Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина
Уральский энергетический институт

В. М. ИВАНОВ

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

Учебное пособие

Министерство образования и науки Российской Федерации
Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина
Уральский энергетический институт
Кафедра прикладной математики

В. М. Иванов

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

Учебное пособие

Рекомендовано
методическим советом УрФУ
для студентов, обучающихся по направлению
231300 — Прикладная математика

Екатеринбург
Издательство Уральского университета
2015

УДК 004.8 (075.8)
ББК 312.813 я73
И20

Рецензенты:
Титов С. С. — д-р физ.-мат. наук, проф., зав. кафедрой прикладной математики и технической графики Уральской государственной архитектурно-художественной академии;
Попов Л. Д. — д-р физ.-мат. наук, ведущий научный сотрудник отдела математического программирования Института математики и механики имени академика Н. Н. Красовского УрО РАН

Научный редактор — проф., д-р физ.-мат. наук *А. Н. Сесекин*

Иванов, В. М.
И20 Интеллектуальные системы : учебное пособие / В. М. Иванов. — Екатеринбург : Изд-во Урал. ун-та, 2015. — 92 с.

ISBN 978-5-7996-1325-9

Учебное пособие разработано на основе государственного образовательного стандарта высшего профессионального образования по направлению подготовки 231300 «Прикладная математика» с профилем подготовки «Применение математических методов к решению инженерных задач». Излагаются сведения по теории искусственного интеллекта, которые следует изучить при подготовке к испытаниям, установленным учебным планом.

УДК 004.8 (075.8)
ББК 312.813 я73

ISBN 978-5-7996-1325-9

Оглавление

ГЛАВА 1
Введение в теорию искусственного интеллекта

1.1. О понятии «искусственный интеллект»

Термин «искусственный интеллект» появился в научном обиходе в начале 60-х годов XX в., но всемирное признание он получил только в 1969 г., когда в Вашингтоне была собрана представительная конференция специалистов, работающих в области использования электронно-вычислительных машин (ЭВМ) для моделирования творческих процессов человека. Она называлась «Международная объединенная конференция по искусственному интеллекту».

У разных авторов можно встретить свои определения понятия «искусственный интеллект». Но, пожалуй, все они сводятся, так или иначе, к одному: *искусственный интеллект — это искусственная система, имитирующая решение человеком сложных задач в процессе его жизнедеятельности*. Такое определение содержится в «Словаре по кибернетике», изданном в Киеве в 1979 г. под редакцией академика В. М. Глушкова. Ключевым словом в этом определении является слово «имитирующая»: системы искусственного интеллекта способны подобно человеку решать сложные задачи, но решают их, возможно, не так, как это делает человек. Именно это определение из «Словаря по кибернетике» используется в качестве рабочего определения в настоящем курсе.

В качестве примера приведем другие определения понятия «искусственный интеллект», которые даны учеными, стоявшими у истоков этого научного направления или учеными, активно работающими в этой научной области.

«Искусственный интеллект — это наука о концепциях, позволяющих вычислительным машинам делать такие вещи, которые у людей выглядят разумными ...Центральные задачи искусственного интеллекта состоят в том, чтобы сделать вычислительные машины более полезными и понять принципы, лежащие в основе интеллекта...» [1].

«Искусственный интеллект ставит перед собой серьезную задачу построения теории интеллекта, базирующейся на обработке информации. Если бы такую теорию интеллекта можно было создать, с ее помощью можно было бы направленно вести разработку интеллектуальных машин. Кроме того, мож-

но было бы прояснить детали интеллектуального поведения, проявляющиеся у людей и животных» [2, С. 12].

«Какова цель искусственного разума? ...Эта цель состоит в создании таких программ для вычислительных машин, поведение которых мы бы назвали «разумным»...» [3, С. 7].

«Искусственный интеллект — это программная система, имитирующая на компьютере мышление человека» [4, С. 9].

«Задачей этой науки является воссоздание с помощью искусственных устройств (в основном с помощью ЭВМ) разумных рассуждений и действий» [5, С. 10].

«Сейчас к искусственному интеллекту принято относить ряд алгоритмов и программных систем, отличительным свойством которых является то, что они могут решать некоторые задачи так, как это делал бы размышляющий над их решением человек» [6, С. 11].

Этот перечень определений может быть продолжен. Резюмируя сказанное, еще раз отметим, что искусственный интеллект — это искусственная система, имитирующая интеллектуальную деятельность человека. Это означает, что такая система способна выполнять некоторые интеллектуальные функции человека, но выполняет их не так, как это происходит в мозгу человека, т. е. система имитирует работу мозга.

Имитация интеллектуальной деятельности человека может быть осуществлена разными способами. В связи с этим назовем три основных направления исследований в искусственном интеллекте: *эвристическое* (или *информационное*), *бионическое* и *эволюционное*.

1.2. Направления исследований в искусственном интеллекте

Эвристическое, *или информационное*, направление исследований в искусственном интеллекте включает специалистов, занимающихся созданием машинных способов решения интеллектуальных задач, а также созданием программ для вычислительных машин, решающих такие задачи. При этом как будут устроены подобные программы, насколько близки или далеки будут те способы, которыми они достигают поставленной цели по сравнению с человеческими способами, абсолютно не имеет никакого значения. Главное — конечный результат, его совпадение с результатом, получаемым человеком при решении той же задачи. Приемы и методы, которыми получаются такие же результаты, могут быть совсем иными, не теми, что реально использует человек.

Бионическое направление исследований в искусственном интеллекте изучает процессы, *протекающие в мозгу человека*, *когда он решает задачи*. Программы

для вычислительной машины создаются для имитации процессов получения результатов решения у человека и для изучения этих процессов. Ученые, работающие в бионическом направлении, пытаются воссоздать техническими средствами сам объект, в котором бы протекали процессы, схожие с психическими процессами, проявляющимися у человека во время решения задач. Такие исследователи специально конструируют сети искусственных нейронов и другие аналоги, присущие нервной системе человека.

Наконец, третье, *эволюционное направление*, *занимается созданием программ*, *способных самообучаться тому*, *чего они раньше не умели делать*. Его представители считают, что интеллектуальные программы надо «выращивать», как мы выращиваем детей.

Общий подход к исследованиям в любом из перечисленных направлений состоит в выделении и изучении специальных, не изучающихся другими науками объектов. Этими объектами в искусственном интеллекте являются *метапроцедуры*, с помощью которых человек выполняет интеллектуальные действия. Исследователи изучают метапроцедуры человеческого интеллекта и на этом основании создают метапроцедуры технических и программных систем, реализующих интеллектуальные функции человека. Понятно, что метапроцедуры технических и программных систем являются искусственными объектами, созданными самим человеком, т. е. объектами-аналогами интеллектуальных метапроцедур человека, и они вполне могут быть не свойственны его интеллекту.

1.3. Предмет и метод искусственного интеллекта

Искусственный интеллект является отдельным научным направлением, так как имеет свой специфический *предмет исследования — это интеллектуальные метапроцедуры человека и метапрограммы*, *реализующие эти метапроцедуры*, и свои специфические методы изучения этих объектов. В искусственном интеллекте используются все три классических типа методов исследования: *дедуктивные*, *эмпирические* и *описательные*.

Дедуктивные методы используются при создании программ, реализующих метапроцедуры. Программы пишутся с использованием языков программирования, теория которых строится на основе дедуктивных методов. Общая теория программирования тоже является теорией, основанной на дедуктивных методах.

Готовые программы или метапрограммы можно многократно выполнять на ЭВМ, изменяя у них входные данные или внутренние параметры. Это позволяет опытным путем выявлять характеристики метапроцедур, созданных человеком. Такие методы относятся к *эмпирическим методам*.

Наконец, классификация и сравнение различных программ, сбор и систематизация всевозможных эвристик и алгоритмов, применяемых в программах, требуют большой описательной работы. Этот вид деятельности составляет неотъемлемую часть любого исследования в искусственном интеллекте и не может обойтись без использования *описательных методов*. Извлечение необходимых знаний и сведений из книг, статей, описаний и других информационных источников является необходимой предпосылкой успешных научных исследований.

