Московский Государственный Университет
имени М.В. Ломоносова
Российская Академия Наук
Международная Академия Технологических Наук
Российская Академия Естественных Наук

Интеллектуальные Системы.

Теория и приложения

ТОМ **29** ВЫПУСК 2 * **2025**

МОСКВА

УДК 519.95; 007:159.955 ISSN 2411–4448
ББК 32.81 Издаётся с 1996 г.

В журнале «Интеллектуальные системы. Теория и приложения» публикуются научные достижения в области теории и приложений интеллектуальных систем, новых информационных технологий и компьютерных наук.
Издание журнала осуществляется под эгидой МГУ имени М.В. Ломоносова, Научного Совета по комплексной проблеме «Кибернетика» РАН, Отделения «Математическое моделирование технологических процессов» МАТН.
Учредитель журнала: ООО «Интеллектуальные системы».
Журнал входит в список изданий, включенных ВАК РФ в реестр публикаций материалов по кандидатским и докторским диссертациям по математике и механике.

Индекс подписки на журнал: 64559 в каталоге НТИ «Роспечать».

Адрес редакции: 119991, Москва, ГСП-1, Ленинские Горы, д. 1, механико-математический факультет, комн. 12-01.
Адрес издателя: 115230, Россия, Москва, Хлебозаводский проезд, д. 7, стр. 9, офис 9. Тел. +7 (495) 939-46-37, e-mail: mail@intsysmagazine.ru

*) Прежнее название журнала: «Интеллектуальные системы».

ОГЛАВЛЕНИЕ

Часть 1
Общие проблемы теории интеллектуальных систем

Введение в логические процессы. Представление задач в решателе

А. С. Подколзин[1]

В статье описывается интерфейс решателя математических задач; рассказывается о логической системе "Искра"; описывается логический язык, используемый в решателе; рассказывается как осуществляется логическая формализация задач.

Ключевые слова: решатель математических задач, логические процессы, логический язык, логическая формализация задач.

Введение

Данная статья начинает цикл статей, посвященных практикуму по решателю математических задач. Решатель и заложенные а него приципы подробно описаны в монографиях [1, 2, 3, 4, 5, 6, 7, 8, 9].

Данный цикл статей посвящен 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

[1] *Подколзин Александр Сергеевич* — д.ф.м.н., профессор каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: alexander.p@yandex.ru.

Podkolzin Alexander Sergeevich — Dr. of Sc., Professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

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 челевека 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 шагам" 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 распределены 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 последова-

тельно 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.

Диаграмма, на которой представлены основные этапы и основные “действующие лица” изложенного процесса алгоритмизации теорем, приведена на рисунке 1.

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.

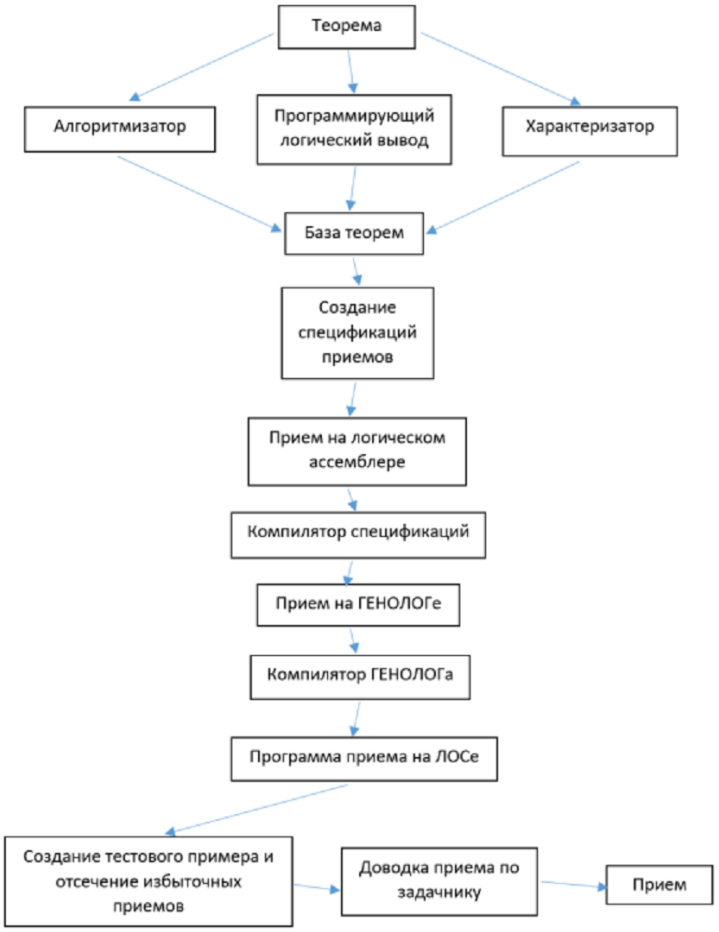

Рис. 1.

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

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 OK, “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 соответствующи-

ми 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:

а) 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.

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

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

г) Редактор ГЕНОЛОГа. Программы просмотра приемов ГЕНОЛОГа и их редактирования.

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

е) 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).

ж) Синтез приемов. В этом разделе содержатся:
   - 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).
   - Программа, позволяющая просмотреть все приемы заданного типа, имеющиеся в системе. Она запускается из просмотра концевогоо пункта оглавления типов приемов нажатием клавиши "л".

– Программа, обновляющая список ссылок на приемы заданных типов путем полного сканирования базы приемов.
– Программа компилятора спецификаций, создающая описание приема на ГЕНОЛОГе по его спецификации. Обращение к ней происходит из программ, связанных с концевыми пунктами оглавления типов приемов.
– 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.
– 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.

з) 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.

и) 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.

к) 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.

л) 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.

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

н) 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.

о) 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.

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

7) Ресурсы и установки (р)

Вход в подменю главного меню, позволяющее выполнять следующие действия:

а) Просматривать логические символы системы, упорядоченные по номерам.

б) Просматривать логические символы системы, упорядоченные по алфавиту.

в) Вводить названия новых символов, удалять либо изменять старые названия.

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

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

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

ж) Выбирать размеры окна системы.

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

и) Редактировать контекстные меню, извлекаемые по F1.

к) Изменять шрифты системы.

л) Выбирать параметры распараллеливания при прокрутках.

8) Уплотнение измененных файлов (y). 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 (x). 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 можно 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 кририллицу 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.

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 $S_1 \dots S_n$)”, где A — название раздела; $S_1, \dots, S_n$ — 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 названия 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 $f(t_1 \dots t_n)$ 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 $f(t_1 \dots t_n)$. 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 $t_1, \ldots, t_n$ вместо переменных $x_1, \ldots, x_n$ 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 утвер-

ждений и выражений, а также о способах ввода термов формульным редактором будет сказано далее.

Упражнения:

1) Найти все свободные и все связанные переменные утверждения
   $\forall_x(\exists_y(P(x,y,z)) \to Q(x,v))$

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 $s_1 \ldots s_n$)", 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 & $\dots$ & 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 бесквантворных 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 “$\mathrm{set}_{x_1 \dots x_n} A$”, описатель “отображение” — вид “$\lambda_{x_1 \dots x_n}(t, A)$”. 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 формульной 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($A_1 \dots A_n$)”, “пересечение($A_1 \dots A_n$)”, “разность(A B)”, “прямоепроизведение($A_1 \dots A_n$)” используются для обозначения простейших операций над множествами. Формульная прорисовка обычная: $\emptyset$, “$A_1 \cup \dots \cup A_n$”, “$A_1 \cap \dots \cap A_n$”, “$A \setminus B$”, “$A_1 \times \dots \times A_n$”.

Для задания конечной последовательности элементов $A_1, \dots, A_n$ используется выражение “набор($A_1 \dots A_n$)”. 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 ... 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. Формульная 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); “взаимноодно-значно(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 — “$a \;\to\; b$”. “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 доопределения 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 взаимно-однозначное 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 рассматриваемую 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 “внутркасаются(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 $K = (k_1 \ldots k_n)$ — 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 "$\forall_{x_1 \ldots x_k}(A_1 \& \ldots \& A_n \to A_0)$" сначала дважды нажимается большая латинская буква A и появляется знак квантора общности. Затем перечисляются (без каких-либо пробелов)переменные $x_1, \ldots, x_k$ кванторной приставки; нажимается "Enter" (здесь появляется

открывающая скобка), после чего вводится конъюнкция $A_1\&\ldots\&A_n$. Далее нажимается "курсор вправо" (появляется стрелка для "если-то"), вводится утверждение A_0, и в конце ставится закрывающая скобка.

Для ввода кванторной записи "$\exists_{x_1\ldots x_k}(A)$" сначала дважды нажимается большая латинская буква E и появляется знак квантора существования. Затем набирается кванторная приставка $x_1\ldots x_k$; 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 (е-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 взаимно-однозначной 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

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

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 “M”, “a”; “M”, “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 “E”, “x” (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 “c”, “ц”; “o”, “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)”, “разрывторогорода(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), “разныесторо-ны(A, B, a)” (здесь тоже допускается попадание точек на a), “симметрич-ны(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”.

Утверждения “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) $d(\sqrt{\mathrm{tg}(1/a + a) + 1}/da$
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) прямая$(AB) \parallel$ прямая(CD)

16) прямая$(AB) \perp$ прямая(CD)

17) $x = S(\text{фигура}(ABCD))$

18) внешкасаются(окружность(AC), окружность(BC))

19) $(x + 1)dy(x)/dx + xy(x) = 0$

20) коорд(прямая$(AB), K) = \mathrm{set}_{xy}(y - \text{число}\ \&\ y = x/2\ \&\ x - \text{число})$

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.

3.1.1. 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, назы-

ваемый 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($x_{j_1} t_1$) $\ldots$ 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 $y_1 \dots y_k$”, где $y_1, \dots, y_k$ — 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 $y_i; i = 1, \dots, k$. Переменные $y_1, \dots, y_k$, 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 неизвестных 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 $x_1, \dots, x_k$).

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 $x_1 \dots x_k$”, перенесенная из задачи на описание и направляющая вывод следствий таким образом, чтобы увеличить “степень разрешенности” утверждений относительно переменных $x_1, \dots, x_k$. 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 $(\text{класс}(x\ \sqrt{\sqrt{x+\frac{1}{4}}+x+\frac{1}{2}}=a))$”. Ответ, получаемый здесь решателем, имеет вид “1, если $\frac{1}{4}\le a$, 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 $64ab(a+b)^2\le(\sqrt{a}+\sqrt{b})^8$ и посылками “число(a)”, “число(b)”. Неот-

рицательность 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: $\mathrm{card}(\mathrm{set}_x(\exists_{BC}(x = \{B, C\}\ \&\ B - \mathrm{set}\ \&\ C - \mathrm{set}\ \&\ A = B \cup C\ \&\ B \cap C = \emptyset\ \&\ \mathrm{card}(B) = \mathrm{card}(C)\ \&\ \mathrm{card}(B \cap D) = 2\ \&\ \mathrm{card}(C \cap D) = 2)))$. 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 $\neg(\text{слой}(x, m) = \emptyset)$; условие различия цифр числа — “взаимнооднозначно(x)”. Таким образом, условие задачи на преобразование имеет вид
$\mathrm{card}(\mathrm{set}_f(\text{Отображение}(f, \{1, \ldots, 5\}, \{1, \ldots, 9\})\ \&$
$\text{взаимнооднозначно}(f)\ \&\ \neg(\text{слой}(f, 2) = \emptyset)\ \&\ \neg(\text{слой}(f, 4) = \emptyset)\ \&$
$\neg(\text{слой}(f, 5) = \emptyset)))$.

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 $a_1 \dots a_n$”, которая указывает все параметры $a_1, \dots, a_n$, 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 $\forall_x$ (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 "суп(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(λ_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 $x = \lambda_y(-y^3 + 3y, \text{число}(y))$. 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 $x = \lambda_y(-y^3+3y^2, \text{число}(y))$ 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$(y)\ \ 0 \le x \quad 0 \le y \quad x + y \le 2\pi))\ u\ v)$”, 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 $0 < p, \text{число}(p)$.

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) $k \leq n$) $a(n)$))”, 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 $\exists_C(x\ln(Cy(x)) - y(x) = 0 \ \&\ \text{число}(C)) \vee y(x) = 0$.

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:
$\exists_C\left(y(-\frac{C+3t}{\sqrt{C+2t}}) = -\frac{t^2}{\sqrt{C+2t}}\ \&\text{число}(C)\right) \vee$
$\vee\ \exists C\left(y(\frac{C+3t}{\sqrt{C+2t}}) = \frac{t^2}{\sqrt{C+2t}}\ \&\ \text{число}(C)\right) \vee y(x) = 0.$

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 установка 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)фигу-

ра(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 имеет 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 ви-

да “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 пере-

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

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 $x_1 \dots x_n$”, “параметры $y_1 \dots y_m$”. Здесь $y_1, \dots, y_m$ — 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-o".

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 ест-

ественной, 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” (“o” — 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 какая-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, соответствующая 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 “c” либо после выделения ранее набранного элемента нажимается “c”.

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.

Список литературы

[1] А.С.Подколзин, *Компьютерное моделирование логических процессов. Том 1. Архитектура и языки решателя задач.*, Физматлит, Москва, 2008, 1024 с.

[2] А.С.Подколзин, *Компьютерное моделирование логических процессов. Том 2. Опыт обучения компьютерного решателя задач: логические приемы, алгебра множеств, комбинаторика и элементарная алгебра.*, ВИНИТИ РАН, Москва, 2015, 1153 с.

