополезных без использования доводки.

Ввиду указанной трудности, пополнение базы теорем при помощи процедуры вывода теорем пока остается полуавтоматическим: отбор полезных теорем по итоговому списку вывода происходит вручную. Впрочем, обычно таких теорем обнаруживается достаточно много, иногда десятки. Все теоремы базы теорем, созданные автоматически, помечаются характериистикой "авт". Иногда такие теоремы составляют большинство теорем ячейки.

Для отбора теорем нужно сначала нажать "р". Тогда из списка исключаются все теоремы, уже имеющиеся в базе приемов. Далее, при помощи клавиши Insert, отби-

раются те теоремы, которые предполагается сохранить. Если отобранную теорему нужно удалить, на ней нажимается Delete. По окончании отбора нажимается клавиша "я". Это нажатие создает в конце списка пунктов текущей ячейки вывода новый пункт, помеченный " - - -". В нем и содержатся все отобранные теоремы. Остается только вручную перенести их в нужные пункты той же самой ячейки вывода (быть может, создать новые пункты). Чтобы выводы новых теорем сохранить в архиве базы теорем, после их регистрации нужно повторно запустить цикл вывода в этой ячейке нажатием "Л".

Анализ определений

Обычно в математическом учебнике или в монографии материал организуется "по понятиям". В каждом разделе вводятся новые понятия, и здесь же устанавливаются их основные свойства. Если ограничиться лишь малым числом понятий, список их свойств будет относительно ограниченным, и исследование окажется исчерпанным. Поэтому компьютерная логическая система, способная развивать собственные разделы теорий, должна уметь находить новые понятия, представляющие интерес.

Очень часто новые понятия в математике вводятся с помощью кванторных определений - начиная с простейшего теоретико-множественного включения и кончая такими "продвинутыми" понятиями, как предел, производная и интеграл. Представляло бы интерес проведение экспериментов с логической системой, в которой она выявляла бы кванторные определения новых отношений и операций, не выразимые бескванторно через уже имеющиеся.

Однако, это слишком упрощенный подход к автоматизации поиска новых интересных понятий. Чтобы охватить все многообразие типов реальных определений, постараться объяснить, как такие определения могли бы быть созданы автоматически и чем они "лучше" других возможных определений, была предпринята работа по созданию необходимого обучающего материала. Таким материалом стало "оглавление определений", в котором все представленные в базе теорем логической системы определения как-то отсортированы. Оглавление доступно из любой точки оглавления базы теорем либо из просмотра теоремы - достаточно нажать клавишу "П". Названием концевой точки оглавления служит логический символ - определяемые отношение либо операция. Нажатие "курсор вправо" на этом пункте переводит в просмотр теоремы (т.е. в базу теорем). Для возвращения в оглавление определений снова нажимается "П".

Синтез приемов без примерки и доводки

Создание приема по теореме происходит в логической системе по следующей последовательности действий.

Прежде всего, характеризатор сопровождает теорему некоторыми характеристиками. Иногда эти характеристики уже жестко фиксированы приемом вывода, получившим теорему.

Затем спецификатор по конкретной характеристике теоремы создает спецификацию приема - его задание на логическом ассмеблере. Он же может определенным образом преобразовать теорему - отбросить часть антецедентов, ввести функциональные переменные и т.п. Эти преобразования носят чисто технический характер и выполняются так называемым конвертором теорем.

Наконец, компилятор спецификаций преобразует спецификацию приема в его описание на ГЕНОЛОГе. Так как целевая направленность приема уже известна, здесь появляется возможность создать необходимые фильтры приема.

В описанной выше версии генератора приемов такие действия предпринимались для каждой теоремы, получаемой в цикле вывода теорем, запускаемом в конкретной ячейке вывода. Описания приемов на ГЕНОЛОГе компилировались в ЛОС-программу приема. По каждому новому приему создавался тестовый пример, позволявший проверить, что этот прием необходим для его решения. Затем вся группа новых приемов прогонялась доводчиком по задачнику системы и предпринимались попытки небольшими изменениями приема (варьирование уровня срабатывания и варьирование типа приема, регулировавшего степень мотивированности срабатывания) добиться того, чтобы новые приемы не портили старых решений задач. Эта версия генератора приемов была успешно опробована, и с ее помощью удалось добавить решателю порядка 2500 новых приемов. Заметим, что автоматически созданные приема ГЕНОЛОГа помечаются символом "авт" в указателях приема. Чтобы просмотреть первые 60 созданных автоматически приемов какого-либо раздела базы приемов, нужно войти в этот раздел и нажать "а". Для подсчета общего числа таких приемов в разделе нажимается Ctr-a. Результатом служит значение переменной x14.