Итак, предмет искусственного интеллекта — это интеллектуальные метапроцедуры, а методы искусственного интеллекта — это весь арсенал известных научных методов: дедуктивных, эмпирических и описательных.

1.4. Основные задачи искусственного интеллекта

Со времени возникновения первых счетных устройств (абак) до первой ЭВМ (в США — ЭНИАК, Джон фон Нейман, Голдстайн, Пенсильванский университет, 1946 год; в СССР — МЭСМ, академик С. А. Лебедев, 1951 год) они использовались в основном для оперирования с числами. Позднее с развитием вычислительной техники появилась возможность обработки символьной информации, текстов и т. п. Этому способствовало также и появление языков программирования. Но и в конце 40-х годов XX в. стали появляться программы, выполняющие творческие функции интеллекта. Первыми такими программами были игровые программы. Например, программы игры в крестики-нолики, шахматы, шашки, го и др.

Первые компьютеры могли выполнять лишь те простейшие интеллектуальные функции, которые хорошо формализуются с помощью некоторого алгоритма. Это объясняется тем, что вычислительные машины неймановского типа основаны на понятиях алгоритма, машины Тьюринга и информации. Раньше первые программы хорошо вписывались в архитектуру Неймана-Тьюринга электронных вычислительных машин. Так что на начальной стадии развития искусственного интеллекта именно игровые задачи послужили основой для исследования интеллектуальных метапроцедур человека. Эти задачи носят невычислительный характер.

В целом возникновение искусственного интеллекта связано с решением следующих классов задач: *игровые задачи* (конец 40-х годов XX в.: крестики-нолики, морской бой, шашки, шахматы, нарды, го...), *задачи доказательства теорем* (середина 50-х годов XX в.: доказательство теорем в исчислении высказываний, в планиметрии, теории групп...; возникает понятие «бэктрекинг», пришедшее в программирование именно из искусственного интеллекта), *задачи распознавания образов* (50–60-е годы XX в.: распознавание изо-

бражений складывается из двух этапов: а) формирование набора признаков, описывающих изображение, б) нахождение решающего правила, по которому изображения относятся к тому или иному классу), *машинный перевод* (середина 50-х годов XX в.: возникает математическая лингвистика; разрабатываются методы автоматического морфологического и синтаксического анализа и синтеза текстов; начинают разделять декларативные и процедурные знания), *задачи автоматического реферирования и информационного поиска* (середина 50-х годов XX в.: возникают методы поиска информации по образцу (типовым конфигурациям)), *задачи сочинения текстов и музыки* (конец 50-х годов XX в.: содержательно эти задачи сводятся к синтезу текстов и музыки по определенным правилам).

Вот примеры «творчества» компьютера по сочинению сказок и стихов.

Сказка

Однажды в одном городе жил царь. Царь не имел детей, поехал царь на охоту. Встретил царь бабу-ягу. Баба-яга дала зелье полезное. Вернулся царь домой. Родился от того зелья сын Иван-царевич. Вырос Иван-царевич. Унесла баба-яга Ивана-царевича. Прожил Иван-царевич у бабы-яги, не знаю сколько, изучил все колдовство. Надумал Иван-царевич уйти от бабы-яги. Баба-яга задала ему задачу достать перо Жар-птицы. Иван-царевич решил задачу, достал перо Жар-птицы. Баба-яга задала задачу достать девицу, Шамаханскую царицу. Иван-царевич решил задачу, привез царицу. Баба-яга задала задачу достать перстень царь-девицы со дна морского. Иван-царевич решил задачу и перстень достал. Иван-царевич взял заработанное. Вернулся Иван-царевич домой.

Стихотворение

Умирающий — в смятеньи.
Вновь, как тень, огни дрожат.
Вновь над бездною движения —
Где-то далеко — душа ...
Крик смертельный рядом, зыбкий,
Тлели в хрустале глаза,
Шелест мечется с улыбкой,
Где-то в чаще небеса.

Сказка сочинена компьютером по мотивам русских народных сказок [7, С. 130], а стихотворение — на основе словаря, составленного по сборнику стихов Осипа Мандельштама «Камень» [7, С. 132].

Естественно перечень задач, обусловивших возникновение и становление нового научного направления, названного «искусственным интеллектом», может быть расширен: например, проблемами создания роботов (робототехника, робототехника), проблемами экспертных консультирующих систем,

задачами автоматического программирования, комбинаторными задачами, задачами составления расписаний и многими-многими другими задачами. Но какие бы задачи искусственного интеллекта не решались, *главными теоретическими проблемами* были и остаются следующие проблемы.

Центральная проблема искусственного интеллекта — это проблема представления знаний в компьютере. Здесь немаловажным является вопрос: «Что́ такое знание?» Следующий вопрос: «Ка́к представлять знания?» — возникает сразу, если мы собираемся использовать их с применением компьютера. Основными *моделями представления знаний* на сегодняшний день являются: *логическая модель*, *семантические сети*, *продукционные системы* и *фреймовые системы*. Решение проблемы представления знаний опирается на исследования в области компьютерной лингвистики и в области компьютерной логики. Компьютерная лингвистика лежит в основе естественно-языкового общения с компьютером и автоматического перевода, а компьютерная логика служит для формализации всего богатства человеческих рассуждений.

Другая главная проблема искусственного интеллекта — это проблема выявления и исследования интеллектуальных метапроцедур человека. Сложность этой проблемы обусловлена специфичностью устройства человеческого мозга, его полушарной асимметрией. Дело в том, что наш мозг состоит из двух полушарий, левого и правого. Левое полушарие строго логично, рационально. Метапроцедуры, характерные для него, могут быть описаны словесно, они могут быть формализованы в четкие алгоритмы, реализуемые на современных компьютерах с архитектурой Неймана-Тьюринга. Правое полушарие имеет дело (можно сказать «мыслит») чувственными образами. Интуитивные рассуждения, озарения, вещие сны и т. п. являются, по-видимому, результатом работы именно правого полушария. Хорошо известен пример вещего сна при открытии Д. И. Менделеевым «Периодического закона» в химии.

Менее известным является пример гениального озарения индийского математика Сринивазы Рамануджана (1887–1920). Ученые, занимавшиеся исследованиями в области теории чисел, долгое время пытались получить точную формулу для величины $p(n)$ — числа способов представления натурального числа в виде суммы натуральных чисел, но всякий раз получались неточные формулы. В одну из них, полученную совместно английским математиком Харди и Рамануджаном, входило составной частью выражение

$$e^{\frac{\pi\sqrt{\frac{2}{3}n}}{q}},$$

где q — натуральное число, по которому производилось суммирование в той формуле. Мучительные поиски долго не приводили Рамануджана к точному результату. Однажды он внес абсолютно «дикое» предложение изменить в формуле указанное выше выражение на следующее:

$$\frac{e^{\frac{\pi\sqrt{\frac{2}{3}\left(n-\frac{1}{24}\right)}}{q}}}{q},$$

а затем еще и продифференцировать его по n. С поправкой Рамануджана формула для $p(n)$ «заиграла» и стала просто точной формулой. Так что выявление метапроцедур правого полушария, изучение их и изучение метапроцедур совместной работы обоих полушарий — одна из важнейших современных проблем искусственного интеллекта.

По прогнозам ученых, дальнейшее развитие исследований в искусственном интеллекте приведет:

- к смене парадигмы ЗНАНИЯ + ВЫВОД парадигмой ЗНАНИЕ + ОБОСНОВАНИЕ, т. е. в интеллектуальных системах будут использоваться методы обоснования и аргументы; современные языки программирования ориентированы на вывод одних знаний из других;
- к совершенствованию инструментальных средств искусственного интеллекта, в частности, языков программирования, ориентированных на обоснование;
- к модернизации архитектуры вычислительных машин пятого и последующих поколений; сейчас модернизация идет в 4-х направлениях: гигантские суперкомпьютеры; нейробионическое направление (сотни тысяч процессоров с программируемой конфигурацией); территориально удаленные компьютеры и базы с высокоскоростными каналами связи; специальные процессоры для обработки зрительных образов и знаний и для проведения рассуждений автономно, т. е. без помощи человека;
- к появлению методов распараллеливания решения задач на уровне архитектурных решений о структуре компьютера и на логическо-теоретическом уровне;
- к появлению новых моделей представления знаний, позволяющих проводить обработку интегрированной информации (символической, текстовой, зрительной, акустической, тактильной ...);
- к синтезу разнотипных экспертных систем, которые будут использоваться совместно для выработки решений, т. е. как консилиум экспертных систем разного типа.