[3] А.С.Подколзин, *Компьютерное моделирование логических процессов. Том 3. Опыт обучения компьютерного решателя задач: математический анализ, дифференциальные уравнения и элементарная геометрия.*, ВИНИТИ РАН, Москва, 2015, 1320 с.

[4] А.С.Подколзин, *Компьютерное моделирование логических процессов. Том 4. Опыт обучения компьютерного решателя задач: аналитическая геометрия, линейная алгебра, теория вероятностей, комплексный анализ и другие разделы.*, ВИНИТИ РАН, Москва, 2017, 969 с.

[5] А.С.Подколзин, *Компьютерное моделирование логических процессов. Том 5. Опыт обучения компьютерного решателя задач: элементарные физика и химия, шахматы.*, ВИНИТИ РАН, Москва, 2019, 938 с.

[6] А.С.Подколзин, *Компьютерное моделирование логических процессов. Том 6. Опыт обучения компьютерного решателя задач: понимание естественного языка и анализ рисунков.*, ВИНИТИ РАН, Москва, 2019, 757 с.

[7] А.С.Подколзин, *Компьютерное моделирование логических процессов. Том 7. Автоматическое создание приемов логической системы: классификация приемов решателя, логический ассемблер, компилятор спецификаций, создание тестовых приемов и доводка приемов.*, ВИНИТИ РАН, Москва, 2021, 739 с.

[8] А.С.Подколзин, *Компьютерное моделирование логических процессов. Том 8. Автоматическое создание приемов логической системы: база теорем, характеризация теорем, создание спецификаций приемов.*, ВИНИТИ РАН, Москва, 2021, 515 с.

[9] А.С.Подколзин, *Компьютерное моделирование логических процессов. Том 9. Автоматическое создание приемов логической системы: логический вывод в базе теорем.*, ВИНИТИ РАН, Москва, 2022, 1494 с.

Introduction to Logical Processes. Representation of Problems in the Solver

Podkolzin A.S.

The article describes the interface of the mathematical problem solver; tells about the logical system "Iskra"; describes the logical language used in the solver; tells how the logical formalization of problems is carried out.

Keywords: mathematical problem solver, logical processes, logical language, logical formalization of problems.

References

[1] A. S. Podkolzin, *Computer modeling of logical processes. Volume 1. Architecture and languages of the problem solver.*, Fizmatlit, Moscow, 2008, 1024 pp.

[2] A. S. Podkolzin, *Computer modeling of logical processes. Volume 2. Experience in training a computer problem solver: logical techniques, set algebra, combinatorics and elementary algebra.*, VINITI RAS, Moscow, 2015, 1153 pp.

[3] A. S. Podkolzin, *Computer modeling of logical processes. Volume 3. Experience in teaching computer problem solver: mathematical analysis, differential equations and elementary geometry.*, VINITI RAS, Moscow, 2015, 1320 pp.

[4] A. S. Podkolzin, *Computer modeling of logical processes. Volume 4. Experience in teaching computer problem solver: analytical geometry, linear algebra, probability theory, complex analysis and other topics.*, VINITI RAS, Moscow, 2017, 969 pp.

[5] A. S. Podkolzin, *Computer modeling of logical processes. Volume 5. Experience in teaching a computer problem solver: elementary physics and chemistry, chess.*, VINITI RAS, Moscow, 2019, 938 pp.

[6] A. S. Podkolzin, *Computer modeling of logical processes. Volume 6. Experience in training a computer problem solver: natural language understanding and image analysis.*, VINITI RAS, Moscow, 2019, 757 pp.

[7] A. S. Podkolzin, *Computer modeling of logical processes. Volume 7. Automatic creation of logic system techniques: classification of solver techniques, logic assembler, specification compiler, creation of test techniques and refinement of techniques.*, VINITI RAS, Moscow, 2021, 739 pp.

[8] A. S. Podkolzin, *Computer modeling of logical processes. Volume 8. Automatic creation of logical system techniques: theorem base, characterization of theorems, creation of technique specifications.*, VINITI RAS, Moscow, 2021, 515 pp.

[9] A. S. Podkolzin, *Computer modeling of logical processes. Volume 9. Automatic creation of logical system techniques: logical inference in the theorem base.*, VINITI RAS, Moscow, 2022, 1494 pp.

Часть 2
Специальные вопросы теории интеллектуальных систем

Изменение формы КАМ созвездий на стороне передатчика в беспроводном канале с фиксированным алгоритмом декодирования.

Д. А. Юдаков[1]

В работе исследуется оптимизация геометрии сигнальных созвездий для повышения пропускной способности в беспроводных системах связи. Рассматривается подход, при котором созвездия могут быть неравномерно распределены для улучшения эффективности передачи. Оптимизация выполняется на стороне базовой станции, а ее эффективность подтверждается численными экспериментами с использованием LDPC кодирования, OFDM модуляции и технологии MIMO.

Ключевые слова: Беспроводная связь, Сигнальное созвездие, Изменение формы созвездий, Взаимная информация.

1. Введение

На производительность систем связи большое влияние оказывает процесс модуляции, в том числе выбор схемы модуляционного кодирования (СМК). Выбор СМК сопровождается выбором кодовой скорости и созвездия. Основным подходом в системах беспроводной связи является использование квадратурной амплитудной модуляции (КАМ, Quadrature Amplitude Modulation, QAM) [6]. Во многих статьях предлагается изменить часть передачи сигнала, относящуюся к созвездиям [3, 8, 10, 7]. В созвездиях можно оптимизировать как геометрию созвездия (путём изменения координат точек созвездия), так и вероятности точек. В англоязычной литературе эти подходы известны как geometric shaping и probabilistic shaping, соответственно. Геометрия созвездия обычно фиксирована, например, используется КАМ. Идея работы состоит в том, чтобы изменить координаты этого созвездия на стороне базовой станции во время передачи по нисходящей линии связи.

В оптической и беспроводной связи было предложено несколько геометрий созвездий, которые улучшают пропускную способность канала,

[1] *Юдаков Даниил Андреевич* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: d.yudakov43@gmail.com.

Yudakov Daniil Andreevich — graduate student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

учитывая входное распределение сигнала. В [13] созвездия изменяются для беспроводных каналов обратной связи, основываясь на показатели частоты ошибок в кадрах, с использованием мягкого и жесткого декодирования с использованием кодов WiMAX и DVB-S2 LDPC. Как вероятностные, так и геометрические методы формирования показывают значительные преимущества по сравнению с равномерно распределенной передачей символов.

В этой работе мы сосредоточимся на оптимизации геометрии созвездий. Этот подход хорошо изучен. Например, созвездие из 256 точек в [15] обеспечивает усиление отношения сигнала к шуму (ОСШ) до 1,18 дБ по сравнению со стандартными созвездиями КАМ. В [3] предложены форматы модуляции с применением созвездий на основе решётки в многомерном евклидовом пространстве, а также описаны быстрые алгоритмы модуляции и демодуляции с низкой сложностью. Для оптимизации геометрии созвездий часто используются методы машинного обучения [1]. В [14] для получения новых созвездий используется автоэнкодер.

Во всех похожих статьях на эту тему предполагается, что мы можем изменять алгоритм модуляции и демодуляции на стороне передатчика и приемника соответственно. Это невозможно реализовать в существующих стандартах систем беспроводной связи. Передача сигнала в беспроводных сетях осуществляется от базовой станции к пользовательскому оборудованию (ПО). Большинство принимающих устройств работают в соответствии с уже установленными стандартами, например, 3GPP TS 38.211 V15.4.0 [5]. Изменить процедуры модуляции (для восходящего канала) и демодуляции (для нисходящего канала) доволно сложно, поэтому нужно изменять сами стандарты, что является довольно сложным процессом.

В данной статье мы рассматриваем тип передачи по нисходящему каналу. Для нисходящего канала базовая станция преобразует последовательность битов в последовательность комплексных чисел с использованием неоднородного созвездия. Пользователь получает этот сигнал и преобразует его в последовательность отношений правдоподобия битов (Likelihood Ratio, LLR) при помощи стандартного КАМ созвездия. И в конечном итоге последовательность LLR-ов преобразуется в последовательность битов. Весь процесс передачи может быть реализован с использованием существующей технологии и не требует изменения стандарта 3GPP. Мы рассматриваем реалистичные коды с малой плотностью проверки чётности (Low-Density Parity-Check, LDPC) и систему OFDM 5G MIMO, проводя численные эксперименты в симуляторе Sionna на уровне физической системы связи [2].

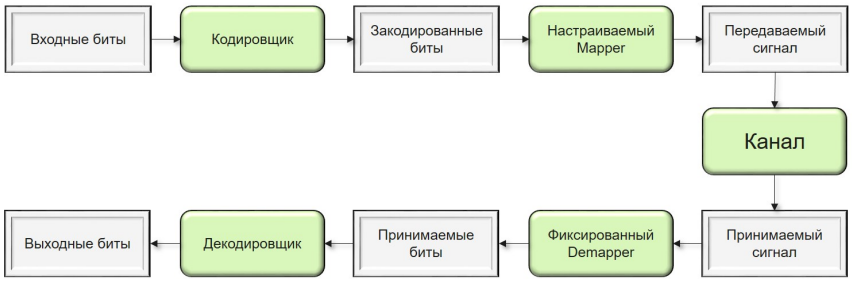

Рис. 1. Схема передачи сигнала с фиксированной процедурой декодирования на стороне пользователя.

2. Модель системы

Рассмотрим реалистичную передачу данных "точка-точка" между пользователем и базовой станцией. Система, которую мы будем настраивать, показана на рисунке. 1. Основным отличием от модели кодированной модуляции с перемешиванием битов (bit-interleaved coded modulation, BICM) [16] является изменение геометрии созвездия на стороне передатчика в низходящем канале и на стороне приёмника в восходящем канале.

В рассматриваемой модели канала на стороне базовой станции к равномерно распределенной последовательности битов (Входные биты) добавляются проверочные биты с помощью LDPC кодирования. Затем последовательность битов преобразуется в аналоговый сигнал, которому соответствует последовательность комплексных значений, обозначающих амплитуду и фазу передаваемого сигнала (Настраиваемый Mapper). Во время процедуры преобразования вся двоичная последовательность делится на части длиной m, и каждая часть преобразуются из битовой последовательности в комплексное число $\mu : \{0,1\}^m \to \mathbb{C}$. Это отображение μ обычно представляется в виде созвездия. Пример созвездия можно увидеть на рис. 4. Полученная последовательность комплексных чисел преобразуется в аналоговый сигнал, проходит через канал с аддитивным белым гауссовым шумом (АБГШ), попадает на приёмник и преобразуется обратно в цифровой вид.

На стороне приёмника линейный эквалайзер минимальной среднеквадратичной ошибки (Linear Minimum Mean Squared Error, LMMSE) уменьшает межсимвольные помехи от принимаемого сигнала. Для этого комплексные значения преобразуются в логарифмические отношения правдоподобия (Likelihood Ratio, LLR) битов, а декодер, наконец, преоб-

разует последовательность LLR в последовательность битов. Качество принимаемого сигнала может быть описано функциями взаимной информации между последовательностью входных символов и последовательностью LLR.

Наша цель - оптимизировать модулирующую часть следующей схемы передачи сигнала:

$$s \xrightarrow{Mapper} x \xrightarrow{Channel} Hx+n \xrightarrow{Equalizer} G(Hx+n) = y \xrightarrow{Demapper} llr \xrightarrow{Decoder} \tilde{s}$$

где s и $\tilde{s}$ — передаваемая и принимаемая последовательность битов, x и y — передаваемая и принимаемая последовательность комплексных чисел, H, G — предрасчитанные комплексные матрицы канала и LMMSE эквалайзера соответственно; обычно можно подразумевать, что $GH \sim I$ (в этом случае сигнал не меняется при нулевом шуме), n — рандомный гауссовский шум (т.е. $\mathcal{N}(0, \sigma^2)$), llr — последовательность действительных чисел, полученная при помощи функций логарифмического отношения правдоподобия:

$$llr_i(y) = \log_2 \frac{P(b_i(x) = 0|y)}{P(b_i(x) = 1|y)} = \log_2 \left(\frac{\sum_{x \in \mathcal{X}_0^i} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right)}{\sum_{x \in \mathcal{X}_1^i} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right)} \right) \quad (1)$$

где $\mathcal{X}_0^i$ и $\mathcal{X}_1^i$ это непересекающиеся множества точек созвездия.
Для $x \in \mathcal{X}_0^i$ бит i равен 0, а для $x \in \mathcal{X}_1^i$ бит i равен 1.

Заметим, что в случае фиксированной процедуры демодуляции, LLR зависит от созвездия приемника. Для нисходящей передачи этим созвездием является KAM, которое берётся из стандарта 3GPP (и мы имеем зависимость $llr_i(x, y)$, где x это координаты KAM созвездия, а y это принимаемое комплексное число). Реальные и мнимые координаты точек KAM созвездия можно определить как декартово произведение двух равномерно распределённых на прямой множеств точек.