Однако, для дальнейшего развития генератора приемов необходимо продолжить обучение основных его блоков - характеризатора, спецификатора и компилятора спецификаций. Здесь должны предвосхищаться возможные конфликты между приемами и оптимизироваться прочие их параметры. Чтобы предпринять такую доработку, была создана полуавтоматическая версия генератора приемов.

Для доработки фиксировался некоторый список типов приемов, и все приемы этих типов создавались без примерки и доводки по выбранному разделу базы теорем. Рассматривались только теоремы, уже имеющиеся в разделе, причем не обязательно отнесенные к какой-либо ячейке вывода. Характеристики теорем уже имелись, так что работал лишь конвейер "спецификатор" - "компилятор спецификаций" - "компилятор ГЕНОЛОГа". Новые приемы регистрировались в разделе "Генератор приемов" буфера базы приемов.

Для запуска указанного процесса выбирался раздел базы теорем, и на нем нажималась клавиша "Щ". Предварительно определялся список логических символов - типов создаваемых приемов (20-30 типов) и вставлялся в некоторый оператор ЛОС-программы. Выйти на этот оператор можно через раздел оглавления программм "Синтез приемов" - "Синтез приемов по разделу базы теорем" - "Запуск создания приемов решателя по текущему пункту оглавления (щ). При запуске по (Щ) используется список типов приемов, использованный в программе". Корректируется оператор "альтернатива(равно(x11 нормзадача) равно(справка(новприем x25)1) входит(x25 (... здесь помещается список типов ....)".

Далее начиналась собственно работа по доучиванию генератора приемов. Во-первых, анализировались действия конвертора теорем - при необходимости в него вводились коррекции, удаляющие избыточные антецеденты (подразумеваемые по о.д.з.) либо сохраняющие отброшенные. Во-вторых, определялись приемы, действия которых заведомо не соответствовали заявленной их целевой направленности. Здесь корректировался спецификатор, иногда уточнялись характеристики теоремы.

Далее выбирался раздел оглавления базы приемов, и все приемы из буфера базы приемов переносились в этот раздел. Для этого нажимается клавиша "Г". Все новые

приемы буфера (кроме не откомпилированных) переносятся в новый концевой пункт раздела, помеченный " - - -". Буфер базы приемов расчищается ("О").

Далее предпринимались прокрутки по задачнику для выявления конфликтов. Для малого раздела задачника прокрутка делалась последовательной, иначе - параллельной. Особые случаи сохранялись в буфере задачника. Далее начиналась работа с задачами, не решенными после добавления новых приемов либо сильно замедлившимися. Чтобы упростить обнаружение срабатывания новых приемов, в их программы делается вставка - обращение к оператору "нов(...)". Этот оператор вырожденный. Если вставить контрольную точку "трассировка(стоп 0" перед его завершающим оператором "выход", при каждом срабатывании любого нового приема будет выполняться выход в отладчик ЛОСа на этой точке. Анализ конфликтов позволил внести множество уточнений в различные блоки генератора приемов - вплоть до приемов вывода теорем. После успешной доработки приемов на одном разделе предпринималась уже полная прокрутка по задачнику (обычно на 3 машинах). Дополнительные особые случаи снова анализировались.

Даже если новые приемы ничего не портят в решениях задач, они могут нарушить цепочки вывода теорем. Поэтому далее предпринималась полная прокрутка вывода теорем по всей базе теорем. Предварительно удалялась контрольная точка из оператора "нов". Как правило, какие-то выводы теорем утрачивались. Например, из-за дополнительных упрощений, выполненных новыми приемами. Иногда какие-то выводы оказывались прерваны - из-за зацикливаний либо появления слишком большого количества следствий. Все это тоже анализировалось, и вводились необходимые коррекции как в новые приемы, так и в генератор приемов.