1.5. Теоретические основы искусственного интеллекта

Ключевыми понятиями информатики были и остаются понятия «модель», «алгоритм» и «программа». С появлением в 60-х годах XX в. нового научного направления, называемого искусственным интеллектом, в научный оби-

ход вошли и новые ключевые понятия: «знание», «представление знаний», «планирование», «общение» и другие. Новые программные системы стали более коммуникабельными благодаря тому, что они приобретали интеллектуальные свойства. Такое стало возможным вследствие новых открытий, сделанных в таких науках, как *теория информации*, *математическая логика* (принцип резолюции Эрбрана, реализованный программно Робинсоном), *лингвистика*, *психология* и *теория познания*. Так что теоретическими основами зарождающегося нового научного направления искусственного интеллекта на ранней стадии стали именно эти разделы знаний.

Внутренние потребности искусственного интеллекта привели к возникновению внутри названных разделов знаний новых направлений исследований, связанных с вычислительной техникой: *компьютерная лингвистика*, *компьютерная логика*, *компьютерная наука* (*в целом*), *когнитивная наука*. Понятия и утверждения этих наук служат теоретической основой для исследований в области искусственного интеллекта. *Компьютерная лингвистика* дает средства для естественного языкового общения с ЭВМ и автоматического перевода текстов. *Компьютерная логика* дает средства для формализации знаний и описывает способы использования знаний по таким правилам, как вывод. *Компьютерная наука* (*в целом*) предоставляет инструментальные средства (программные и технические) для обработки знаний. *Когнитивная наука* рассматривается как основа при создании различных моделей представления и использования знаний, а также при выявлении интеллектуальных метапроцедур человека.

1.6. Основные понятия искусственного интеллекта

Любая программная система, создаваемая в рамках искусственного интеллекта, всегда ориентирована на использование знаний. Знания, выраженные на естественном языке, черпаются из книг, статей и других источников и в том виде, в котором содержатся в этих источниках, не могут быть использованы для обработки на компьютере. Требуется выбрать подходящий способ их формализации (представления) для получения возможности обработки знаний на вычислительных машинах. Сама обработка знаний на компьютере заключается в получении по определенным правилам вывода других знаний на основе имеющихся. Первичными базовыми понятиями искусственного интеллекта являются понятия *знание*, *представление знаний* и *вывод*.

Знаниями принято называть хранимую (*в компьютере*) *информацию*, *формализованную в соответствии с определенными структурными правилами*, *которую компьютер может автономно использовать при решении проблем по таким алгоритмам*, *как логические выводы*. Знания можно разделить

на *факты* (фактические знания), *правила* (знания для принятия решений) и *метазнания* (знания о знаниях). Факты указывают обычно на хорошо известные в данной предметной области обстоятельства, например, «Сократ — человек», «лев — хищник», «курс доллара растет». Под правилами подразумеваются знания вида «ЕСЛИ ..., ТО ...», например, «Если некто — человек, то он смертен», «Если курс доллара растет, то рубль обесценивается». Правила позволяют принимать решения, например, сопоставление факта «Сократ — человек» с правилом «Если некто — человек, то он смертен» позволяет принять решение «Сократ смертен». К метазнаниям относятся знания о способах использования знаний и знания о свойствах знаний. Метазнания необходимы для управления базой знаний, логическим выводом, обучением и т. п. Например, принцип резолюции, используемый в механизме вывода языка логического программирования Prolog, является метазнанием, т. е. знанием о том, как использовать знания для получения новых знаний.

Для того чтобы манипулировать всевозможными знаниями о реальном мире с помощью компьютера, необходимо сначала представить их в виде, пригодном для использования на компьютере. Типичными *моделями представления знаний* являются:

- *логическая модель*, основанная на логике предикатов первого порядка и выведении заключений с помощью силлогизма;
- *продукционная система*– это модель, основанная на использовании правил, т. е. утверждений в форме «ЕСЛИ ..., ТО ...»; продукционные модели бывают двух типов: с прямым и обратным выводами,
- *фреймовая система* (frame (англ.) — рамка, каркас); каждый фрейм описывает один объект какой-либо предметной области (экономики, юриспруденции, химии, медицины и т. д.), а конкретные свойства этого объекта описываются в слотах (компонентах фрейма); у каждого фрейма имеется отдельный слот, содержащий процедуру, реализующую вывод на фреймах;
- *сематическая сеть* — это граф, узлы которого соответствуют понятиям и объектам предметной области, а дуги (ребра) графа соответствуют отношениям (взаимосвязям) между объектами; семантические сети легко представляются в виде фремовой системы.

Иногда отдельно рассматривают *модели представления нечетких знаний*, т. е. знаний, о которых нельзя однозначно сказать, истинны они или ложны. Такие модели основаны на использовании модификаций продукционных систем или логической модели.

Наконец, под *выводом подразумевается механизм получения новых знаний на основе имеющихся фактов и правил*. Механизм вывода основан на метазнаниях. Например, в Prolog-системах механизм вывода основан на принципе резолюции.

1.7. О компьютерах пятого поколения

Интенсивное развитие искусственного интеллекта привело к необходимости разработки соответствующих программных и технических средств, поддерживающих решение новых задач на вычислительных машинах. Появились языки программирования высокого уровня LISP и Prolog.

В 1980 г. японскими учеными была объявлена программа создания компьютеров пятого поколения, рассчитанная на 10 лет. Она предусматривала использование новой элементной базы ЭВМ в виде светооптических элементов и биочипов, основанных на применении микроорганизмов и бактерий. А также эта программа предусматривала использование в качестве основных языков разработки программного обеспечения языки LISP и Prolog. Предусматривалось, что базовыми операциями микропроцессоров компьютеров пятого поколения будут не только элементарные арифметические и булевские (логические) операции над битами и байтами, но и операции логического вывода в понимании математической, а точнее, в понимании логики предикатов 1-го порядка.

Именно с появлением интеллектуальных языков программирования типа LISP и Prolog связано введение новой единицы измерения скорости (быстродействия) интеллектуальных ЭВМ — липс (lips — аббревиатура от английской фразы **l**ogical **i**nnferences **p**er **s**econd, означающей «логические выводы в секунду»). Один липс — это один логический вывод в секунду. В настоящее время имеются компьютеры с быстродействием до 10000 липс.

К сожалению, программа разработки компьютеров пятого поколения так и осталась невыполненной, но работа по этой программе способствовала дальнейшему развитию искусственного интеллекта.

ГЛАВА 2
Представление задач на естественном и формализованном языках

2.1. Что такое «представление задачи»

Начнем с самого термина «задача». В качестве синонима слова «задача» будет использоваться слово «проблема». Если попытаться дать определение понятию «задача», то мы столкнемся, пожалуй, с непреодолимыми трудностями. Трудность состоит в том, что, имея ясное интуитивное представление о том, что означает термин «задача», мы вряд ли сможем дать ему строгое определение, не впадая в ситуацию «порочного круга».

Наверное, самое точное определение этого понятия может быть следующее. Нужно сначала сформулировать все задачи, какие только были в прошлом, есть сейчас или возникнут в будущем, а затем сказать, что задача — это любая из приведенных формулировок. Но возможно ли дать формулировки всех задач?! Ответ очевиден: невозможно! Любое другое определение понятия «задача» не дает нам полной уверенности в отсутствии ситуации «порочного круга», когда одно понятие определяется через другие, в определении которых используется исходное определяемое понятие.

Поэтому поступим аналогично тому, как это сделано в геометрии для понятий «точка», «прямая» и «плоскость». Их определения вообще отсутствуют, а под «точкой», «прямой» и «плоскостью» подразумеваются любые объекты с заданными свойствами, перечисленными в аксиомах геометрии.