2.1. Взаимная информация как функция потерь

Для произвольного канала от $X \to Y$ и входного распределения $P_X(x)$ мы рассматриваем ансамбль (X, Y) с распределением $P_{(X,Y)}(x, y)$. Мерой информации в этом случае является взаимная информация $I(X;Y)$:

$$I(X;Y) := H(X) - H(X|Y) = \mathbb{E}_{(X,Y)} \left(\log_2 \frac{P_{(X,Y)}(x,y)}{P_X(x)P_Y(y)} \right), \quad (2)$$

где $H(X)$ — информационная энтропия входной последовательности комплексных чисел и $H(X|Y)$ — условная энтропия.

Математическое ожидание может быть вычислено в виде суммы или интеграла, в зависимости от того, является ли распределение дискретным или непрерывным. Согласно теореме Шеннона [4], $I(X;Y)$ — это строгая верхняя граница пропускной способности канала при использовании оптимальных кодов. Современные LDPC-коды из стандарта [5] позволяют приблизится к этому пределу с предсказуемыми средними потерями в скорости передачи сигнала.

В этой работе мы хотим сосредоточиться на увеличении взаимной информации BICM, которая была рассчитана в [16]:

$$I^{BICM}(X;Y) = \sum_{j=1}^{m} \mathbb{E}_{Y,B} \left(\log_2 \frac{\sum_{x \in \mathcal{X}_B^j} P_{Y|X}(y|x))}{\frac{1}{2} \sum_{x \in \mathcal{X}} P_{Y|X}(y|x))} \right), \tag{3}$$

где m — количество битов, соответствующее одной точке созвездия, Y — непрерывная случайная величина, сопоставляемая принятой точке на комплексной плоскости, B — зависимая от Y случайная величина, показывающая какой j-й бит передавался, $\mathcal{X}$ — все точки созвездия, $\mathcal{X}_B^j$ — все точки созвездия, в которых j-й бит равен B.

Наш подход предполагает изменение созвездия только на стороне пользователя. Для нисходящей передачи данное созвездие отвечает за распределение сигнала, принимаемого пользователем от базовой станции $P_Y(y)$. Изменение этого созвездия влечет за собой изменение функции взаимной информации. С другой стороны, созвездие КАМ, предназначенное для демодуляции сигнала на стороне приёмника, фиксировано.

Мы можем объяснить факт изменения взаимной информации следующим образом: y - это сигнал, полученный пользователем после прохождения по каналу. Распределение $P_Y(y)$ описывает вероятность обнаружения данного сигнала в определенной области. Для канала АБГШ это распределение представлено в виде облаков шума вокруг точек созвездия. Таким образом, если мы изменяем передаваемые точки созвездия, то изменяется само распределение $P_Y(y)$ (см. рис. 2). Перемещение одной точки созвездия приводит к увеличению помех для других точек и уменьшению для других. Функция взаимной информации помогает нам правильно выбрать созвездие. Отображенные точки созвездия показывают ожидаемый сигнал.

Пользователь будет декодировать сигнал с учетом точек КАМ. Это означает, что приёмник подразумевает, что испускаемый сигнал имеет не изменённую геометрию. Итак, чтобы минимизировать функцию взаимной информации, нам нужно максимизировать (3) с фиксированными множествами $\mathcal{X}_B^j$.

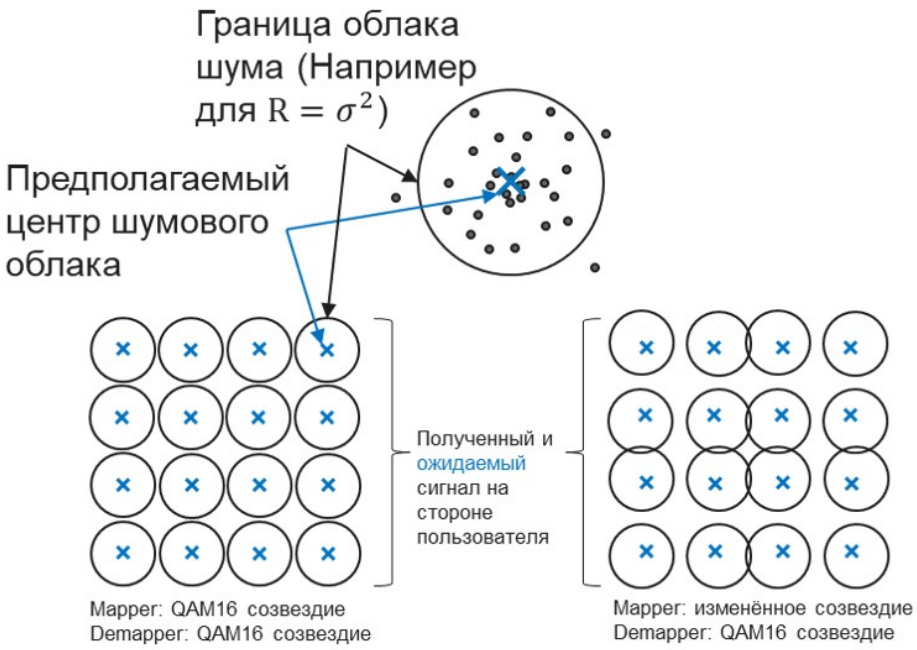

Рис. 2. Иллюстрация изменений на стороне пользователя. Здесь пользователь считает, что принятый сигнал распределен как гауссовский шум вокруг точек созвездия КАМ (вокруг синего креста). Процедура декодирования вычисляет LLR на основе этого предположения. Но реальный сигнал имеет разное распределение для разных созвездий. На рисунках эти распределения представлены в виде окружностей. Можно сказать, что математическое ожидание каждой группы полученных точек, связанных с фиксированной точкой созвездия, будет находиться в центре некоторой окружности. Радиус R этой окружности показывает дисперсию этого шума. Из-за гауссовой природы шума мы не можем построить строгую границу шума (некоторые точки могут находиться за пределами окружности).

2.2. Одномерные созвездия как частный случай двумерных

Для упрощения рассуждений важно рассмотреть не только двумерные, но и одномерные созвездия. Интересно, что на практике они являются репрезентацией созвездий для амплитудной модуляции.

Одномерное созвездие определяется как набор точек на вещественной прямой. Как и в случае двумерных созвездий, любому количеству битов m соответствует 2^m точек созвездия (у нас имеется отображение $\mu : \{0,1\}^m \to \mathbb{R}$). По аналогии с КАМ созвездиями, можно ввести базовую модуляцию как одномерные равномерно распределённые на отрезке точки.

В дальнейших рассуждениях точки одномерных созвездий будут обозначаться с верхним подчёркиванием, например $\overline{y}$. Соответствующие множества данных точек тоже будут обозначаться с верхним подчёркиванием, например $\overline{\mathcal{X}}_B^j$.

2.3. Кодирование кодами Грея точек КАМ созвездия

Перед постановкой задачи важно определить, как задаются множества $\mathcal{X}_B^j$ для КАМ созвездий. В это случае используется специальная техника кодирования точек созвездия кодами Грея. Её основное свойство: соседние (по вертикали или горизонтали) точки созвездия отличаются ровно в одном бите.

Суть алгоритма состоит в симметричном отображении.

Допустим наши точки созвездия имеют координаты $\{y_{re} + iy_{im} | y_{re}, y_{im} \in \{1, ..., M\}\}$, где $M = 2^{\frac{m}{2}}$. Тогда процедура построения кодов Грея может быть описана следующим индуктивным алгоритмом:

1. База индукции: для $m = 2$. Тогда определим

$$\mathcal{X}_0^1 = \{(1,1);(1,2)\}; \quad \mathcal{X}_1^1 = \{(2,1);(2,2)\};$$
$$\mathcal{X}_0^2 = \{(1,1);(2,1)\}; \quad \mathcal{X}_1^2 = \{(1,2);(2,2)\};$$

2. Шаг индукции: пусть для $m = n - 2$ мы получили множества $\mathcal{X'}_B^j$. Тогда определим $\mathcal{X}_B^j$ для $m = n$ следующим образом:

$$\mathcal{X}_B^j = \left\{ (y_{re}, y_{im}) | \begin{array}{l} y_{re} \in \{y'_{re}, m - y'_{re} + 1\} \\ y_{im} \in \{y'_{im}, m - y'_{im} + 1\} \end{array}, (y'_{re}, y'_{im}) \in \mathcal{X'}_B^j \right\}$$
для $j \leq m - 2$

$$\mathcal{X}_0^{m-1} = \{(y_{re}, y_{im}) | y_{re} \in \{1, ..., M\}, y_{im} \in \{1, ..., \tfrac{M}{2}\}\}$$

$$\mathcal{X}_1^{m-1} = \{(y_{re}, y_{im}) | y_{re} \in \{1, ..., M\}, y_{im} \in \{\tfrac{M}{2} + 1, ..., M\}\}$$

$$\mathcal{X}_0^m = \{(y_{re}, y_{im}) | y_{re} \in \{1, ..., \tfrac{M}{2}\}, y_{im} \in \{1, ...M\}\}$$

$$\mathcal{X}_1^m = \{(y_{re}, y_{im}) | y_{re} \in \{\frac{M}{2} + 1, ..., M\}, y_{im} \in \{1, ..., M\}\}$$

В реальности координаты созвездия могут быть любыми, но для произвольных координат $\{x + iy | x \in \{x_1, ..., x_M\}, y \in \{y_1, ..., y_M\}\}$ при условии $x_i <= x_j, y_i <= y_j$ при $i < j$ коды Грея строятся аналогично.

Для одномерных созвездий также существует процедура кодирования кодами Грея. Она очень просто определяется через коды Грея двумерного созвездия. Если мы возьмём из двумерного созвездия подмножество точек с одинаковой координатой y_{im} (назовём их $\mathcal{Y}$) и биты j, в которых $\mathcal{Y} \not\subset \mathcal{X}_0^j$ и $\mathcal{Y} \not\subset \mathcal{X}_1^j$. Тогда множества $\mathcal{Y} \cap \mathcal{X}_0^j$ и $\mathcal{Y} \cap \mathcal{X}_1^j$ будут определять кодирование кодами Грея соответствующих одномерных точек созвездия.

2.4. Постановка задачи и её решение для нисходящей передачи

Из рассуждений, приведенных в предыдущих главах, мы можем сделать вывод, что наша главная цель - выбрать нормализованное созвездие задаваемое отображением
$\mu : \{0, 1\}^m \to \mathbb{C}$, которое максимизирует взаимную информацию:

$$\sum_{j=1}^{m} \mathbb{E}_{Y(\mu),B} \left(\log_2 \frac{\sum_{x \in \mathcal{X}_B^j} P_{Y|X}(Y|x))}{\sum_{x \in \mathcal{X}} P_{Y|X}(Y|x))} \right) \to \max_{\mu} \tag{4}$$

$$s.t. \quad \frac{1}{2^m} \sum_{s \in \{0,1\}^m} \|\mu(s)\|^2 = 1 \tag{5}$$

Важно отметить, что математическое ожидание берётся от зависимых случайных величин $Y(\mu)$ и B. Случайная величина $Y(\mu) = \frac{1}{2^m} \sum_{i=1}^{2^M} Y_i(\mu)$, где $Y_i(\mu) = \mathcal{N}(y_i(\mu), \sigma^2)$ является нормальной случайной величиной, соответствующей передаче i-той точке созвездия. Случайная величина B соответствует j-тому биту i-той точки созвездия.

Можно заметить, что целевая функция взаимной информации не является выпуклой. Например, при варьировании только одной точки и фиксации всех остальных, мы можем понять, как функция зависит от своих двух параметров. Из рис. 3 видно, что у функции двух переменных есть как области выпуклости вверх, так и области выпуклости вниз. Невыпуклость функции приводит к проблемам с локальным максимумом. Это увеличивает сложность оптимизации.

Несмотря на это, можно увидеть, что функция имеет несколько хороших свойств, в том числе она выпукла вверх в некоторой окрестности КАМ созвездия.

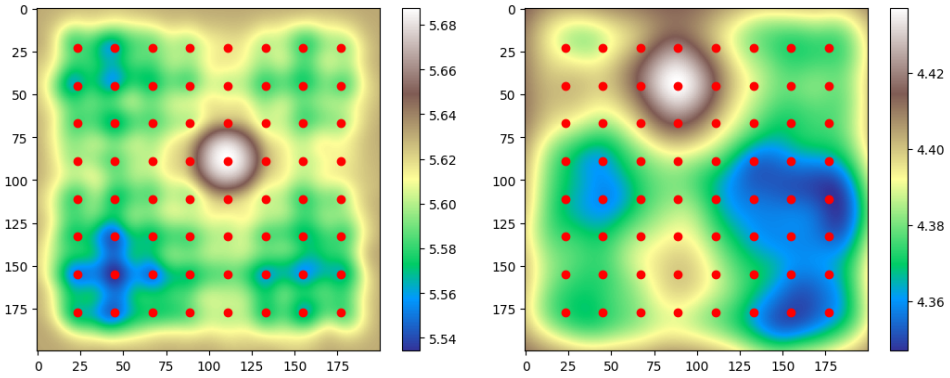

Рис. 3. Зависимость функции взаимной информации от двух параметров — действительной и мнимой координаты одной из точек созвездия. Левый график показывает зависимость для малого шума, а правый — для большого

Утверждение 2.1. При нормальном шуме со среднеквадратическим отклонением σ меньше, чем половина расстояния между соседними точками КАМ созвездия $d_x = min_{i \neq j}(\frac{|\overline{x}_i - \overline{x}_j|}{2})$, функция взаимной информации одномерного созвездия

$$I(\overline{y}_1, ..., \overline{y}_{2^m}) = \frac{1}{m} \sum_{j=1}^{m} \mathbb{E}_{\overline{y},B} \log_2 \frac{\sum_{\overline{x} \in \overline{\mathcal{X}}_B^j} \exp\left(-\frac{1}{2\sigma^2} |\overline{y} - \overline{x}|^2\right))}{\frac{1}{2} \sum_{\overline{x} \in \overline{\mathcal{X}}} \exp\left(-\frac{1}{2\sigma^2} |\overline{y} - \overline{x}|^2\right)}$$

строго выпукла вверх в области

$$\overline{y}_k \in [\overline{x}_k - d_x, \overline{x}_k + d_x]$$

Доказательство.