Наконец, после того, как приемы прошли все проверки, выбирался тот концевой пункт базы приемов, в котором они находятся, и на нем нажималась клавиша "Н". Это удаляет вставки "нов(...)" в программы новых приемов. Далее остается только перенести вручную новые приемы в соответствующие разделы оглавления, а затем удалить пункт " - - -".

При обнаружении дублирования - заведомого поглощения действий нового приема действиями старого - один из приемов отбрасывался. Следует заметить, что в действительности лишь малая доля новых приемов требовала каких-либо коррекций - подавляющее большинство обычно сохранялись без изменений. Быть может, для небольшого ускорения иногда вручную подправлялись уровни срабатывания.

Предполагается продолжать указанный процесс обучения генератора приемов. Пока проработана лишь крайне малая часть типов приемов - самые простые и хорошо мотивированные действия. Однако, уже вместо 2500 автоматически созданных приемов их стало примерно 5500. В ряде случаев новый прием ускорял решение какой-либо задачи задачника. Таким образом, данная "полуавтоматическая" версия генератора приемов, в сочетании с системой автоматического вывода теорем, уже сейчас представляет собой серьезный инструмент, облегчающий обучение решателя задач.

К достоинствам этой версии следует отнести то, что приемы создаются почти "в реальном времени", в то время как версия с доводкой оказалось исключительно трудоемкой и медленной. Конечно, эта версия - лишь инструмент для доучивания генератора приемов. В перспективе, хотелось бы, анализируя всевозможные конфликты между приемами, предвосхищать их настолько хорошо, чтобы примерка и доводка оказались вообще не нужны. Разумеется, точный учет взаимодействия приемов потребует анализа имеющихся в предметной области теорем и создания протоколов

базы теорем, регламентирующих такое взаимодействие. По существу, это развитие науки о логических процессах.

Компилятор с ЛОСа на СИ

Логические вычисления выполняются при помощи простейших операторов, работающих с термами и задачами. Такие операторы реализованы на ЛОСе в виде программ на СИ, причем обращения интерпретатора к этим программам занимают малое время по сравнению с работой самих программ. Переход к компилятору здесь вряд ли дал бы существенное ускорение. Другое дело - нелогические вычисления, имеющие дело с числами, матрицами, графами и т.п. Здесь уже действия интерпретатора начинают поглощать существенную относительную трудоемкость, и целесообразна компиляция таких программ с ЛОСа на СИ.

Сделаем небольшое отступление, связанное с автоматическим синтезом программ. Для создания алгоритма, решающего задачу, используется группа теорем, взаимодействующих между собой в вычислительных циклах. Такие теоремы либо извлекаются из базы теорем уже готовыми, либо получаются с использованием решателя. Эти алгоритмы легко задаются в виде вычислительных пакетов ГЕНОЛОГа. В логической системе в этом формате были записаны вычислительные процедуры из различных разделов математики - теория графов, комбинаторика, вещественные и комплексные многочлены, матрицы.

Вычислительный пакет ГЕНОЛОГа, по существу, представляет собой группу теорем с приоритетами их применения. Он легко компилируется в программу на ЛОСе. Даже интерпретатор ЛОСа обеспечивает для разумных размеров входных данных почти мгновенную выдачу ответа. Однако, чтобы получить полноценные вычислительные программы, был создан компилятор с ЛОСа на СИ. По существу, он завершает цепочку синтеза программы: "Вывод теорем" - "Группировка теорем в вычислительный пакет" - "Компиляция на ЛОС" - "Компиляция на СИ".

Входным данным компилятора служит либо ссылка на вычислительный пакет ГЕНОЛОГа, где уточнены все типы входных параметров, либо ссылка на задачу на программирование, созданную в базе теорем. Главным в этой ссылке является логический символ - заголовок S компилируемого оператора либо операторного выражения ЛОСа. Сначала компилятор преобразует ЛОС-программу в своего рода "СИ - подстрочник" - набор термов ЛОСа, кодирующих операторы языка СИ. Этот подстрочник оптимизируется, затем он переписывается в файл vichprog.cpp, являющийся частью проекта интерпретатора ЛОСа. Наконец, интерпретатор перекомпилируется. Далее интепретатор ЛОСа, встречая в программе оператор с заголовком S, уже не будет обращаться к его ЛОС - программе, а сразу же выполнит соответствующую функцию СИ.