Опираясь исключительно на интуитивное представление о том, что такое задача, укажем отличительные свойства этого объекта.

1^0. Любая задача имеет отношение к определенной *предметной области*: игра в шахматы, выбор профессии, история, философия, физика, математика, ...Предметную область можно рассматривать как непустое множество X некоторых объектов, релевантных решаемой задаче. Например, при игре в шахматы в качестве предметной области X удобно рассматривать множество всех позиций. В этом множестве X имеется начальная позиция и конеч-

ные позиции, соответствующие пату или мату. Предметная область всегда наделена некоторой иерархической структурой.

2^0. Знания о предметной области X всегда выражены в форме свойств объектов $x \in X$ или в форме допустимых правил оперирования с этим объектами x из предметной области X. Так, в теории чисел нам известны свойства чисел и допустимые операции над числами. В общем случае, говоря о предметной области X некоторой задачи, имеются в виду не только элементы $x \in X$, но и *разрешенные операции* над элементами множества X.

3^0. В любой задаче всегда требуется что-либо найти, вычислить и т. п. Иными словами, любая задача обладает своими *целями*. Целью всякой задачи, по сути, является поиск в предметной области X одного или нескольких объектов x, обладающих заданными свойствами — обозначим набор этих свойств $K(x)$ — или удовлетворяющих заданным ограничениям, которые можно обозначить также $K(x)$. Цели задачи и ограничения иногда диктуют определенную *стратегию управления* поиском решения задачи.

Итак, любая задача характеризуется:

— предметной областью X,

— разрешенными операциями над элементами x предметной области X,

— целями задачи или стратегиями управления поиском решения задачи.

В самом общем виде условия задачи математически могут быть записаны следующим образом:

Найти в заданном множестве $X \neq \varnothing$ *элементы* x,

удовлетворяющие заданным ограничениям $K(x)$.

Задачи, записанные в такой форме, называются *задачами в замкнутой форме*.

В искусственном интеллекте под представлением задачи (*формализацией задачи*) *подразумевается преобразование формулировки исходной задачи в задачу в замкнутой форме*. Так что формализовать задачу — это представить ее в виде трех компонент: предметная область, операции над элементами предметной области и цели или стратегии управления поиском решения. Результатом представления (формализации) задачи является *постановка задачи*.

2.2. Способы и средства представления задач

Большинство задач, с которым мы встречаемся в реальной жизни, не являются полностью определенными. Чаще всего формулировки задач делаются голосом (причем интонация здесь играет не последнюю роль), жестами, рисунками. Основным средством передачи информации при этом служит естественный язык. Однако с точки зрения решения задачи естественный язык обладает существенными недостатками: он *неполон*, *избыточен*, *неоднозначен*, *неточен* и *грамматически некорректен*.

Неполнота естественного языка выражается в отсутствии в нем слов для выражения смысловых нюансов в диалоге или новых понятий. Недостаток выразительных средств языка иногда приводит к невозможности выразить словами отдельные мысли и понятия. Вот, в частности, почему в языке появляются слова иностранного происхождения, т. е. слова, заимствованные из другого языка.

Примером выражения *избыточности языка* является наличие в нем слов-синонимов, когда разные по написанию и звучанию слова выражают одно и то же содержание, например: точный и безошибочный, идентичный и одинаковый и т. д.

Неоднозначность обычного языка выражается в наличии в нем слов, имеющих различный смысл в различных контекстах (коса, лук, изумрудный). Такие слова называются омонимами.

Неточность естественного языка выражается в «размытости» шкалы смысловых интерпретаций слов. Например, слово «зеленый» может означать как «светло-зеленый», так и «темно-зеленый», а также все оттенки зеленого цвета между этими двумя крайними оттенками.

Грамматическая некорректность обычного языка проявляется в наличии исключений во многих правилах языка. Например, в правиле: частица «не» с глаголами пишется раздельно за исключением тех случаев, когда без нее глагол не употребляется, например: ненавидеть.

Наличие такого «букета» недостатков естественного языка, бывает, не только лишает возможности найти нужное решение задачи, но и приводит просто к неверному решению. В этой связи опишем курьезный эпизод, имевший место в действительности в Вычислительном центре Сибирского отделения Академии наук СССР. Кадровой компьютерной системе задали при ее испытании вопрос: «Сколько на ВЦ работает котов?». Она ответила: «Один, Ко́тов, — заместитель директора».

Выполнение постановки задачи в замкнутой форме, т. е. ее формализация, служит именно для того, чтобы лишить формулировку задачи неоднозначности и любых других неточностей и недостатков, свойственных естественному языку.

Существует множество различных способов представления задач в замкнутой форме. Каждый из них основан на использовании средств той или иной формальной системы: алгебры, геометрии, тригонометрии, математического анализа, теории вероятностей, линейного программирования и т. д. В искусственном интеллекте рассматриваются два общих способа представления задач в замкнутой форме безотносительно к конкретной формальной системе.

2.2.1. Первый способ формализации задач

Элементы x предметной области X рассматриваются как состояния некоторого, возможно абстрактного, объекта. Среди всех состояний из множества X выделяются два вида состояний: одни называют *начальными состояниями* $x_{нач}$, другие — *конечными*, или *целевыми*, *состояниями* $x_{кон}$. Остальные состояния играют роль промежуточных состояний.

Над объектом производятся определенные операции, в результате которых объект переходит из одного состояния $x \in X$ в другое $y \in X$. Набор разрешенных операций над состояниями (обозначим их O_1, O_2, ..., O_n) считается конечным.

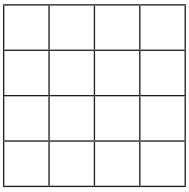
Смысл задачи при таком ее формализованном представлении состоит в том, чтобы перевести объект из заданного начального состояния $x_{нач}$ в одно из конечных состояний $x_{кон}$, используя только разрешенные операции O_1, O_2, ..., O_n. Когда конечные (целевые) состояния невозможно заранее указать явно, как в случае с игрой в шахматы или го, задаются *целевые условия*. Это — условия, однозначно определяющие, какие состояния следует считать конечными. Роль этих условий в шахматах играют те положения Правил игры в шахматы, в которых описываются ситуации окончания игры: мат, пат, вечный шах и др.

Соответствующая задача в замкнутой форме, т. е. задача, формализованная первым способом, представляется теперь так: найти такую последовательность

$$O_{i_1}, O_{i_2}, ..., O_{i_k}$$

разрешенных операций O_1, O_2, ..., O_n, переводящую объект из заданного начального состояния $x_{нач}$ в одно из конечных состояний $x_{кон}$. Или, если конечные состояния не заданы явно, указанная последовательность разрешенных операций должна переводить объект из заданного начального состояния $x_{нач}$ в состояние, удовлетворяющее целевым условиям.

В качестве примера такого способа представления (формализации) задачи рассмотрим игру «Пятнадцать». Имеется квадратное игровое поле размером 4×4



и 15 квадратных фишек, пронумерованных числами от одного до пятнадцати

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

.

Фишки размещаются в клетках игрового поля в некотором порядке, при этом одна клетка игрового поля оказывается незаполненной (пустой), например:

$x_{\text{нач}}=$

3	7	9	11
15		6	5
4	13	1	8
14	2	10	12

.

Требуется, перемещая фишки в пустую клетку и не отрывая при этом их от игрового поля, получить нужный порядок фишек, например, получить расположение фишек в порядке возрастания их номеров:

$x_{\text{кон}}=$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

.

В этой задаче удобно перемещения пронумерованных клеток интерпретировать как перемещения пустой клетки

.

Так что разрешенных операций оказывается всего четыре: движение пустой клетки вправо, влево, вниз и вверх. Обозначим эти операции R, L, D, U соответственно. В формализованном виде постановка задачи теперь имеет следующий вид: найти такую последовательность

$$O_{i_1}, O_{i_2}, \ldots, O_{i_k}$$

операций R, L, D, U движения пустой клетки, переводящих состояние $x_{\text{нач}}$ в конечное состояние $x_{\text{кон}}$.