Для простоты обозначим

$$f_k(\overline{y}) = \sum_{j=1}^{m} \log_2 \frac{\sum_{\overline{x} \in \overline{\mathcal{X}}_{B(\overline{x}_k)}^j} \exp\left(-\frac{1}{2\sigma^2} |\overline{y} - \overline{x}|^2\right))}{\sum_{\overline{x} \in \overline{\mathcal{X}}} \exp\left(-\frac{1}{2\sigma^2} |\overline{y} - \overline{x}|^2\right)}$$

Заметим, что плотность входного распределения равна

$$p(\overline{y}) = \frac{1}{2^m} \sum_{t=1}^{2^m} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2} \|\overline{y} - \overline{y}_t\|^2}$$

Из этого

$$\frac{\partial I}{\partial \overline{y}_k} = \frac{1}{2^{m+1}\pi\sigma^2} \int\limits_{\mathbb{R}} \frac{\overline{y} - \overline{y}_k}{\sigma^2} e^{-\frac{1}{2\sigma^2}(\overline{y}-\overline{y}_k)^2} f_k(\overline{y}) d\overline{y}$$

Далее

$$\frac{\partial^2 I}{\partial \overline{y}_k^2} = \frac{1}{2^{m+1}\pi\sigma^4} \int\limits_{\mathbb{R}} \left(\left(\frac{\overline{y} - \overline{y}_k}{\sigma} \right)^2 - 1 \right) e^{-\frac{1}{2\sigma^2}(\overline{y}-\overline{y}_k)^2} f_k(\overline{y}) d\overline{y}$$

$$\frac{\partial^2 I}{\partial \overline{y}_{k_1} d\overline{y}_{k_2}} = 0$$

Следуя условиям выпуклости функции и факту, что $\int\limits_{\mathbb{R}} (x^2 - 1) e^{-\frac{x}{2}} dx = 0$, для доказательства утверждения достаточно доказать, что

$$\sum_{j=1}^{m} \int\limits_{\mathbb{R}} \left(\left(\frac{\overline{y} - \overline{y}_k}{\sigma} \right)^2 - 1 \right) e^{-\frac{1}{2\sigma^2}(\overline{y}-\overline{y}_k)^2} f_k(\overline{y}) d\overline{y} < 0$$

Рассмотрим свойства функции $f_k(\overline{y})$. Она будет выпукла в области $[\overline{x}_k - 2d_x, \overline{x}_k + 2d_x]$ и принимать свой максимум в области $[\overline{x}_k - d_x, \overline{x}_k + d_x]$.

Условие $\sigma < d_x$ позволяет сделать разницу $\exp\left(-\frac{(2d_x)^2}{2\sigma^2}\right)$ между экспонентами от соседних точек пренебрежительно малой. Таким образом, из-за природы функции $f_k(\overline{y})$, в окрестностях точек КАМ созвездия, отличных от $\overline{x}_k$, будет преобладать экспонента знаменателя одного из слагаемых. Поэтому для внутренних точек (с индексами $1 < k < 2^m$) значение этой функции при $\overline{y} > \overline{x}_k + 2d_x$ будет меньше, чем $f_k(\overline{x}_k + 2d_x)$. Аналогично, значение при $\overline{y} < \overline{x}_k - 2d_x$ будет меньше, чем $f_k(\overline{x}_k - 2d_x)$.

Для внешних точек ($k \in \{1, 2^m\}$) данное свойство не будет выполнено, но, используя то же свойство внутренних точек для одного из лучей ($\overline{y} < \overline{x}_k - 2d_x$ или $\overline{y} > \overline{x}_k + 2d_x$) и пренебрежительно малое значение $f_k(\overline{y})$ на другом луче, можно показать, что $f_k(\overline{y_k} - \Delta) + f_k(\overline{y_k} + \Delta) < f_k(\overline{y_k} - d_x) + f_k(\overline{y_k} + d_x)$ при $\Delta > d_x$. То же свойство будет выполнено для внутренних точек.

Функция $\left(\left(\frac{\overline{y} - \overline{y}_k}{\sigma} \right)^2 - 1 \right) e^{-\frac{1}{2\sigma^2}(\overline{y}-\overline{y}_k)^2}$ симметрична относительно $\overline{y}_k$ и принимает отрицательные значения при $\overline{y}_k - \sigma < \overline{y} < \overline{y}_k + \sigma$ и положительные на лучах $\overline{y} < \overline{y}_k - \sigma$ и $\overline{y} > \overline{y}_k + \sigma$. Применяя свойство выпуклости $f_k(\overline{y_k})$ в области $[\overline{x}_k - 2d_x, \overline{x}_k + 2d_x]$, мы получаем что $f_k(\overline{y_k} - \Delta) + f_k(\overline{y_k} + \Delta) < f_k(\overline{y_k} - \sigma) + f_k(\overline{y_k} + \sigma)$ для $|\Delta| < \sigma$ и $f_k(\overline{y_k} - \Delta) + f_k(\overline{y_k} + \Delta) > f_k(\overline{y_k} - \sigma) + f_k(\overline{y_k} + \sigma)$ для $|\Delta| > \sigma$. Таким образом

$$\int\limits_{\mathbb{R}} \left(\left(\frac{\overline{y} - \overline{y}_k}{\sigma} \right)^2 - 1 \right) e^{-\frac{1}{2\sigma^2}(\overline{y}-\overline{y}_k)^2} f_k(\overline{y}) =$$

$$= \int\limits_{\overline{y_k}}^{\infty} \left(\left(\frac{\overline{y} - \overline{y}_k}{\sigma} \right)^2 - 1 \right) e^{-\frac{1}{2\sigma^2}(\overline{y}-\overline{y}_k)^2} (f_k(-\overline{y}) + f_k(\overline{y}))d\overline{y} <$$

$$< \int\limits_{\overline{y_k}}^{\infty} \left(\left(\frac{\overline{y} - \overline{y}_k}{\sigma} \right)^2 - 1 \right) e^{-\frac{1}{2\sigma^2}(\overline{y}-\overline{y}_k)^2} (f_k(\overline{y_k} - d_x) + f_k(\overline{y_k} + d_x)))d\overline{y} = 0 \tag{6}$$

что и требовалось для завершения доказательства.

□

Утверждение 2.2. Функция взаимной информации одномерного созвездия

$$\frac{1}{m} \sum_{j=1}^{m} \mathbb{E}_{\overline{y},B} \log_2 \frac{\sum_{x \in \overline{\mathcal{X}}_B^j} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right))}{\frac{1}{2}\sum_{x \in \overline{\mathcal{X}}} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right)}$$

симметрична.

Доказательство. Пусть у нас даны точки исходного одномерного созвездия $\mathcal{Y}'$. Докажем, что созвездие с точками

$$\mathcal{Y} = \{\overline{y}_k | \overline{y}_k = -\overline{y'}_{m-k+1}, \overline{y'}_{m-k+1} \in \mathcal{Y}'\}$$

имеет то же значение взаиной информации.

Заметим, что для $\overline{y}_k$ и $\overline{y'}_{m-k+1}$ все соответствующие множества $\overline{\mathcal{X}}_B^j$ симметричны. Из этого следует, что соответствующие математические ожидания будут совпадать:

$$\mathbb{E}_{\overline{y}_k,B} \log_2 \frac{\sum_{x \in \overline{\mathcal{X}}_B^j} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right))}{\frac{1}{2}\sum_{x \in \overline{\mathcal{X}}} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right)} =$$

$$= \mathbb{E}_{\overline{y'}_{m-k+1},B} \log_2 \frac{\sum_{x \in \overline{\mathcal{X}}_B^j} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right))}{\frac{1}{2}\sum_{x \in \overline{\mathcal{X}}} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right)}$$

Из этого следует равенство итоговых математических ожиданий (так как из равновероятности точек созвездия следует $\mathbb{E}_{\overline{y},B} = \frac{1}{2^m}\sum_{k=1}^{2^m} \mathbb{E}_{\overline{y}_k,B}$).

□

Данные утверждения имеет практическое применение. Как видно из постановки задачи, количество оптимизационных параметров y (4) равно удвоенному количеству точек созвездия. То есть для кодировки m битов нужно оптимизировать 2^{m+1} действительных переменных (или 2^m комплексных). Это число довольно большое. Из-за специфики задачи это число можно существенно снизить.

Теорема 2.1. При нормальном шуме со среднеквадратическим отклонением σ меньше, чем половина расстояния между соседними точками КАМ созвездия d_x, внутри области

$$y_k \in \left\{ y_{re} + iy_{im} | \begin{array}{l} y_{re} \in [(y_k)_{re} - d_x, (y_k)_{re} + d_x] \\ y_{im} \in [(y_k)_{im} - d_x, (y_k)_{im} + d_x] \end{array} \right\}$$

существует оптимальное созвездие, являющееся декартовым произведением двух одинаковых и симметричных одномерных созвездий.

Доказательство.

Введём обозначения $y = y_{re} + i \cdot y_{im}$ и $x = x_{re} + i \cdot x_{im}$.
Заметим, что $\|y-x\|^2 = \|y_{re}-x_{re}\|^2+\|y_{im}-x_{im}\|^2$. Это ведёт к следующему разложению:

$$\sum_{x \in \mathcal{X}_B^j} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right) =$$

$$\sum_{x \in \mathcal{X}_B^j} \exp\left(-\frac{1}{2\sigma^2}|y_{re}-x_{re}|^2\right) \exp\left(-\frac{1}{2\sigma^2}|y_{im}-x_{im}|^2\right) =$$

$$\left(\sum_{x_{re} \in (\mathcal{X}_B^j)_{re}} \exp\left(-\frac{|y_{re}-x_{re}|^2}{2\sigma^2}\right)\right)\left(\sum_{x_{im} \in (\mathcal{X}_B^j)_{im}} \exp\left(-\frac{|y_{im}-x_{im}|^2}{2\sigma^2}\right)\right) \tag{7}$$

где $(X_B^j)_{re}$ и $(X_B^j)_{im}$ являются множествами реальных и мнимых частей множества X_B^j соответственно. Это разложение верно при условии, что само множество X_B^j является декартовым произведением одномерных множеств.

Аналогично получаем

$$\sum_{x\in\mathcal{X}} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right) =$$
$$= \left(\sum_{x_{re}\in\mathcal{X}_{re}} \exp\left(-\frac{|y_{re}-x_{re}|^2}{2\sigma^2}\right)\right)\left(\sum_{x_{im}\in\mathcal{X}_{im}} \exp\left(-\frac{|y_{im}-x_{im}|^2}{2\sigma^2}\right)\right) \quad (8)$$

Используя свойство логарифма в итоге получаем, что наше математическое ожидание раскладывается на сумму двух отвечающих за действительную и мнимую части соответственно:

$$\sum_{j=1}^{m} \mathbb{E}_y \log_2 \frac{\sum_{x\in\mathcal{X}_B^j} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right))}{\frac{1}{2}\sum_{x\in\mathcal{X}} \exp\left(-\frac{1}{2\sigma^2}|y-x|^2\right)} =$$
$$= \sum_{j=1}^{m} \mathbb{E}_{y_{re}} \log_2 \frac{\sum_{x_{re}\in(\mathcal{X}_B^j)_{re}} \exp\left(-\frac{1}{2\sigma^2}|y_{re}-x_{re}|^2\right))}{\frac{1}{2}\sum_{x_{re}\in\mathcal{X}_{re}} \exp\left(-\frac{1}{2\sigma^2}|y_{re}-x_{re}|^2\right)} +$$
$$+ \sum_{j=1}^{m} \mathbb{E}_{y_{im}} \log_2 \frac{\sum_{x_{im}\in(\mathcal{X}_B^j)_{im}} \exp\left(-\frac{1}{2\sigma^2}|y_{im}-x_{im}|^2\right))}{\frac{1}{2}\sum_{x_{im}\in\mathcal{X}_{im}} \exp\left(-\frac{1}{2\sigma^2}|y_{im}-x_{im}|^2\right)} \quad (9)$$

Теперь вспомним про наше ограничение (4):

$$\sum_{k=1}^{2^m} \|y_k\|^2 = \sum_{k=1}^{2^m} \|(y_k)_{re}\|^2 + \sum_{k=1}^{2^m} \|(y_k)_{im}\|^2 = N_{re} + N_{im} = 2^m$$

Из выпуклости функции (9) (по утверждению 2.1) следует, что в оптимальном решении $N_{re} = N_{im}$. Если это не так, то сделаем подстановку $(y_k)_{re} \to \frac{(y_k)_{re}+(y_k)_{im}}{2}, (y_k)_{im} \to \frac{(y_k)_{re}+(y_k)_{im}}{2}$) и это увеличит значение взаимной информации.

Так как $N_{re} = N_{im}$, то задача (4) распадается на две одинаковые подзадачи, имеющие одинаковое решение. И их итоговое решение также будет одинаковым.