Разумеется, можно было бы компилировать ЛОС-программу не на Си, а прямо в машинные коды. Тогда можно было бы избежать паузы в работе системы, связанной с перекомпиляцией интерпретатора ЛОСа. В старых версиях решателя, еще при операционной системе DOS, такая компиляция была реализована. К сожалению, современные операционные системы, с их требованиями к безопасности, такой возможности не предоставляют.

Подробное описание данного компилятора предполагается включить в десятый том монографии. Пока лишь заметим, что посвященный ему раздел оглавления программ

может быть найден вдоль пути "Редактор ЛОСа" - "Компиляция ЛОС-программы в СИ-программу". Некоторую информацию по компилятору можно найти в справочных сведениях для символов "Си", "си".

Примеры задач на программирование, для которых тестировалась компиляция на СИ, можно найти в разделе "Программы" оглавления базы теорем. В верхней части постановки задачи располагается шаблон $f(x1...xn)$ обращения к оператору либо операторному выражению, после которого идут указатели "вход(x_i t_i)", "выход(x_j t_j)" типов значений входных и выходных переменных. Ниже идут условия, связывающие входные и выходные переменные. Эти условия рассматриваются как задача на описание, неизвестными которой являются выходные параметры, а известными - входные. Ответ задачи - совокупность утверждений, передаваемых компилятору ГЕНОЛОГа для построения ЛОС-программы символа f. Этот символ - новый, который вводится перед постановкой задачи. Наконец, ЛОС-программа компилируется в СИ-программу. Таким образом, здесь получается даже не компиляция с ЛОСа на СИ, а полноценный цикл синтеза программы по исходной системе логических условий, связывающих входные и выходные данные.

Перейти к ЛОС-программам, обеспечивающим указанные действия, можно через пункт "База теорем" - "Задачи на программирование" оглавления программ.

Автоматическое создание приемов распознавания рукописных букв по примерам

Искусственные нейросети, обучаясь на примерах, начинают хорошо распознавать изображения вообще и рукописные буквы в частности. Существенную роль в данном распознавании играет логика изображения. Поэтому естественно было бы исследовать возможности логических систем для такого распознавания. Это позволило бы лучше понимать указанную логику изображений и механизмы извлечения ее из примеров. К сожалению, даже хорошо обученная нейросеть ответов на эти вопросы не дает. Кроме того, наличие собственного распознавания, безусловно, пригодилось бы логической системе для ее обучения по внешним источникам информации.

В этом направлении, прежде всего, была предпринята попытка создать (вручную) на ГЕНОЛОГе приемы, распознающие рукописные буквы. Она описана в предыдущих томах монографии. Здесь мы лишь напомним, что рисунок представлялся в виде системы "сквозных линий", продолжаемых из соображений гладкости в точках пересечения с другими линиями. Возникал своего рода граф сквозных линий: для каждой линии указывались точки пересечения ее с другими линиями. Для отслеживания близости линий использовалась сетка - прямоугольная область рисунков разбивалась на квадраты, и для каждого квадрата приводился список проходящих через него линий. Далее рассматривались различные количественные и качественные характеристики линий, и прием ГЕНОЛОГа содержал совокупность утверждений, определяющих характеристики заданного шаблона буквы. Для ускорения проверок такие характеристики определялись даже не программами ЛОСа, а непосредственно программами на СИ.

Совокупность характеристик, достаточных для распознавания букв, при некоторых допустимых дефектах их написания, пополнялась по мере проработки обучающего материала. Таким образом был получен список характеристик, достаточных для автоматического описания образа буквы. Оставалось только извлечь из обучающих

рисунков (обычно - слов, иногда слитных) всевозможные написания одной и той же буквы и научить программу анализировать близость их по данным характеристикам. В результате возникла некоторая процедура, разбивающая список примеров на типы написаний и создающая для каждого типа прием ГЕНОЛОГа, перечисляющий характеристики графа линий этого типа. Сначала анализировались только примеры данной буквы, затем приемы прогонялись по всем обучающим рисункам и автоматически выполнялась коррекция ошибочных срабатываний.