Таким образом, при первом способе формализации задачи должны быть заданы:

1) множество **P** всех состояний, которое называют *пространством состояний*; в нем должно быть указано начальное состояние;
2) множество всех *разрешенных операций* $O_1, O_2, \ldots, O_n$ над состояниями;

3) множество всех *конечных состояний*; при невозможности задать эти состояния явно задаются условия, однозначно характеризующие целевые состояния — *целевые условия*.

2.2.2. Второй способ формализации задач

Предметная область рассматривается как множество подзадач, к которым может быть сведено решение исходной задачи. В этом множестве должны быть выделены: исходная задача и простейшие (примитивные) подзадачи. *Примитивными подзадачами* называются такие задачи, решения которых известны заранее. Например, в задаче вычисления неопределенного интеграла примитивными следует считать задачи на вычисление табличных интегралов.

Способы сведения исходной задачи к подзадачам называются *правилами декомпозиции*. Набор правил декомпозиции (обозначим их $П_1$, $П_2$, ..., $П_n$) считается конечным. Применение того или иного правила декомпозиции к некоторой задаче сводит ее решение к одной или нескольким, вообще говоря, более простым подзадачам. Примерами правил декомпозиции в задаче вычисления неопределенного интеграла могут служить как свойства неопределенного интеграла, так и известные методы его вычисления.

Соответствующая задача в замкнутой форме, т. е. формализованная задача, представляется теперь так: найти такую последовательность

$$П_{i_1}, П_{i_2}, ..., П_{i_m}$$

правил декомпозиции $П_1$, $П_2$, ..., $П_n$, сводящую решение исходной задачи к решению одной или нескольких примитивных подзадач. Когда примитивные подзадачи невозможно заранее указать явно, как в случае с задачей на упрощение алгебраического выражения, так как неизвестно, как должно выглядеть окончательное упрощенное выражение, задаются *целевые условия*. Это — условия, однозначно определяющие, какие задачи следует относить к примитивным подзадачам.

Часто при использовании второго способа формализации задачи незаданной оказывается сама цель, например, в задаче: упростить выражение

$$\sum_{k=1}^{n} \sin kx.$$

В задачах, представленных вторым способом, искусство математика как раз и заключается в том, чтобы отыскать те правила декомпозиции, которые окажутся полезными для данного условия задачи. Более того, ему прежде нужно установить для себя цель задачи, если таковая не указана явно в постановке задачи.

При втором способе формализации задачи должны быть заданы:

1) множество **P** всех подзадач, к которым может быть сведено решение исходной задачи; это множество называют *пространством подзадач*; в нем должны быть указаны исходная задача, которую следует решить, и простейшие (примитивные) подзадачи, к которым может быть сведено решение исходной задачи и решения которых известны заранее;
2) *множество правил декомпозиции* $П_1$, $П_2$, ..., $П_n$, позволяющие сводить решение одних задач к решению других задач;
3) *множество примитивных подзадач*; при невозможности явно указать эти подзадачи задаются условия, однозначно характеризующие примитивные подзадачи; такие условия называют *целевыми условиями*.

Пространство состояний в задачах, формализованных первым способом, можно представить графически в виде ориентированного графа. Тогда решение задач, представленных первым способом, сводится к поиску решающего пути на графе, ведущему из начального состояния в одно из конечных состояний. При этом могут использоваться различные стратегии поиска: поиск в глубину, поиск в ширину, эвристический поиск, т. е. поиск, направляемый оценочной функцией или отношением предпочтения.

Пространство подзадач для задач, формализованных вторым способом, можно представить графически в виде ориентированного графа особого типа — И-ИЛИ дерева. Тогда решение задач, представленных вторым способом, сводится к поиску на И-ИЛИ дереве решающего поддерева, в котором корневая вершина соответствует исходной задаче, а листья И-ИЛИ дерева соответствуют примитивным подзадачам. Общий метод (стратегия) решения здесь — сведение к подзадачам, каждая из которых может быть представлена своим способом решения. При этом могут использоваться те же стратегии поиска, что и для решения задач, формализованных первым способом: поиск в глубину, поиск в ширину или эвристический поиск, например, алгоритм программы *GPS* (General problem solver — *Общий решатель задач*), предложенной А. Ньюэллом, К. Шоу и Г. Саймоном [8, С. 289].

Пространство состояний и пространство подзадач называют *пространствами поиска*, в которых ищется решение задачи. При графическом представлении пространств поиска *решение ищется в форме решающего пути при первом способе представления задач и в форме решающего поддерева при втором способе представления задач*.

2.2.3. Графическое представление пространства состояний

При первом способе представления задачи считается заданным пространство состояний *X*. Среди состояний выделены начальные состояния (обычно одно) и целевые состояния (может несколько) или, если целевые состоя-

ния не заданы явно, указаны целевые условия. Заданными считаются также операции перехода — разрешенные операции — из одного состояния в другое. Пусть A — начальное состояние, Z — одно из конечных состояний. Тогда задача в замкнутой форме (постановка задачи) имеет следующий вид: *построить последовательность разрешенных операций*, *переводящих состояние A в состояние Z.*

Представим пространство состояний в виде ориентированности графа следующим способом. Считаем, что вершины графа соответствуют состояниям из множества X, а ребра графа соответствуют разрешенным операциям из заданного конечного набора $\{O_1, O_2,..., O_n\}$. Ребро графа направляем из вершины U в вершину V, если среди разрешенных операций перехода имеется такая, которая переводит состояние U в состояние V. Тогда решение исходной задачи сводится к построению на графе, который представляет пространство состояний, пути, ведущего из начальной вершины A в одну из конечных вершин Z. Такой путь называется *решающим путем*.

Для иллюстрации представления задачи, формализованной первым способом, в виде ориентированного графа рассмотрим игру «Восемь». Она полностью аналогична игре «Пятнадцать». Игра «Восемь» имеет квадратное игровое поле размером 3×3

и 8 квадратных фишек, пронумерованных числами от одного до восьми

1	2	3	4	5	6	7	8

.

Фишки размещаются в клетках игрового поля в некотором порядке, при этом одна клетка игрового поля оказывается незаполненной (пустой), например:

$x_{\text{нач}} =$

2	8	3
1	6	4
7		5

.

Требуется, перемещая фишки в пустую клетку и не отрывая при этом их от игрового поля, получить нужный порядок фишек, например, получить расположение фишек в следующем порядке:

$$x_{\text{кон}} = \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 8 & & 4 \\ \hline 7 & 6 & 5 \\ \hline \end{array}\ .$$

В этой задаче удобно перемещения пронумерованных клеток интерпретировать как перемещения пустой клетки в позицию соответствующей пронумерованной клетки

$$\square\ .$$

Так что разрешенных операций оказывается всего четыре: движение пустой клетки вправо, влево, вниз и вверх. Обозначим эти операции R, L, D, U соответственно.

Под состоянием в этой задаче будем понимать матрицу размерности 3×3. Ее элементами могут быть только цифры 0, 1, 2, 3, 4, 5, 6, 7, 8. Все элементы различны. Цифра 0 соответствует пустой позиции игрового поля, остальные располагаются в матрице соответственно фишкам с теми же номерами. Так что движению пустой клетки на игровом поле соответствует такое же движение цифры 0 в матрице: она попросту меняется местами с той или иной цифрой. В формализованном виде постановка задачи теперь имеет следующий вид: найти такую последовательность

$$O_{i_1}, O_{i_2}, ..., O_{i_k}$$

операций R, L, D, U движения цифры 0, переводящих начальное состояние

$$x_{\text{нач}} = \begin{pmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 0 & 5 \end{pmatrix}$$

в конечное состояние

$$x_{\text{кон}} = \begin{pmatrix} 1 & 2 & 3 \\ 8 & 0 & 4 \\ 7 & 6 & 5 \end{pmatrix}.$$

Фрагмент ориентированного графа, представляющего пространство состояний в данной задаче, приведен на рис. 1. На этом фрагменте графа пространства состояний игры «Восемь» можно увидеть решающий путь, ведущий из начального состояния в конечное состояние. Этот путь состоит из выделенных дуг графа.