Симметричность созвездия следует из симметричности функции взаимной информации (утверждение 2.2). Здесь также, если мы возьмём решение y^1 и симметричное к нему y^2 (отличное от y^1) с той же взаимной информацией, то созвездие $y = \frac{y^1+y^2}{2}$ будет иметь большую взаимную информацию.

□

Данная теорема позволяет существенно облегчить оптимизацию функции внутри окрестности КАМ созвездия. Она позволяет существенно сократить количество переменных. В обычной оптимизационной задаче (4)

у нас $2^{m+1}-1$ независимых и одна зависимая переменная. Представление созвездия в виде декартова произведения позволяет преобразовать эту задачу, сократив количество независимых переменных до $2^{\frac{m}{2}-1}-1$.

Например для оптимизации созвездия для 4 бит нам потребуется всего один оптимизационный параметр, для 6 бит понадобится 3 параметра, а для 8 бит нам понадобится 7 параметров.

Заметим, что теорема доказывает существование оптимального созвездия только для шумов с среднеквадратичным отклонением $\sigma < d_x$, которые соответствуют случаям, которые встречаются на практике При больших же шумах данная теорема не работает. В этом случае точки оптимального созвездия начинают склеиваться и оптимальная точка в этом случае может быть очень далеко от соответствующей точки КАМ созвездия. Но несмотря на это, свойство декартовости оптимального созвездия сохраняется.

3. Результаты симуляций

Тесты проводились на симуляционной платформе Sionna [2]. Первая часть моделирования была проведена для упрощенного канала. Здесь были получены оптимальные созвездия путем максимизации взаимной информации (смотрите некоторые примеры созвездий на рис. 4).

Algorithm 3.1 Алгоритм Adam для оптимизации созвездия картографов

Input
SNR, $QAM\_Constalletion$ ▷ ОСШ и КАМ созвездие
Output
$Mapper\_Constellation$
$Mapper\_Constellation \leftarrow QAM\_Constallation$
$Demapper\_Constellation \leftarrow QAM\_Constallation$
for $t = 1 : T$ **do** ▷ T это число эпох Adam
$bits \leftarrow BinarySource()$ ▷ Генерация случайной последовательности
$x \leftarrow Mapper(bits, Mapper\_Constellation)$
$y \leftarrow AWGN\_Channel(x, SNR)$ ▷ Прохождение сигнала через АБГШ канал
$llrs \leftarrow Demapper(y, Demapper\_Constellation)$
$loss \leftarrow BinaryCrossentropy(bits, llrs)$
$Mapper\_Constellation \leftarrow AdamUpdate(Mapper\_Constellation, loss)$
end for
return $Mapper\_Constellation$

Обучение созвездий проводилось с использованием оптимизатора Adam [9]. Мы описываем процесс обучения в алгоритме 3.1. Мы можем

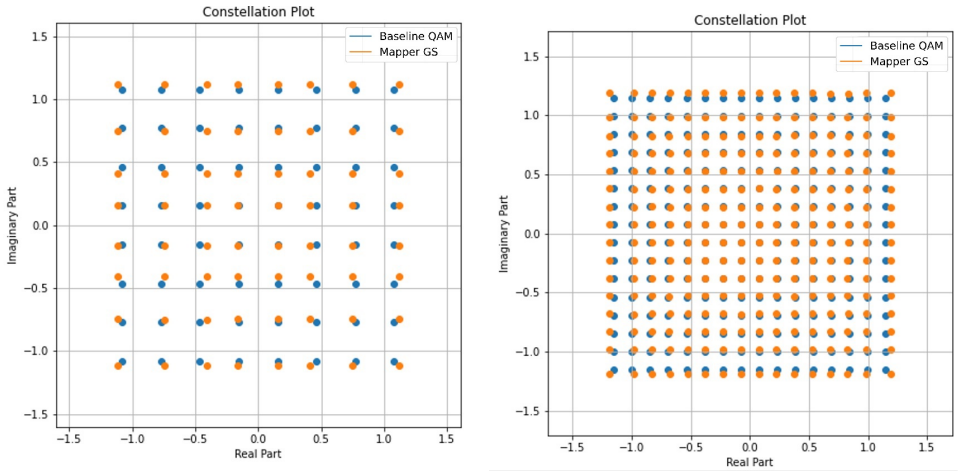

Рис. 4. Примеры созвездий. Синие точки обозначают базовое созвездие КАМ. Оранжевые точки обозначают изменённое созвездие

применить этот алгоритм к любому уровню сигнала-шум (ОСШ) и начать с любого базового созвездия КАМ. Здесь функция взаимной информации рассчитывается методом Монте-Карло [17]. Для случайной битовой последовательности с использованием отображения μ мы получаем последовательность комплексных значений x, которая с дополнительным шумом преобразуется в последовательность комплексных значений y. Используя (1), мы получаем llr и, наконец, вычисляем функцию потерь как двоичную перекрестную энтропию. Сложность этого подхода зависит от требуемой точности. Но точки созвездия вычисляются в автономном режиме и не влияют на скорость передачи.

Для каждого ОСШ и каждого начального КАМ-созвездия был рассчитан объем передаваемой информации в случаях однородных и неоднородных созвездий (см. рис. 5). Здесь мы рассмотрим соотношение пропускной способности между “Baseline QAM”, “Mapper GS” и "Shannon limit". “Baseline QAM” - это алгоритм с фиксированой системой созвездий (стандартная КАМ модуляция), “Mapper GS” - алгоритм с неоднородной системой созвездий, который максимизирует функцию взаимной информации, а "Shannon limit максимальную теоретическую скорость передачи данных. Мы видим, что созвездия “Mapper GS” имеют выигрыш во взаимной информации до 1,5% при высоких значениях ОСШ (отношения сигнал/шум).

Полученные созвездия были протестированы на более сложных сценариях. Сначала был рассмотрен сценарий, в котором был учтён эффект LDPC-кодирования. Симулятор Sionna использует LDPC-коды из стан-

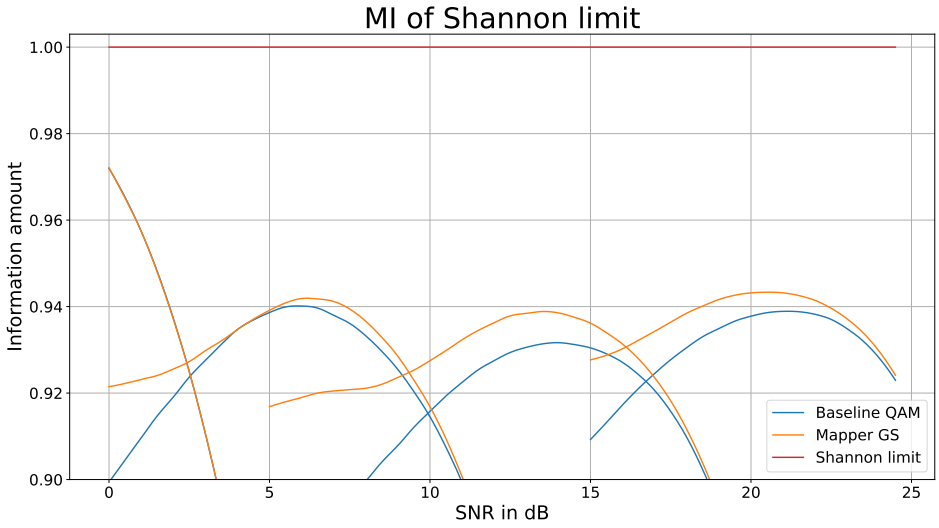

Рис. 5. Взаимная информация. Ось x представляет собой частоту ОСШ в дБ. Ось y представляет процент дополнительной информации, которую мы можем передать пользователю, по сравнению с пределом Шеннона. Здесь синяя линия показывает процент передачи для базовой КАМ-модуляции. Оранжевая линия показывает процент передачи с помощью изменённого созвездия.

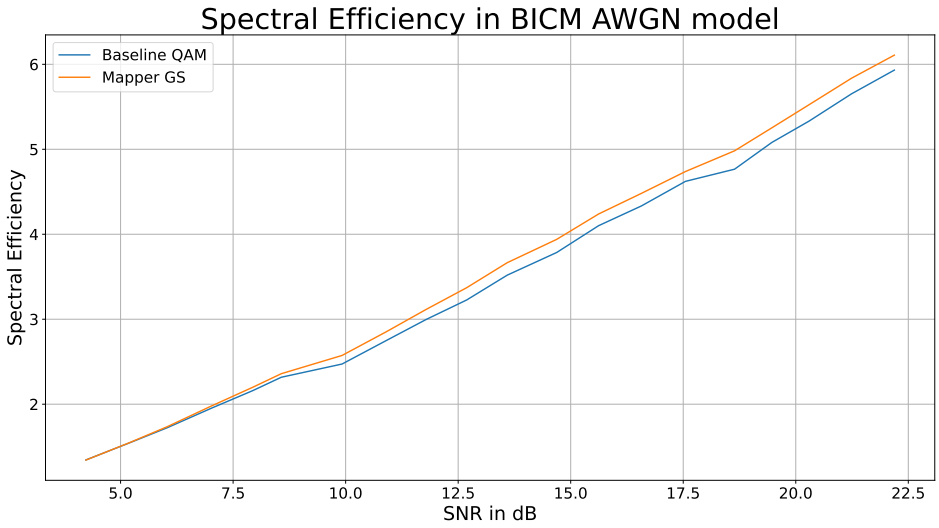

Рис. 6. Спектральная эффективность. По оси x отложено значение ОСШ в дБ. По оси y - количество информации, которое мы можем передать пользователю за единицу времени. Здесь синяя линия показывает количество информации, передаваемой с помощью базовой КАМ-модуляции. Оранжевая линия показывает объем информации, передаваемой с помощью изменённого созвездия.

дарта 3GPP TS 38.211 V17.0.0 (2021-12). Результат можно увидеть на рис. 6.

Спектральная эффективность (в битах на единицу времени) является показателем эффективности полосы пропускания. В нашем случае она определяется как

$$SE = (1 - BLER) \cdot Coderate \cdot num\_bits$$

где $BLER$ (Block Error Rate) это вероятность ошибки при передаче одного блока LDPC, $Coderate$ это скорость кодирования на основе СМК, num_bits это количество битов, кодирующих созвездие. Например, для КАМ16: $num\_bits = log_2(16) = 4$. Значение num_bits также зависит от СМК.

Значения $Coderate$ и num_bits получены из 3GPP стандарта. Значение $BLER$ было получено путём симуляции LDPC кодирования.

Для сравнения со старым подходом, рассматривается график относительного выигрыша (Рис. 7).

$$SE_{gain} = \frac{SE_{after}}{SE_{before}} - 1$$

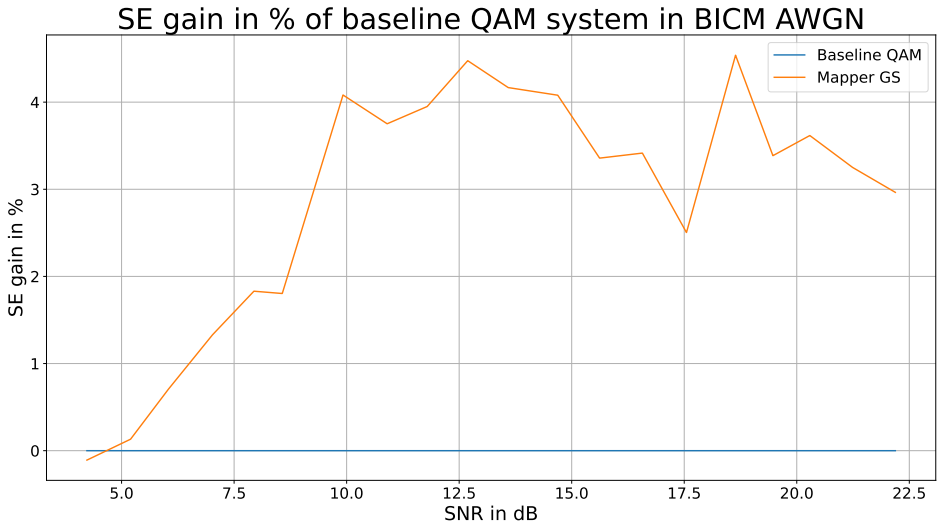

Рис. 7. График относительного выигрыша в спектральной эффективности. По оси x отложено значение ОСШ в дБ. По оси y - процент дополнительной информации, которую мы можем передать пользователю, по сравнению с базовой подходом с модуляцией КАМ. Оранжевая линия показывает процент передачи с использованием формованного созвездия.

В этой серии экспериментов мы получаем выигрыш в размере до 4%.

Вторая серия экспериментов была проведена с учетом модуляции с ортогональным частотным мультиплексированием (Orthogonal frequency-division multiplexing, OFDM). Архитектура системы состоит из LDPC-кода с прямой коррекцией ошибок, устройства перемешивания битов, средства преобразования цифрового сигнали и обратно, средства оценки канала методом наименьших квадратов, эквалайзера LMMSE, модулятора OFDM. В системе используются различные модели беспроводных каналов 3GPP с кластеризованной линией задержки (Clustered delay line, CDL) вне зоны прямой видимости (Non-Line-of-Sight, NLoS): A, B, C; и модели каналов прямой видимости (Line-of-Sight, LoS) CDL: D, E (Fig. 8) [11]. Временные модели моделируются в режиме реального времени с учетом помех между символами и разными частотами, в то время как частотные

модели моделируются непосредственно в частотной области без учета этих помех.