Фактически, проработана была только буква "а". Для нее создавались 11 приемов ГЕНОЛОГа, соответствующих различным типам написания этой буквы (с учетом искажений в написании). Время, затрачиваемое на проработку примеров, крайне малое (меньше 1/2 минуты). Архитектура программы, создающей такие приемы, неоднократно изменялась, и последняя версия ее выглядит сравнительно стабильной. Работа была прервана из-за необходимости развивать прочие направления. Однако, стало понятно, что логическая система вполне способна обучаться распознаванию на примерах. В определенных ситуациях здесь используются даже возможности решателя.

Подробное описание процедуры предполагается привести в десятом томе монографии. Здесь лишь укажем раздел оглавления программ "Приемы решателя" - "Анализ рисунков" - "Работа с примерами рисунков в базе теорем", в котором содержатся ссылки на блоки данной процедуры. Обучающий материал для запуска процедуры находится в разделе базы теорем "Примеры рисунков" - "Примеры рукописных букв (кириллица)". Для запуска процедуры на букве "а"достаточно выделить пункт "А" и нажать клавишу "а". После остановки на контрольной точке отладчика ЛОСа нужно выйти главное меню, и в разделе "Генератор приемов" базы приемов найти новые приемы. Заметим, что при этом автоматически отключаются старые приемы, созданные вручную. Чтобы они снова заработали, нужно из просмотра любой задачи на анализ рисунка нажать "т" и выбрать "Режим новой версии", чтобы "+" исчез. Затем, чтобы изменения произошли, выходить из выбора режима нужно через Enter. Удалять новые приемы из буфера можно обычным образом - через "О".

Продолжение обучения текстового анализатора

Архитектура процедуры текстового анализатора и многие ее приемы были описаны в предыдущих томах монографии. Развитие этой процедуры постоянно продолжается. Во-первых, различные вставки в программу разбиения слова на корень, суффикс и окончание, которые приходилось делать, если корень определял слово неоднозначно, были вынесены в специальные справочники. Эти справочники усматривают нереализуемые группы фрагментов слова и уточняют слово с заданным корнем по суффиксу и окончанию, а иногда и по его окружению в фразе. Именно, созданы справочники "словпроверка", "словфильтр", "словкорень" и "словстоп". Описания их можно найти в справочной информации о соответствующих логических символах. Данные справочники существенно упростили добавление новых слов, корень которых уже встречался в другом слове.

Во-вторых, продолжалось развитие синтаксического анализа. Для этого использовались достаточно сложно устроенные фразы из различных источников. За несколько последних лет число таких фраз увеличилось с 106 до 236. В основном, проблему составляли новые обороты, изменявшие всю схему перевода фразы в логический подстрочник. Связанные с ними коррекции приемов синтаксического анализа были несложны, но их было много. Работа велась эпизодически, что и объясняет весьма

умеренное увеличение числа проработанных фраз. Впрочем, коррекции хорошо укладывались в уже имеющуюся процедуру, и после нескольких коррекций программа выдавала готовый логический подстрочник всей фразы. Складывается впечатление, что обучение системы синтаксическому анализу можно будет завершить и вручную. Что касается семантического анализа, то здесь, конечно, создание приемов нужно будет автоматизировать. Пока можно говорить лишь о накоплении достаточного количества таких приемов, созданных вручную, которое могло бы составить обучающий материал для автоматизации их синтеза.

Работа по развитию текстового анализатора, конечно, предстоит еще большая, но она того стоит. В результате логическая система будет способна получать точный логический перевод прочитанных текстов, а не какой-то туманный "смысл", основанный на интуитивных связях между словами.