2.2.4. Графическое представление пространства подзадач

В задачах, формализованных вторым способом, предметная область X представляет собой множество (пространство) подзадач, к которым может быть сведено решение исходной задачи. В пространстве подзадач указываются исходная задача, которую следует решить, примитивные подзадачи, к которым следует сводить решение исходной задачи и решения которых известны заранее. Кроме того, имеются правила декомпозиции задач. Иногда вместо примитивных подзадач формулируются целевые условия, описывающие примитивные подзадачи.

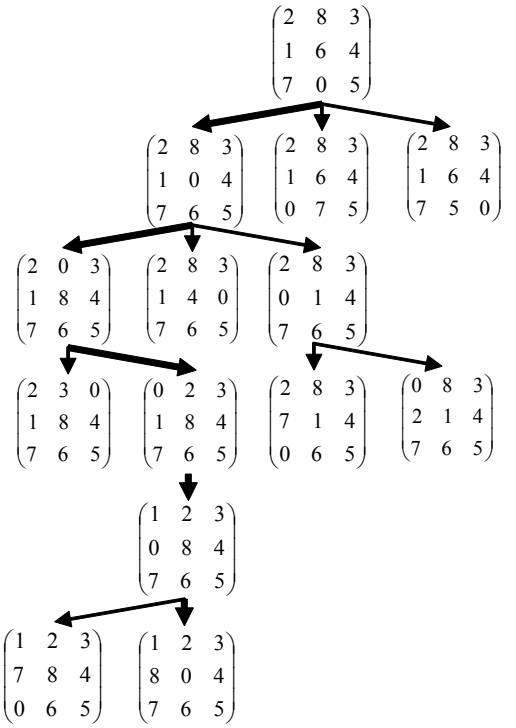


Рис. 1

Пространство подзадач можно представить графически в виде *И-ИЛИ* дерева следующим способом. Считаем, что вершины графа соответствуют подзадачам из пространства подзадач X, а ребра графа соответствуют правилам декомпозиции, выбираемым из заданного конечного набора $\{П_1, П_2, ..., П_n\}$. Корневой вершиной (в нее не входит ни одно ребро графа) является вершина, соответствующая исходной задаче. Висячими вершинами, «листьями», графа (т. е. вершин, из которых нет исходящих ребер графа) будут тупиковые вершины или вершины, соответствующие примитивным подза-

дачам. Если текущая вершина соответствует задаче P_0 и имеется правило декомпозиции, которое сводит решение задачи P_0 к решению подзадач P_1, P_2, ..., P_n, то из текущей вершины P_0 исходит n направленных ребер: по одному ребру к каждой из упомянутых подзадач P_1, P_2, ..., P_n.

Ребра стягиваются у вершины P_0 общей сплошной линией, если для решения задачи P_0 нужно решить все задачи P_1, P_2, ..., P_n (рис. 2, *а*). Вершина P_0 в этом случае называется *И-вершиной*.

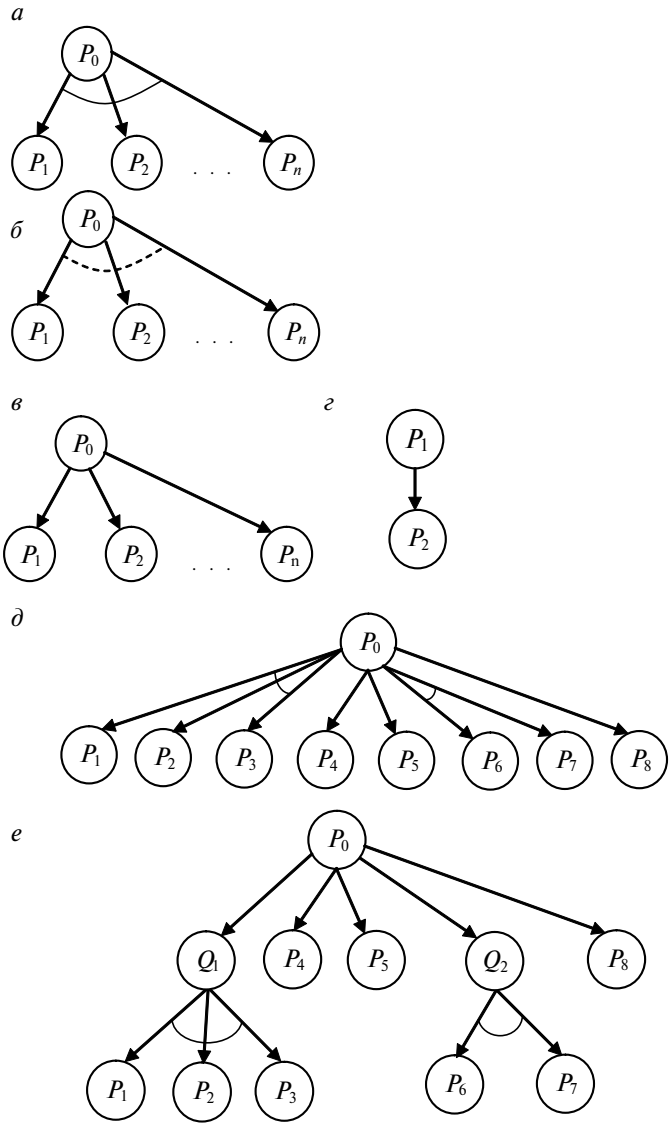


Рис. 2

Если для решения задачи P_0 достаточно решить только одну из задач P_1, P_2, ..., P_n, то ребра стягиваются общей пунктирной линией или вообще не стягиваются общей линией (см. рис. 2, *б* или рис. 2, *в*). Вершина P_0 в этом случае называется *ИЛИ-вершиной*.

Ребро графа направляем из вершины P_1 в вершину P_2, если имеется правило декомпозиции, которое сводит решение задачи P_1 к решению одной подзадачи P_2 (см. рис. 2, *г*). Вершину P_1 в этом случае будем относить к *И*-вершинам.

При построении графа пространства подзадач могут возникать висячие вершины, «листья», из которых не исходит ни одного ребра. Среди них имеются *тупиковые вершины*, т. е. висячие вершины, из которых не исходит ни одного ребра из-за отсутствия правил декомпозиции для задач, соответствующих этим вершинам. Но может быть и другая ситуация, когда из вершины не исходит ни одного ребра. Эта ситуация возникает для конечных (целевых) вершин, которые соответствуют примитивным подзадачам.

Ориентированный граф называется *И-ИЛИ деревом*, если все его вершины — это *ИЛИ*-вершины либо *И*-вершины.

Могут быть более сложные случаи декомпозиции (см. рис. 2, *д*). Здесь вершина P_0 не является ни *И*-вершиной, ни *ИЛИ*-вершиной. Введение вспомогательных *И*-вершин Q_1 и Q_2 позволяет вершину P_0 преобразовать в *ИЛИ*-вершину (см. рис. 2, *е*). Этим обеспечивается преобразование графа, представляющего любое пространство подзадач, в *И-ИЛИ* дерево, в котором каждая вершина является *ИЛИ*-вершиной либо *И*-вершиной.

Для *И-ИЛИ* деревьев имеется обширный арсенал алгоритмов построения решающих поддеревьев. Этим и диктуется необходимость в получении именно *И-ИЛИ* дерева, представляющего пространство подзадач. Определим точнее понятие «решающее поддерево».

Начнем с понятия «разрешимая вершина». *И*-вершина называется *разрешимой*, если все ребра, исходящие из нее, направлены к разрешимым вершинам. *ИЛИ*-вершина называется *разрешимой*, если среди всех ребер, исходящих из нее, имеется хотя бы одно, приводящее к разрешимой вершине. Вершина, соответствующая простейшей задаче, считается разрешимой. Тупиковая вершина, т. е. вершина, из которой не исходит ни одного ребра, считается *неразрешимой*. *Решающим поддеревом И-ИЛИ* дерева, представляющего пространство подзадач, называется подграф этого *И-ИЛИ* дерева, у которого корневая вершина соответствует исходной задаче, все промежуточные вершины разрешимы и все висячие вершины тоже разрешимы.

В качестве примера представления задачи деревом подзадач рассмотрим следующую задачу:

Вычислить интеграл $\int \frac{x^4 dx}{(1-x^2)^{\frac{5}{2}}}$.