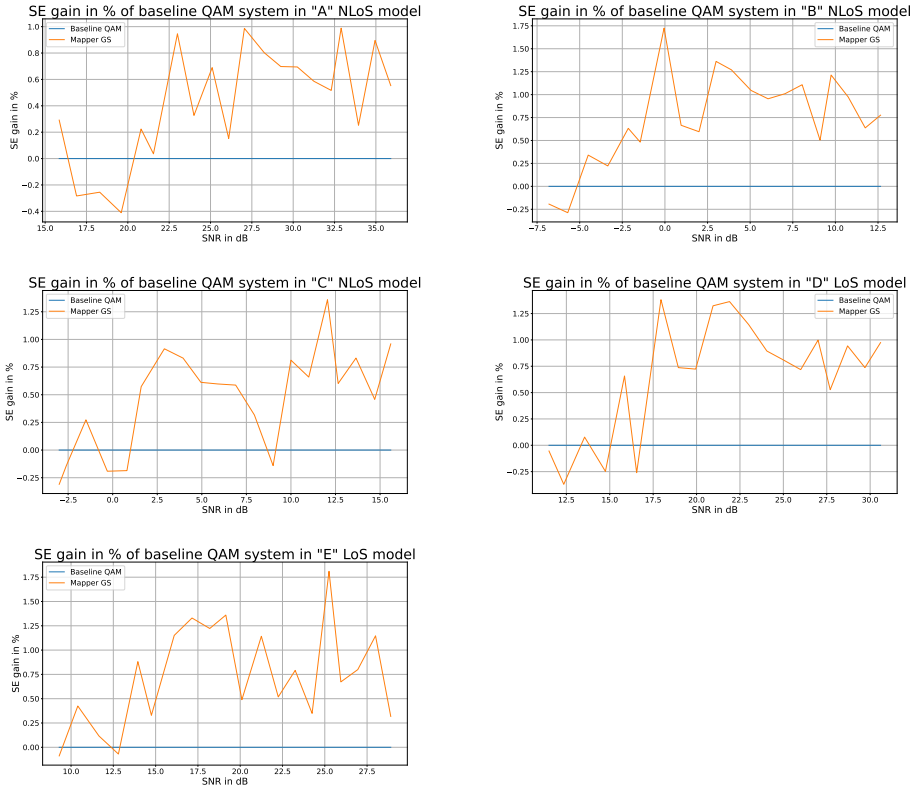

Рис. 8. Графики относительного выигрыша в спектральной эффективности для моделей беспроводных каналов CDL 3GPP:
NLoS A, NLoS B, NLoS C, LoS D, LoS E.

Для этих серий экспериментов мы получаем выигрыш в размере до 1.75%.

4. Заключение

В этой статье представлено решение проблемы формирования созвездий с условием, что оно может быть изменено только на одной стороне. Созвездия, оптимизированные на простой модели, улучшают функцию спектральной эффективности на более сложных моделях, которые учитывают различные кодирования. Было показано, что такой подход может дать выигрыш около 4% для простых моделей каналов АБГШ с кодом LDPC и около 1,5% для моделей OFDM LoS и NLoS.

Geometric shaping of the QAM constellation on the transmitter side in a wireless channel with a fixed decoding algorithm.
Yudakov D.A.

The paper explores the optimization of constellation geometry to improve throughput in wireless communication systems. It presents an approach where constellations can be unevenly distributed to enhance transmission efficiency. The optimization is performed on the base station side, and its effectiveness is validated through numerical experiments using LDPC coding, OFDM modulation, and MIMO technologies.

Keywords: Wireless, Constellation diagram, Geometric Shaping, Mutual Information.

References

[1] O'shea Timothy, Hoydis Jakob, "An introduction to deep learning for the physical layer", *IEEE Transactions on Cognitive Communications and Networking*, **3**:4 (2017), 563–575.

[2] Hoydis Jakob, Cammerer Sebastian, Aoudia Fay, Vem Avinash, Binder Nikolaus, Marcus Guillermo, Keller Alexander, "Sionna: An open-source library for next-generation physical layer research", *arXiv preprint arXiv:2203.11854*, 2022.

[3] Mirani Ali, Agrell Erik, Karlsson Magnus, "Low-complexity geometric shaping", *Journal of Lightwave Technology*, **39**:2 (2020), 363–371.

[4] Shannon Claude E, "A mathematical theory of communication", *The Bell system technical journal*, **27**:3 (1948), 379–423.

[5] Channels, NR Physical, "Modulation (Release 15), V15. 4.0, document TS 38.211, 3GPP, Dec. 2018".

[6] Svensson Arne, "An introduction to adaptive QAM modulation schemes for known and predicted channels", *Proceedings of the IEEE*, **95**:12 (2007), 2322–2336.

[7] Bobrov Evgeny, Dordzhiev Adyan, "On Probabilistic QAM Shaping for 5G MIMO Wireless Channel with Realistic LDPC Codes", *arXiv e-prints*, 2023.

[8] Qu Zhen, Djordjevic Ivan B, "On the probabilistic shaping and geometric shaping in optical communication systems", *IEEE Access*, **7** (2019), 21454–21464.

[9] Da Kingma, "A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[10] Hu Fangchen, Zou Peng, Li Guoqiang, Yu Weixiang, Chi Nan, "Enhanced performance of CAP-modulated visible light communication system utilizing geometric shaping and rotation coding", *IEEE Photonics Journal*, **11**:5 (2019), 1–12.

[11] Barb Gordana, Otesteanu Marius, "On the influence of delay spread in tdl and cdl channel models for downlink 5g mimo systems", *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, **5** (2019), 0958–0962.

[12] Chen Dechao, Zhang Jingwen, Zhao Rui, "Adaptive modulation and coding in satellite-integrated 5G communication system", *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, **6** (2021), 1402–1407.

[13] Hassan Najeeb Ul, Xu Wen, Kakkavas Anastasios, "Applying Coded Modulation with Probabilistic and Geometric Shaping for Wireless Backhaul Channel", *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, **5** (2018), 1–5.

[14] Jones Rasmus T, Yankov Metodi P, Zibar Darko, "End-to-end learning for GMI optimized geometric constellation shape", *45th European Conference on Optical Communication (ECOC 2019)*, **4** (2019), 1–4.

[15] Chen Bin, Okonkwo Chigo, Hafermann Hartmut, Alvarado Alex, "Increasing achievable information rates via geometric shaping", *2018 European Conference on Optical Communication (ECOC)*, **3** (2018), 1–3.

[16] Caire Giuseppe, Taricco Giorgio, Biglieri Ezio, "Bit-interleaved coded modulation", *Proceedings of ICC'97-International Conference on Communications*, **3** (1997), 1463–1467.

[17] Rubinstein Reuven Y, Kroese Dirk P, *Simulation and the Monte Carlo method*, John Wiley & Sons, 2016.

Часть 3
Математические модели

О растущей нижней оценке функции Шеннона длины единичного проверяющего теста при константных неисправностях на выходах элементов в формулах над базисами, близкими к стандартному

Чж. Цуй[1], Д. С. Романов[2]

В статье установлены асимптотически равные числу переменных нижние оценки функций Шеннона длины единичного проверяющего теста при константных неисправностях на выходах элементов в булевых формулах над базисами $\{x\&y,\ x\vee y,\ \bar{x}\}$, $\{x\&y,\ \bar{x}\}$, $\{x\vee y,\ \bar{x}\}$, $\{x\,\&\,\bar{y},\ \bar{x}\}$, $\{x\vee\bar{y},\ \bar{x}\}$.

Работа выполнена при финансовой поддержке Минобрнауки в рамках реализации программы Московского центра фундаментальной и прикладной математики по соглашению № 075-15-2022-284.

Ключевые слова: проверяющий тест, константные неисправности, булева формула

1. Введение

В настоящей работе демонстрируется эффект, показывающий наличие качественной разницы (для базисов $B_0 = \{x\,\&\,y,\ x\vee y,\ \bar{x}\}$, $B_{0,1} = \{x\,\&\,y,\ \bar{x}\}$, $B^*_{0,1} = \{x\vee y,\ \bar{x}\}$, $B_{0,2} = \{x\ \&\ \bar{y},\ \bar{x}\}$, $B^*_{0,2} = \{x\vee\bar{y},\ \bar{x}\}$) между поведениями функций Шеннона длины единичного проверяющего теста при константных неисправностях на выходах элементов в булевых формулах и в схемах из функциональных элементов (СФЭ). Именно, указанные функции Шеннона для СФЭ ограничены сверху константами для любого значения аргумента, тогда как в случае формул эти функции Шеннона растут асимптотически не медленнее, чем число n переменных, от которых зависят булевы функции.

Все определения, не введенные в этой статье, можно почерпнуть из книг [1, 2].

[1] *Цуй Чжэнюй* — асп. факультета ВМК МГУ имени М.В. Ломоносова, e-mail: ourobros1234@gmail.com.

Cui Zhenyu — postgraduate student of CMC Faculty, Lomonosov MSU

[2] *Романов Дмитрий Сергеевич* — проф. факультета ВМК МГУ имени М.В. Ломоносова, e-mail: romanov@cs.msu.ru.

Romanov Dmitrii Sergeevich — professor of CMC Faculty, Lomonosov MSU

Булева формула может быть представлена как СФЭ (над тем же базисом, что и формула) с одним выходом без ветвлений на выходах функциональных элементов (ФЭ) [1, §§ 2–3 гл. 2]. Поэтому при описании формул (как СФЭ формульного типа) будет повсеместно использоваться схемная терминология. Будем в любой базис неявно включать тождественную функцию.

Пусть на реализовывавшую булеву функцию $f(x_1, \ldots, x_n)$ СФЭ S над полным базисом B подействовал источник неисправностей U, превратив схему S в любую схему из конечного содержащего схему S множества H схем. Обычно источник неисправностей сохраняет множество входов и множество выходов исходной схемы и определяется теми поломками, которые он может произвести в схеме. При тестовом исследовании схемы анализируются значения на выходе схемы, являющиеся откликом схемы на подачу на входы схемы входных наборов.

Константная неисправность на выходе ФЭ есть результат замены исходного ФЭ на функциональный элемент, реализующий константу. Через O_1^{c} (IO_1^{c}) обозначается источник одиночных произвольных константных неисправностей на выходах ФЭ (соответственно, источник одиночных произвольных константных неисправностей на входах и выходах ФЭ).

Множество T входных наборов схемы S является *проверяющим тестом* для схемы S относительно источника неисправностей U тогда и только тогда, когда для любой схемы S' из множества H имеет место импликация: если S' реализует булеву функцию $g(x_1, x_2, \ldots, x_n)$, не равную f, то в T найдется набор $\tilde{\alpha}$ такой, что $f(\tilde{\alpha}) \neq g(\tilde{\alpha})$.

Число наборов в тесте T обозначается через $L(T)$ и называется *длиной* теста T. *Минимальный тест* — это тест минимальной длины. Длина минимального проверяющего теста для схемы S относительно источника неисправностей U будет обозначаться через $L^{\mathrm{dt}}(U, S)$.

Схема S считается *неизбыточной* относительно источника неисправностей U, если при любой меняющей хоть на каком-то входном наборе значение на выходе хоть какого-то функционального элемента порожденной источником U одиночной поломке функционального элемента (т. н. нетривиальной поломке) полученная схема реализует функцию, не равную исходной функции f (реализуемой S в при отсутствии поломок). Под *длиной минимального проверяющего теста* для реализуемой СФЭ (формулами) над базисом B булевой функции f относительно источника неисправностей U понимается величина $L^{\mathrm{dt}}_{\mathrm{C},B}(U, f)$ (соответственно $L^{\mathrm{dt}}_{\mathrm{F},B}(U, f)$), представляющая собой минимум по всем неизбыточным реализующим f схемам (соответственно формулам) S над базисом B величин $L^{\mathrm{dt}}(U, S)$. Если для функции f отсутствуют реализующие ее неизбыточные относительно источника неисправностей U СФЭ (формулы) над базисом B, то будем считать, что $L^{\mathrm{dt}}_{\mathrm{C},B}(U, f) = 0$ (соответственно

$L^{\mathrm{dt}}_{\mathrm{F},B}(U,f)=0$). *Функцией Шеннона длины проверяющего теста относительно источника неисправностей* U для СФЭ (формул) над базисом B называется величина $L^{\mathrm{dt}}_{\mathrm{C},B}(U,n)=\max\limits_{f\in P_2(n)} L^{\mathrm{dt}}_{\mathrm{C},B}(U,f)$ (соответственно $L^{\mathrm{dt}}_{\mathrm{F},B}(U,n)=\max\limits_{f\in P_2(n)} L^{\mathrm{dt}}_{\mathrm{F},B}(U,f)$).

Перечислим результаты, связанные с оцениванием функций Шеннона длины единичного проверяющего теста относительно произвольных константных неисправностей на выходах функциональных элементов в СФЭ. Если не оговорено иное, оценки функций Шеннона приведены для произвольного целого положительного n.