Предварительная обработка изображений

Роль логики при анализе изображений достаточно велика. В условиях плохой видимости она позволяет воссоздавать картину по мелким деталям. Этот процесс естественно было бы осуществлять с помощью логической системы. Да и самой логической системе возможность анализа изображений дала бы важный источник дополнительной информации. Речь идет не о конкуренции с нейросетями, а о изучении логики изображений, каковая остается загадкой не только для искусственных, но даже для естественных нейросетей.

Пока удалось реализовать лишь предварительную обработку изображения, целью которой является переход от битмэпа с его сотнями тысяч пикселей к сравнительно маломерной структуре - графу цветных пятен, насчитывающему лишь сотни элементов. После объединения таких пятен в объекты, даже без какого-либо распознавания, используя лишь общие соображения близости цветов и гладкости границы, возникают уже лишь десятки элементов, и эти данные можно было бы далее обрабатывать на уровне логической системы. Вся указанная предобработка реализована на СИ как часть интерпретатора ЛОСа. Работает достаточно быстро.

Посмотреть, как это выглядит, можно в разделе задачника "Изображения". Обучение происходило на 10 различных фотографиях. Как обычно, для запуска "задачи" нажимается "o". Красными линиями прорисовываются границы объектов. Если после этого еще раз нажать "o", будут выделяться отдельные объекты. Клавишами "курсор вверх" - "курсор вниз" можно переходить от одного объекта к другому. Впрочем, предобработка пока не доведена до конца: остается учет геометрии границ цветных пятен.

Подробное описание процедуры предобработки предполагается привести в десятом томе монографии. В справочной информации логического символа "изображение" приводится список операторов ЛОСа "изображение(s ...)", выполняющих различные действия в конвейере предобработки либо в интерфейсе обучения. В разделе "Анализ изображений" оглавления программ содержатся ссылки на фрагменты ЛОС-программ, реализующие предобработку. В разделе справочника по системе "Программирование на ЛОСе" - "Интерпретатор ЛОСа" - "Основные структуры данных интерпретатора" содержится описание различных структур данных, используемых при анализе изображений.

Глава 27

Выводы

Работа над компьютерным решателем задач продолжается вот уже почти 40 лет. За это время было проработано множество различных предметных областей. Процессы решения задач в них разбивались на простейшие шаги - так называемые приемы. Каждый прием представлял собой, по существу, некоторую теорему, снабженную управлением. Управление уточнялось в процессе проработки потока задач. Была предпринята классификация приемов по типам целей, ради которых используется теорема, и это позволило перейти к автоматическому созданию приемов. Управление теоремой формировалось как на основе логических характеристик самой теоремы, так и на основе учета ее возможных взаимодействий с другими теоремами. Элементы этого управления подсказывались возникавшими при обучении конфликтами между приемами. Анализировались источники теорем, лежащих в основе приемов, и процесс получения таких теорем из "базисных" теорем учебников тоже был разложен на приемы.

Все это однозначно подтвердило возможность расшифровки механизмов управления рассуждениями при решении задач, а также механизмов самообучения решателей, и реализации их в виде компьютерных программ. Для выполнения этих программ, очевидно, не нужны никакие нейросети - гораздо эффективнее они выполняются на обычном процессоре.

Все-таки, главное свойство интеллекта - это способность рассуждать, способность решать задачи. Искусственные нейросети пока не очень хорошо владеют этой способностью. Разумеется, в конечном счете они ею овладеют. НО. Решая задачи, они лишь будут применять все те же приемы, что и решатель - приемы ведь определяются логикой предметной области, а она одна и та же и у решателя, и у естественной нейросети, и у искусственной. Фактически, нейросеть будет попросту моделировать работу логической системы, сильно проигрывая ей в эффективности. То же самое относится и к самообучению - логика развития теорий и алгоритмизации теорем никак не связана с выбором вычислительного устройства.

Наконец, даже такие направления, как понимание естественного языка и понимание изображений, где пока искусственные нейросети далеко ушли вперед, вовсе не являются чем-то недоступным для логических систем. Трудность здесь лишь в том, что детальной анализ этих процессов будет длительным, но при понимании их алгоритмов опять-таки логическая система может оказаться более эффективной.