В качестве правил декомпозиции будем использовать:
$П_1$ — алгебраические подстановки, например: $t^2 = 1 - x^2$,
$П_2$ — тригонометрические подстановки, например: $x = \sin t$,
$П_3$ — тригонометрические тождества, например: $\operatorname{tg} x = (\operatorname{ctg} x)^{-1}$,
$П_4$ — деление числителя на знаменатель, например:

$$\frac{z^4}{1+z^2} = z^2 - 1 + \frac{1}{1+z^2},$$

$П_5$ — выделение полного квадрата, например:

$$x^2 - 4x + 13 = (x-2)^2 + 9,$$

$П_6$ — разбиение интеграла на сумму или разность интегралов, например:

$$\int\left(z^2 - 1 + \frac{1}{1+z^2}\right)dz = \int z^2 dz - \int 1 dz + \int \frac{dz}{1+z^2}.$$

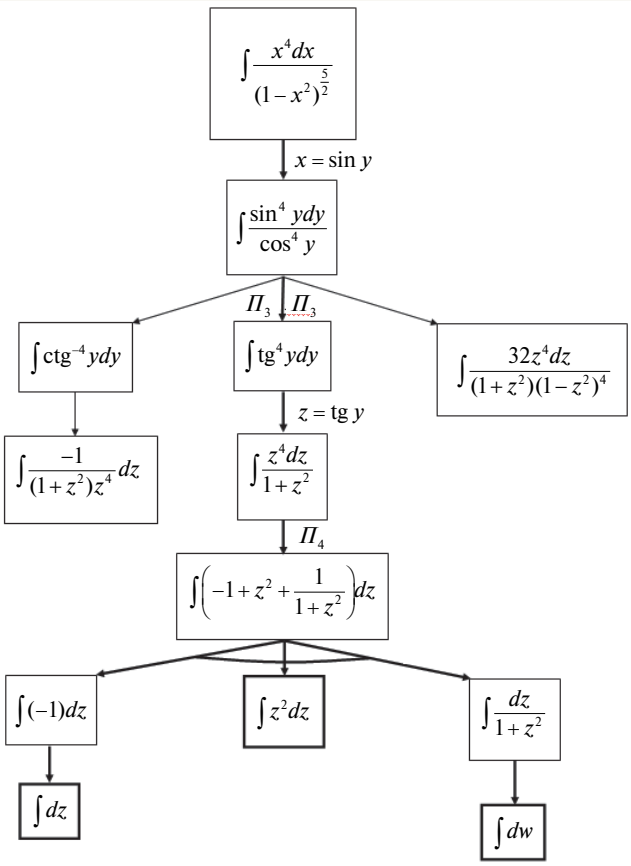


Рис. 3

Простейшими подзадачами считаются все задачи на вычисление табличных интегралов. Построим фрагмент дерева подзадач для исходной задачи (см. рис. 3). На нем вершины представляют собой выражения, которые необходимо проинтегрировать. Выделенные ребра этого дерева показывают дерево решения этой задачи — решающее поддерево. Выделенные вершины соответствуют табличным интегралам (простейшим подзадачам). Ребра дерева помечены либо формулой подстановки, либо обозначением соответствующего правила декомпозиции. Используя это дерево и табличные интегралы, найдем ответ этой задачи.

Итак, выше показано, что формализация задачи первым или вторым способом приводит к представлению ее в виде графа или *И-ИЛИ* дерева. Поиск решения задачи, следовательно, представляет собой поиск решающих путей на ориентированных графах или решающих поддеревьев в *И-ИЛИ* деревьях.

2.3. Общий подход к решению задачи

Подход человека к решению задачи включает, как правило, семь основных этапов:

1. Выяснение смысла условий задачи

Это существенно зависит от наших органов чувств (слуха и зрения). Человек обладает ограниченными способностями к немедленному запоминанию, поэтому нужно более или менее длительное время для осмысления условий задачи.

2. Первые выводы из условий задачи

Человек, используя свои знания, восполняет недостающую в условиях задачи информацию. При этом могут строиться рисунки, графы или записываться формулы.

3. Проигрывание ситуации

На этом этапе проверяется, что ничто не пропущено и нет существенных ошибок в интерпретации условий задачи. Определяется, в чем же сложность задачи.

4. Обдумывание

Это основной этап, в ходе которого отыскивается предметная область, устанавливаются допустимые операции над ее элементами и четко формулируются цели задачи для получения наилучшей формализации задачи, облегчающей поиск ее решения.

5. Выбор наилучшего представления задачи — поиск замкнутой формы задачи

Задаче придается полный однозначный и безызбыточный вид в форме формализованной постановки задачи. Оценивается трудность решения задачи, исходя из «объема» пространства поиска и сложности ограничений.

6. Частичное (возврат к этапу 2) или общее решение задачи

Осуществляется поиск решения задачи и, если необходимо, постановка задачи уточняется или выполняется заново (с этапа 2), чтобы сузить пространство поиска.

7. Проверка и обобщение решения.

Обсуждается решение задачи, оценивается адекватность найденного решения исходным условиям задачи. Выясняется существенность тех или иных условий задачи. Определяется поведение решения в особых точках. Оценивается общность использованного метода решения.

2.4. Стратегии и процедуры решения задачи

Под *стратегией решения задачи* будем подразумевать общее правило (метапроцедуру) решения широкого круга задач. При непосредственном решении задачи любая стратегия решения реализуется в форме выполнения отдельных шагов (этапов) решения. По сути, стратегия решения — это некоторая метапроцедура, управляющая последовательностью выполнения отдельных процедур. Под *процедурой решения задачи* понимается каждый отдельный этап решения, а точнее, совокупность действий, выполняемых на этом этапе.

Если имеется некоторый набор процедур решения конкретной задачи $\{P_1, P_2,..., P_n\}$, то собственно решение задачи представляет собой выполнение последовательности процедур $P_{i_1}, P_{i_2},..., P_{i_k}$, где $P_{i_1}, P_{i_2},..., P_{i_k} \in \{P_1, P_2, ..., P_n\}$. А какие именно процедуры и в какой последовательности они должны выполняться, определяется выбранной стратегией решения задачи.

Перечислим некоторые стратегии решения задач. Ряд стратегий решения задач, направленных на построение решающего пути на графе, основываются на так называемой оценочной функции. *Оценочная функция* определяется на вершинах графа и, как правило, принимает в качестве своих значений вещественные числа. Для произвольной вершины значение оценочной функции указывает на степень предпочтительности продолжения поиска решающего пути из этой вершины. Чем предпочтительнее вершина, тем более перспективной является она для применения к ней поисковых процедур. Стратегия решения задачи, направляемая оценочной функцией, является разновидностью *эвристического поиска*.

Другой разновидностью стратегии решения, направляемой оценочной функцией, является поиск в глубину и поиск в ширину. При *поиске в глубину* значение оценочной функции любой вершины прямо пропорционально «расстоянию» от этой вершины до начальной вершины. При *поиске в ширину* эта зависимость обратно пропорциональна. В обоих случаях расстояние между вершинами можно измерять количеством ребер графа в кратчайшем пути, связывающих эти вершины.

Заметим, что оценочная функция не обязательно должна быть числовой функцией. Можно между вершинами установить некоторое качественное упорядочение. В остальном принципы работы с качественной оценочной функцией те же, что и с количественной.

Перечислим другие стратегии решения задач. Для поиска на графе, соответствующем пространству состояний (при первом способе формализации), используются: метод ветвей и границ, алгоритм кратчайших путей Мура, алгоритм Дейкстры, алгоритм Дорана и Мичи поиска с низкой стоимостью, алгоритм Харта, Нильсона и Рафаэля. Для поиска на *И-ИЛИ* дереве, соответствующем пространству подзадач (при втором способе формализации), используются: алгоритм Ченга и Слейгла, метод ключевых операторов, метод планирования общего решателя задач (GPS), дедуктивные методы на основе принципа резолюции, методы продукций.