В [4] фактически установлено, что в базисе $B_1=\{x\,\&\,y,\,x\oplus y,\,1\}$ любая булева функция n переменных моделируется формулой (с одной добавочной переменной), обладаающей универсальным единичным проверяющим тестом длины не более $n+4$ относительно IO^{c}_1. В [5] эта верхняя оценка понижается до $n+3$, а в [2, стр. 113–116] фактически доказано неравенство $L^{\mathrm{dt}}_{\mathrm{F},B_1}(IO^{\mathrm{c}}_1,n)\leqslant n+3$. В работах [6]–[9] продемонстрировано, что $L^{\mathrm{dt}}_{\mathrm{C},B}(O^{\mathrm{c}}_1,n)\leqslant n+3$ в произвольном полном базисе B (в ряде базисов, по существу, строятся формулы, а не СФЭ). В [10] для базиса $B'_1=\{x\,\&\,y,\,x\oplus y,\,x\sim y\}$ получено неравенство $L^{\mathrm{dt}}_{\mathrm{C},B'_1}(IO^{\mathrm{c}}_1,n)\leqslant 16$. В [11] установлено, что $2\leqslant L^{\mathrm{dt}}_{\mathrm{C},B}(O^{\mathrm{c}}_1,n)\leqslant 4$ для произвольного конечного полного базиса B. В [12] доказано, что при $n\geq 3$ в любом полном базисе B, содержащемся в множестве элементарных конъюнкций с одинаковыми степенями переменных, линейных функций двух переменных и функций, представляющих собой конъюнкцию $x_1\bar{x}_2$ и некоторой функции, $L^{\mathrm{dt}}_{\mathrm{C},B}(O^{\mathrm{c}}_1,n)\geqslant 3$. В [13] установлено точное значение $L^{\mathrm{dt}}_{\mathrm{C},\{xy,\,\bar{x},\,x\oplus y\oplus z\}}(O^{\mathrm{c}}_1,n)=2$. В [14] доказано, что $L^{\mathrm{dt}}_{\mathrm{C},\{xy,\,x\oplus y,\,1,\,0\}}(O^{\mathrm{c}}_1,n)\leqslant 3$. Отметим, что в работах [3]–[8], [12]–[14] применяется иное определение неизбыточности схемы, не делающее исключений для тривиальных поломок неконстантных ФЭ.

2. Формулировка и доказательство основного результата

В данной статье устанавливается следующее утверждение о значениях функции Шеннона длины единичного проверяющего теста относительно одиночных константных неисправностей на выходах элементов для формул над базисами $B_0=\{x\&y,\,x\vee y,\,\bar{x}\}$, $B_{0,1}=\{x\&y,\,\bar{x}\}$, $B^*_{0,1}=\{x\vee y,\,\bar{x}\}$, $B_{0,2}=\{x\,\&\,\bar{y},\,\bar{x}\}$, $B^*_{0,2}=\{x\vee\bar{y},\,\bar{x}\}$.

Теорема 1. *Для любого базиса* B *из множества* $\{B_0,\ B_{0,1},\ B^*_{0,1},\ B_{0,2},\ B^*_{0,2}\}$ *при* $n\to\infty$ *для почти всех булевых функций*

$f(x_1, \ldots, x_n)$ имеет место следующая асимптотическая нижняя оценка: $L^{\mathrm{dt}}_{\mathrm{F},B}(O^{\mathrm{c}}_1, f(x_1, \ldots, x_n)) \geq n \cdot (1 + o(1))$.

Из этого утверждения мгновенно вытекает теорема 2.

Теорема 2. *Для любого базиса B из множества $\{B_0, B_{0,1}, B^*_{0,1}, B_{0,2}, B^*_{0,2}\}$ при $n \to \infty$ имеет место следующая асимптотическая нижняя оценка:* $L^{\mathrm{dt}}_{\mathrm{F},B}(O^{\mathrm{c}}_1, n) \geq n \cdot (1 + o(1))$.

Для доказательства теоремы 1 докажем несколько лемм, вводя некоторые понятия.

Лемма 1. *Для любой тождественно не равной константе булевой функции $f(x_1, \ldots, x_n)$ существует реализующая ее неизбыточная формула над произвольным базисом B из множества $\{B_0, B_{0,1}, B^*_{0,1}, B_{0,2}, B^*_{0,2}\}$.*

Доказательство. Пусть $D_f = K_1 \vee K_2 \vee \cdots \vee K_s$ — формула над базисом B_0, представляющая собой какую-то тупиковую ДНФ функции f так, что слагаемые реализованы цепочками конъюнкторов (на незадействованные в цепочках входы конъюнкторов подаются переменные или отрицания переменных) и собраны в указанном порядке в логическую сумму цепочкой дизъюнкторов. В силу тупиковости D_f найдется такой набор α, что $K_1(\alpha) = 1$, тогда как $K_2(\alpha) = \cdots = K_s(\alpha) = 0$. На наборе α обнаруживаются все неисправности типа 0 на выходах всех дизъюнкторов. Так как $f \not\equiv const$, найдется набор β такой, что $f(\beta) = 0$. На наборе β обнаруживаются все неисправности типа 1 на выходах всех дизъюнкторов. В силу тупиковости ДНФ D_f для каждого $i = \overline{1, s}$ найдется такой набор α_i, что $K_i(\alpha_i) = 1$, тогда как остальные слагаемые на этом наборе обращаются в 0. На этом наборе обнаруживаются все неисправности типа 0 на выходах конъюнкторов из K_i и всех инверторов из K_i. В силу тупиковости ДНФ D_f для каждого множителя $x_j^{\xi_j}$ произвольной простой импликанты K_i ($i = \overline{1, s}$) найдется такой набор $\beta_{i,j}$, что $f(\beta_{i,j}) = 0$, $x_j = \bar{\xi}_j$, но при выбрасывании множителя $x_j^{\xi_j}$ из слагаемого K_i полученная элементарная конъюнкция (или константа 1, если множитель единственный) будет обращаться в 1. На этом наборе $\beta_{i,j}$ обнаруживается неисправность типа 1 на выходе того конъюнктора из K_i, на вход которого подается буква $x_j^{\xi_j}$, и того инвертора (если $\xi_j = 0$) из K_i, который связан с входом x_j. Таким образом, для случая базиса B_0 все неисправности обнаружены.

Переход к базису $B_{0,1}$ осуществляется заменой цепочки дизъюнкторов на отрицание цепочки конъюнкторов, на входы которых подаются отрицания слагаемых, упоминавшихся в предыдущем абзаце. Ясно, что тест для тупиковой ДНФ над базисом B_0 окажется и тестом для перестроенной указанным образом формулы над базисом $B_{0,1}$. Доказательство утверждения для базиса $B^*_{0,1}$ осуществляется теперь по принципу двойственности.

Переход от базиса $B_{0,1}$ к базису $B_{0,2}$ осуществляется добавлением части инверторов и избавлением от иной части инверторов включением отрицания в базисную функцию $x \,\&\, \bar{y}$. Тест не изменит своего вида. Доказательство утверждения для базиса $B^*_{0,2}$ осуществляется теперь по принципу двойственности. Лемма доказана. □

Нагруженным ФЭ формулы Σ над базисом B из ФЭ с не более чем двумя входами называется всякий из двухвходовых ФЭ в Σ, к каждому входу которого проведена дуга от функционального элемента.

Под *нагруженной сложностью* $\check{\mathcal{C}}(\Sigma)$ (*нагруженной глубиной* $\check{\mathcal{D}}(\Sigma)$) формулы Σ над базисом B из ФЭ с не более чем двумя входами понимается количество нагруженных ФЭ в Σ (соответственно, максимум — по всем ориентированным цепям в Σ — количества нагруженных элементов цепи).

Под *нагруженной сложностью* $\check{\mathcal{C}}^{\mathrm{F}}_B(f)$ (*нагруженной глубиной* $\check{\mathcal{D}}^{\mathrm{F}}_B(f)$) реализации функции f формулами над базисом B из ФЭ с не более чем двумя входами понимается минимум — по всем реализующим f формулам Σ над базисом B — величины $\check{\mathcal{C}}(\Sigma)$ (соответственно величины $\check{\mathcal{D}}(\Sigma)$).

Лемма 2. *Пусть Σ — неизбыточная относительно O^{c}_1 формула над базисом B ($B \in \{B_0,\ B_{0,1},\ B^*_{0,1},\ B_{0,2},\ B^*_{0,2}\}$), реализующая отличную от тождественной константы булеву функцию $f(x_1, \ldots, x_n)$. Тогда $L^{\mathrm{dt}}(O^{\mathrm{c}}_1, \Sigma) \geq \check{\mathcal{D}}(\Sigma)$.*

Доказательство. Если в формуле Σ имеются ФЭ, на выходах которых реализуются константы, осуществим приведение формулы Σ, последовательно заменяя ФЭ, на входы которых подаются константы, на константы, проводники или инверторы и удаляя висячие ФЭ до тех пор, пока не получится неизбыточная формула Σ', в которой нет ФЭ, на выходах которых реализуются константы (это возможно в силу того, что $f \not\equiv const$). Очевидно, $L^{\mathrm{dt}}(O^{\mathrm{c}}_1, \Sigma) \geq L^{\mathrm{dt}}(O^{\mathrm{c}}_1, \Sigma')$.

Будем теперь предполагать, что в неизбыточной формуле Σ нет функциональных элементов, на выходах которых реализуются константы. Тогда для любого единичного проверяющего теста для Σ относительно O^{c}_1 на наборах этого теста на выходах каждого ФЭ Σ будут появляться оба булевых значения. Рассмотрим цепь P максимальной нагруженной глубины в Σ и произвольный нагруженный ФЭ E в ней, которому приписана булева функция $(x \,\&\, y^{\sigma_1})^{\sigma_2}$ из базиса B.

Если передача сигналов вдоль цепи P осуществляется через левый вход E, то для проверки неисправности типа σ_1 на выходе того ФЭ, дуга от которого подается на правый вход E, в тест должен входить такой набор, что на входах E на этом наборе возникают значения $(1, \bar{\sigma}_1)$. Но

на этом наборе в силу того, что $(x \,\&\, \bar{\sigma}_1^{\sigma_1})^{\sigma_2} = \bar{\sigma}_2$, никакая неисправность нагруженного ФЭ, лежащего на цепи P выше E, не может быть обнаружена.

Аналогично, если передача сигналов вдоль цепи P осуществляется через правый вход E, то для проверки неисправности типа 1 на выходе того ФЭ, дуга от которого подается на левый вход E, в тест должен входить такой набор, что на входах E на этом наборе возникают значения $(0, \sigma_1)$. Но на этом наборе в силу того, что $(0 \,\&\, y^{\sigma_1})^{\sigma_2} = \bar{\sigma}_2$, никакая неисправность нагруженного ФЭ, лежащего на цепи P выше E, также не может быть обнаружена.

Значит, для каждого следующего нагруженного ФЭ цепи P в тест должен входить по крайней мере один новый набор, откуда с очевидностью следует утверждение леммы. □

Примитивной цепью в формуле Σ назовем всякую максимальную по включению функциональных элементов и входов схемы цепь в Σ, не содержащую нагруженных ФЭ (в случае двух последовательных нагруженных ФЭ в некоторой цепи будем считать, что примитивная цепь между этими элементами не содержит ФЭ).

Через $||\mathcal{U}_B^{\mathrm{F}}(\check{\mathcal{C}}, n)||$ обозначим число попарно неравных булевых функций вида $f(x_1, \ldots, x_n)$, реализуемых формулами над базисом B, имеющими нагруженную сложность не выше $\check{\mathcal{C}}$. В следующей лемме приводится асимптотическая нижняя оценка величины $\check{\mathcal{C}}_B^{\mathrm{F}}(f)$ для почти всех булевых функций $f(x_1, \ldots, x_n)$ (то есть для такой доли функций из множества $P_2(n)$ всех булевых функций от переменных $(x_1, \ldots, x_n)$, которая стремится к 1 при $n \to \infty$).