Все это заставляет предположить, что искусственный интеллект будущего - это все-таки не искусственные нейросети, а мощные логические вычислители, реализованные

на специальных логических процессорах. Такие вычислители, в отличие от нейросетей, абсолютно прозрачны, не имеют каких-либо личных интересов, а представляют собой лишь логический инструмент, позволяющий человеку решать более сложные задачи.

Список литературы

1. А.Черч. Введение в математическую логику. Том 1. М.: ИЛ, 1960, с.484.
2. С.К.Клини. Введение в метаматематику. М.: ИЛ, 1957. 526с.
3. С.К.Клини. Математическая логика. М.: Мир, 1973, 480с.
4. Д. Пойа. Математика и правдоподобные рассуждения. М.: Наука, 1975, 463с.
5. Д. Пойа. Математическое открытие. М.: Наука, 1976, 448с.
6. Подколзин А.С. Об организации баз знаний, ориентированных на автоматическое решение задач. "Дискретная математика", 1990, т.2., вып.1., с. 13-30.
7. Подколзин А.С. Система автоматического решения задач по элементарной алгебре. "Дискретная математика", 1994, т.6., вып.4., с. 35-57.
8. Подколзин А.С. Компьютерный решатель математических задач. ДАН РФ, 1994, т.335, № 4.
9. Подколзин А.С. Компьютерное моделирование процессов решения математических задач. Изд-во ЦПИ при мех.-мат. факультете МГУ, 2001. 235 с.
10. Подколзин А.С. Компьютерное моделирование логических процессов. Том 1. Архитектура и языки решателя задач. М., "Физматлит", 2008. 1022 с.
11. Подколзин А.С. О самообучении интеллектуальной системы. "Интеллектуальные системы", 2014, том 18, выпуск 2, с. 197 - 266.
12. Подколзин А.С. Компьютерное моделирование логических процессов. Том 2. Опыт обучения компьютерного решателя задач: логические приемы, алгебра множеств, комбинаторика и элементарная алгебра. МГУ. - М., 2015. 1153 с. Деп. в ВИНИТИ РАН 09.11.2015, № 184-В2015.
13. Подколзин А.С. Компьютерное моделирование логических процессов. Том 3. Опыт обучения компьютерного решателя задач: математический анализ, дифференциальные уравнения и элементарная геометрия. МГУ. - М., 2015. 1320 с. Деп. в ВИНИТИ РАН 09.11.2015, № 185-В2015.
14. Подколзин А.С. Компьютерное моделирование логических процессов. Том 4. Опыт обучения компьютерного решателя задач: аналитическая геометрия, линейная алгебра, теория вероятностей, комплексный анализ и другие разделы. МГУ. - М., 2017. 969с. Деп. в ВИНИТИ РАН 27.02.2017, № 18-В2017.
15. Подколзин А.С. Компьютерное моделирование логических процессов. Том 5. Опыт обучения компьютерного решателя задач: Элементарные физика и химия, шахматы. МГУ. - М., 2019. 939с. Деп. в ВИНИТИ РАН 12.08.2019, № 66-В2019.

16. Подколзин А.С. Компьютерное моделирование логических процессов. Том 6. Опыт обучения компьютерного решателя задач: Понимание естественного языка и анализ рисунков. МГУ. - М., 2019. 758с. Деп. в ВИНИТИ РАН 12.08.2019, № 67-В2019.

17. Подколзин А.С. Компьютерное моделирование логических процессов. Том 7. Автоматическое создание приемов логической системы: классификация приемов решателя; логический ассемблер; компилятор спецификаций; создание тестовых примеров и доводка приемов. МГУ. - М., 2021. 739с. Деп. в ВИНИТИ РАН 06.12.2021, № 65-В2021.

18. Подколзин А.С. Компьютерное моделирование логических процессов. Том 8. Автоматическое создание приемов логической системы: база теорем, характеризация теорем, создание спецификаций приемов. МГУ. - М., 2021. 518с. Деп. в ВИНИТИ РАН 06.12.2021, № 66-В2021.

19. Подколзин А.С. Компьютерное моделирование логических процессов. Том 9. Автоматическое создание приемов логической системы: логический вывод в базе теорем. МГУ. - М., 2022. 1500с. Деп. в ВИНИТИ РАН 24.10.2022, № 33-В2022.