Лемма 3. *Для почти всех булевых функций $f(x_1, \ldots, x_n)$ из множества $P_2(n)$ при $n \to \infty$ и $B \in \{B_0,\ B_{0,1},\ B_{0,1}^*,\ B_{0,2},\ B_{0,2}^*\}$ имеет место асимптотическая нижняя оценка*

$$\check{\mathcal{C}}_B^{\mathrm{F}}(f(x_1, \ldots, x_n)) \geq \frac{2^n}{2n \log n} \cdot (1 + o(1)).$$

Доказательство. Занумеруем нагруженные ФЭ формулы Σ в порядке обхода в глубину. Ясно, что на каждый вход нагруженного элемента подается некоторая примитивная цепь, и от выхода ближайшего к корню формулы ФЭ до корня формулы Σ ведет примитивная цепь. Если примитивная цепь начинается от входа схемы, то этот вход схемы всегда относится к примитивной цепи, даже если иных вершин в примитивной цепи нет. Если примитивная цепь начинается после нагруженного ФЭ, будем использовать новую переменную x_0 для кодирования нагруженного ФЭ, после которого начинается примитивная цепь. Используя стандартные эквивалентные преобразования формул, легко показать,

что произвольная не равная константе булева функция, реализуемая примитивной цепью, допускает реализацию бесповторной примитивной цепью (без повторов одинаковых переменных). Но, как мы, фактически, видели в доказательстве леммы 2, для реализации неконстантных булевых функций можно обойтись без констант, реализуемых примитивными цепями. Используя правила де Моргана, всякую неконстантную булеву функцию, реализуемую примитивной цепью над базисом B, можно представить в виде $((\cdots((x_0^{\tau_0} \circ_0 x_{i_1}^{\tau_1}) \circ_1 x_{i_2}^{\tau_2}) \circ_2 \cdots) \circ_{t-1} x_t^{\tau_t})$ или в виде $((\cdots(x_{i_1}^{\tau_1} \circ_1 x_{i_2}^{\tau_2}) \circ_2 \cdots) \circ_{t-1} x_t^{\tau_t})$, где $\circ_j$ — конъюнкция или дизъюнкция, $\tau_j \in \{0,1\}$, $i_1, \ldots, i_t$ — попарно различные числа из множества $\{1, \ldots, n\}$ ($j = \overline{0,t}$, $t \leq n$). Поставим формуле Σ в соответствие вектор $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_{\check{\mathcal{C}}})$ длины $\check{\mathcal{C}} + 1$, в котором γ_0 — булева функция g_0, реализуемая или примитивной цепью, ведущей от выхода ближайшего к корню формулы ФЭ до корня формулы Σ, или самой формулой Σ, если в Σ нет нагруженных ФЭ, а $\gamma_\nu = (\varphi_\nu, g'_\nu, g''_\nu)$ ($\nu = \overline{1, \check{\mathcal{C}}(\Sigma)}$), где φ_ν — тип ν-го нагруженного ФЭ (функция из B), g'_ν (g''_ν) булева функция, реализуемая примитивной цепью, подающейся на левый (соответственно правый) вход ν-го нагруженного ФЭ. Элементы γ_ν ($\nu = \overline{\check{\mathcal{C}}(\Sigma) + 1, \check{\mathcal{C}}}$) устроены аналогично предыдущим, но произвольны (то есть строятся для фиктивных ФЭ и примитивных цепей). Заметим: количество видов каждой из булевых функций g_0, g'_1, $g''_1, \ldots$, $g'_{\check{\mathcal{C}}}$, $g''_{\check{\mathcal{C}}}$ не превосходит величины $n! \cdot 2^n \cdot 3^{n+1} = 3 \cdot n! \cdot 6^n$ ($n!$ — число перестановок переменных с x_1 по x_n, 2^n — количество способов расстановки n конъюнкторов или дизъюнкторов, 3^{n+1} — количество способов вхождения переменных с x_0 по x_n: каждая переменная либо входит без отрицания, либо входит с отрицанием, либо не входит в представление булевой функции). Очевидно, что количество $||\check{\mathcal{U}}_B^{\mathrm{F}}(\check{\mathcal{C}}, n)||$ не больше числа наборов γ, а, значит, удовлетворяет неравенствам:

$$||\check{\mathcal{U}}_B^{\mathrm{F}}(\check{\mathcal{C}}, n)|| \leq (3 \cdot n! \cdot 6^n)^{2\check{\mathcal{C}}+1} \cdot 2^{\check{\mathcal{C}}} + 2 \leq (18 \cdot (n!)^2 \cdot 36^n)^{\check{\mathcal{C}}+1}. \qquad (1)$$

Занумеруем все функции f из $P_2(n)$ по неубыванию значения $\check{\mathcal{C}}_B^{\mathrm{F}}(f)$ их нагруженной сложности в классе формул над базисом B и обозначим через $\hat{f}$ функцию с номером $\lceil 2^{2^n}/n \rceil$, тогда для нагруженной сложности функции $\hat{f}$ в качестве $\check{\mathcal{C}}$ будет, очевидно, выполняться неравенство

$$||\check{\mathcal{U}}_B^{\mathrm{F}}(\check{\mathcal{C}}_B^{\mathrm{F}}(\hat{f}), n)|| \geq 2^{2^n}/n. \qquad (2)$$

Сличая (1) и (2), получим:

$$(18 \cdot (n!)^2 \cdot 36^n)^{\check{\mathcal{C}}_B^{\mathrm{F}}(\hat{f})+1} \geq 2^{2^n}/n,$$

$$(\check{\mathcal{C}}_B^{\mathrm{F}}(\hat{f}) + 1) \cdot \log(18 \cdot (n!)^2 \cdot 36^n) \geq 2^n - \log n,$$

а, значит, для почти всех функций $f(x_1, \ldots, x_n)$ из $P_2(n)$ будет иметь место неравенство:

$$\check{\mathcal{C}}_B^{\mathrm{F}}(f(x_1, \ldots, x_n)) \geq \frac{2^n}{2n \log n} \cdot (1 + o(1)).$$

□

В полной аналогии со стандартным неравенством между сложностью и глубиной формулы [1, лемма 2.1 на стр. 83] из предыдущей леммы мгновенно получаем следующее утверждение.

Лемма 4. *Для почти всех булевых функций $f(x_1, \ldots, x_n)$ из множества $P_2(n)$ при $n \to \infty$ и $B \in \{B_0,\ B_{0,1},\ B_{0,1}^*,\ B_{0,2},\ B_{0,2}^*\}$ имеет место асимптотическая нижняя оценка*

$$\check{\mathcal{D}}_B^{\mathrm{F}}(f(x_1, \ldots, x_n)) \geq n \cdot (1 + o(1)).$$

Теперь утверждение теоремы 1 получается последовательным применением лемм 1, 2 и 4.

Авторы выражают глубокую благодарность профессору Сергею Андреевичу Ложкину за полезные советы и внимание к работе.

Список литературы

[1] С. А. Ложкин, *Лекции по основам кибернетики*, Изд. отдел ф-та ВМиК МГУ, М., 2004.

[2] Н. П. Редькин, *Надежность и диагностика схем*, Изд-во Моск. ун-та, М., 1992.

[3] Н. П. Редькин, "Единичные проверяющие тесты для схем при инверсных неисправностях элементов", *Матем. вопр. киберн.*, Физматлит, М., 2003, 217–230.

[4] S. M. Reddy, "Easily testable realization for logic functions", *IEEE Trans. Comput.*, **C-21**:11 (1972), 1183–1188.

[5] K. L. Kodandapani, "A note on easily testable realizations for logic functions", *IEEE Trans. Comput.*, **C-23**:3 (1974), 332–333.

[6] С. С. Коляда, "О единичных проверяющих тестах для константных неисправностей на выходах функциональных элементов", *Вестн. Моск. ун-та. Сер. 1. Матем. Мех.*, 2011, № 6, 47–49.

[7] С. С. Коляда, “Единичные проверяющие тесты для схем из функциональных элементов в базисах из элементов, имеющих не более двух входов”, *Дискр. анализ и иссл. операций*, **20**:2 (2013), 58–74.

[8] С. С. Коляда, “Единичные проверяющие тесты для схем из функциональных элементов”, *Вестн. Моск. ун-та. Сер. 1. Матем. Мех.*, 2013, № 4, 32–34.

[9] С. С. Коляда, *Верхние оценки длины проверяющих тестов для схем из функциональных элементов*, Дис. ... канд. физ.-мат. наук: 01.01.09, М., 2013.

[10] Д. С. Романов, Е. Ю. Романова, “Метод синтеза неизбыточных схем, допускающих единичные проверяющие тесты константной длины”, *Дискр. матем.*, **29**:4 (2017), 87–105.

[11] Д. С. Романов, “Метод синтеза легкотестируемых схем, допускающих единичные проверяющие тесты константной длины”, *Дискр. матем.*, **26**:2 (2014), 100–130.

[12] К. А. Попков, “Нижние оценки длин единичных тестов для схем из функциональных элементов”, *Дискр. матем.*, **29**:2 (2017), 53–69.

[13] К. А. Попков, “Короткие единичные тесты для схем при произвольных константных неисправностях на выходах элементов”, *Дискр. матем.*, **30**:3 (2018), 99–116.

[14] К. А. Попков, “Короткие единичные тесты для схем в базисе Жегалкина при произвольных константных неисправностях элементов”, *Матем. заметки*, **117**:5 (2025), 736–749.

The Growing Lower Bound for the Shannon Function of the Detection Test Set Cardinality with Respect to Single Stuck-at Faults at the Outputs of Gates in Formulas over Bases Close to the Standard One

Cui Zh., Romanov D.S.

Lower bounds asymptotically equal to the number of variables are established for Shannon functions of the cardinality of single fault detection test set with respect to stuck-at faults at outputs of gates in Boolean formulas over bases $\{x \,\&\, y,\, x \vee y,\, \bar{x}\}$, $\{x \,\&\, y,\, \bar{x}\}$, $\{x \vee y,\, \bar{x}\}$, $\{x \,\&\, \bar{y},\, \bar{x}\}$, $\{x \vee \bar{y},\, \bar{x}\}$.

The paper was published with the financial support of the Ministry of Education and Science of the Russian Federation as part of the program of the Moscow Center for Fundamental and Applied Mathematics under the agreement No 075-15-2022-284.

Keywords: fault detection test set, stuck-at faults, Boolean formula

References

[1] S. A. Lozhkin, *Lectures on Fundamentals of Cybernetics*, Publishing Dept. of CMC Faculty of Lomonosov Moscow State University, Russia, Moscow, 2004 (In Russian).

[2] N. P. Red'kin, *Reliability and Diagnostics of Circuits*, Moscow University Publishing House, Russia, Moscow, 1992 (In Russian).

[3] N. P. Red'kin, "Single fault detection test sets for circuits with respect to inverse faults of gates", *Matematicheskie Voprosy Kibernetiki*, Fizmatlit, Russia, Moscow, 2003, 217–230 (In Russian).

[4] S. M. Reddy, "Easily testable realization for logic functions", *IEEE Trans. Comput.*, **C-21**:11 (1972), 1183–1188.

[5] K. L. Kodandapani, "A note on easily testable realizations for logic functions", *IEEE Trans. Comput.*, **C-23**:3 (1974), 332–333.

[6] S. S. Kolyada, "Single checking output tests under constant faults for functional elements", *Vestnik Moskov. Univ. Ser. 1. Mat. Mekh.*, 2011, № 6, 47–49 (In Russian).

[7] S. S. Kolyada, "Single checking tests for circuits of functional elements in fan-in 2 bases", *Diskretn. Anal. Issled. Oper.*, **20**:2 (2013), 58–74 (In Russian).

[8] S. S. Kolyada, "Single checking tests for circuits of functional elements", *Vestnik Moskov. Univ. Ser. 1. Mat. Mekh.*, 2013, № 4, 32–34 (In Russian).

[9] S. S. Kolyada, *Upper bounds fof length of checking tests for circuits of functional elements*, Thesis. ... cand. phys.-math. sciences: 01.01.09, Moscow, 2013 (In Russian).

[10] D. S. Romanov, E. Yu. Romanova, "A method of synthesis of irredundant circuits admitting single fault detection tests of constant length", *Discrete Math. Appl.*, **29**:1 (2019), 35–48.

[11] D. S. Romanov, "Method of synthesis of easily testable circuits admitting single fault detection tests of constant length", *Discrete Math. Appl.*, **24**:4 (2014), 227–251.

[12] K. A. Popkov, "Lower bounds for lengths of single tests for Boolean circuits", *Discrete Math. Appl.*, **29**:1 (2019), 23–33.

[13] K. A. Popkov, "Short single tests for circuits with arbitrary stuck-at faults at outputs of gates", *Discrete Math. Appl.*, **29**:5 (2019), 321–333.

[14] K. A. Popkov, “Short single fault tests for circuits in Zhegalkin basis with arbitrary stuck-at faults of gates”, *Matem. zametki*, **117**:5 (2025), 736–749 (In Russian).

К сведению авторов публикаций в журнале «Интеллектуальные системы. Теория и приложения»

В соответствии с требованиями ВАК РФ к изданиям, входящим в перечень ведущих рецензируемых научных журналов и изданий, в которых могут быть опубликованы основные научные результаты диссертаций на соискание ученой степени доктора и кандидата наук, статьи в журнал «Интеллектуальные системы. Теория и приложения» предоставляются авторами в следующей форме:

1. Статьи, набранные в пакете LaTeX, предоставляются к загрузке через WEB-форму http://intsysmagazine.ru/generator_form .
2. К статье прилагаются файлы, содержащие название статьи на русском и английском языках, аннотацию на русском и английском языках (не более 50 слов), список ключевых слов на русском и английском языках (не более 20 слов), информация об авторах: Ф.И.О. полностью, место работы, должность, ученая степень и/или звание (если имеется), для аспирантов ФИО научного руководителя, контактные телефоны (с кодом города и страны), e-mail, почтовый адрес с индексом города (домашний или служебный).
3. Список литературы оформляется в едином формате, установленном системой Российского индекса научного цитирования. Список на русском языке приводится в конце файла с текстом статьи, в то время как список, переведённый на английский язык, прилагается отдельным файлом.
4. За публикацию статей в журнале «Интеллектуальные системы. Теория и приложения» с авторов (в том числе аспирантов высших учебных заведений) статей, рекомендованных к публикации, плата не взимается. Авторам бесплатно предоставляется номер журнала, в котором вышла статья. Журнал распространяется по подписке, экземпляры журнала рассылаются подписчикам наложенным платежом. Условия подписки публикуются в каталоге НТИ «Роспечать», индекс журнала 64559.
5. Доступ к электронной версии последнего вышедшего номера осуществляется через НЭБ «Российский индекс научного цитирования». Номера, вышедшие ранее, размещаются на сайте

http://intsysmagazine.ru,

и доступ к ним бесплатный. Там же будут размещены полные тексты всех публикуемых статей.

Подписано в печать: 25.06.2025
Дата выхода: 10.07.2025
Тираж: 200 экз.
Цена свободная
Свидетельство о регистрации СМИ: ПИ № ФС77–58444 от 25 июня 2014 г., выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций(Роскомнадзор).