ГЛАВА 3
Стратегии решения задач

3.1. Формализованное представление задачи

Выше были описаны два общих способа формализации задач. Первый способ основан на представлении задачи пространством состояний некоторого, возможно условного, объекта. Для этого пространства задаются процедуры допустимых переходов из одного состояния в другое, другими словами, разрешенные операции над состояниями, не выводящие за пределы этого пространства. Смысл решения задачи, формализованной первым способом, заключается в построении последовательности разрешенных операций, переводящих объект из одного, начального, состояния в некоторое целевое, конечное, состояние. Если пространство состояний представлено ориентированным графом, то для решения задачи следует построить решающий путь, ведущий из начального состояния в одно из конечных состояний.

Второй способ основывается на представлении задачи пространством подзадач. Для пространства подзадач задаются правила декомпозиции задач и подзадач, позволяющие сводить решение одних задач к решению одной или нескольких других задач. Смысл решения задачи, формализованной вторым способом, заключается в построении последовательности правил декомпозиции, сводящих решение исходной задачи к решению других, примитивных, подзадач, решения которых известны заранее. Если пространство подзадач представлено *И-ИЛИ*-деревом, то для решения задачи следует построить решающее поддерево, в котором единственная корневая вершина соответствует исходной задаче, а все висячие вершины соответствуют примитивным подзадачам.

Под *стратегией решения задачи* понимается общее правило решения широкого круга задач. Это правило направляет поиск решающего пути или решающего поддерева, если пространства поиска представлены графами. Ниже на примере задачи, формализованной первым способом, дается описание стратегий решения задач: поиск в глубину, поиск в ширину и эвристический поиск.

3.2. Стратегия поиска в глубину

Пусть некоторая условная задача формализована первым способом. Это означает, что заданы

- пространство состояний, в котором указаны начальное состояние и одно или несколько конечных состояний;
- разрешенные операции над состояниями;
- целевые условия, если конечные состояния не указаны явно; целевые условия однозначно характеризуют, какие состояния следует считать конечными.

Так, например, в шахматной игре пространство состояний можно считать состоящим из возможных позиций на шахматной доске. Разрешенные операции — это разрешенные в соответствии с правилами шахматной игры ходы, переводящие одну позицию в другую. Если рассматривается обычная партия в шахматы между двумя игроками, то роль начальной позиции (начального состояния) играет известная стандартная начальная позиция. Конечной, целевой, позицией является любая из позиций, соответствующих окончанию партии. Условия окончания партии также описываются в правилах шахматной игры и, по сути, являются целевыми условиями. В частности, к целевым относятся условия, описывающие позиции мата, пата, вечного шаха и т. д.

Возвращаясь к условной задаче, будем считать, что построен ориентированный граф пространства состояний. Припишем каждому ребру этого графа число 1, которое называют «длиной» этого ребра. Иногда это число называют «глубиной» или «расстоянием». Содержательно «длина» может выражать в зависимости от задачи вес (кг), цену (руб.), длину (м), затраченную энергию (кВт) и т. д. Число 1 в рассматриваемой условной задаче интерпретируется как число операций перехода из одного состояния в другое.

Идея *алгоритма поиска в глубину* состоит в следующем: *начальная вершина графа принимается за начало решающего пути; а далее всегда*, *когда надлежит выбрать из нескольких альтернативных вершин ту*, *в которую следует перейти из текущей вершины для продолжения поиска решающего пути*, *нужно выбрать самую «глубокую» из них*, *т. е. ту*, *которая расположена дальше других от начальной вершины*. Термины «самая глубокая» и «дальше» здесь следует понимать в смысле «длины» ребра, определенной выше.

При поиске в глубину в случае, когда самых глубоких вершин оказывается несколько, можно выбрать любую из них, например, самую «левую». Здесь термин «левый» — условный термин, означающий в одних случаях первую из альтернативных вершин в порядке их обнаружения, в других случаях на самом деле визуально самую левую из альтернативных вершин, если граф реально представлен графически. В иных случаях термин «левый» мо-

жет означать предпочтения, продиктованные условиями задачи или, вообще, самыми разными соображениями.

Если поиск приводит в тупик, т. е. к вершине, не являющейся целевой и не имеющей связей с более глубокими вершинами, то следует вернуться в предыдущую вершину и продолжить из нее поиск в глубину. Процедура возвращения назад к предыдущей вершине при неудаче — попадании в тупиковую вершину — называется *возвратом* (*бэктрекингом*) или *откатом*.

На рис. 4, *а* в качестве примера показано, в каком порядке алгоритм поиска в глубину проходит вершины пространства состояний. Здесь вершина *a* соответствует начальному состоянию, вершины *j* и *f* соответствуют конечным состояниям.

Выделенные пути [*a*, *b*, *e*, *j*] и [*a*, *c*, *f*] — это найденные решающие пути, и [*a*, *c*, *f*] — кратчайший решающий путь, отыскиваемый, если продолжать поиск в глубину до полного перебора всех вершин графа.

Поиск в глубину часто работает хорошо, как в рассмотренном примере на рис. 4, *а*. Но бывает, что он дает сбои, например, зацикливание, если не предусмотрен анализ наличия циклов в ориентированном графе пространства состояний решаемой задачи. На рис. 4, *б* представлен граф, содержащий циклы [*d*, *h*, *d*] и [*b*, *e*, *i*, *b*]. Включение в алгоритм поиска в глубину механизма обнаружения циклов позволит избежать ситуаций зацикливания. Механизм обнаружения циклов должен работать так, чтобы ни одна из вершин, уже содержащихся в построенном пути, не рассматривалась вторично в качестве возможной альтернативы продолжения поиска.

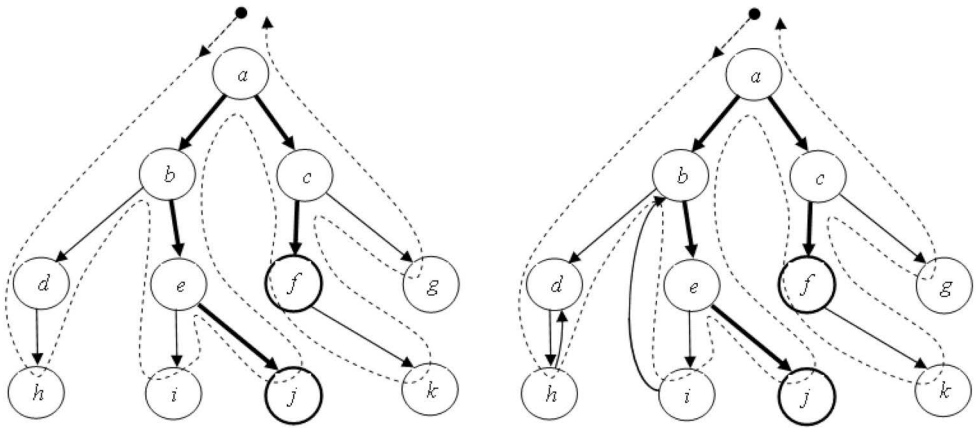


Рис. 4

Рассмотрим пример использования стратегии поиска в глубину при решении задачи переупорядочивания кубиков. Задача состоит в выработке плана перестановки кубиков, поставленных друг на друга (см. рис. 5). Слева ука-

зано начальное состояние, а справа — конечное состояние, к которому следует прийти. На каждом шаге решения можно переставлять только одни кубик. Кубик можно переставлять только тогда, когда на нем не стоит другой кубик. Кубик можно поставить либо на стол, либо на другой кубик.

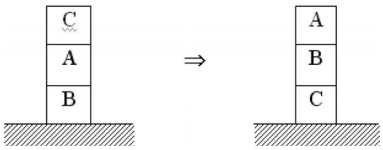


Рис. 5

Пространство состояний этой задачи представлено на рис. 6. Обозначения состояний в вершинах этого графа указывают на расположение кубиков относительно друг друга. Двунаправленные стрелки обозначают циклы, образованные соответствующими смежными вершинами.

Указанный на рис. 6 путь обхода вершин соответствует стратегии поиска в глубину с механизмом обнаружения циклов и дает решение

$$\begin{bmatrix} C & & & & & & & & & & & A \\ A & A & & & & & B & & B & & & B \\ B, & B & C, & A & B & C, & A & C, & C & A, & C \end{bmatrix}.$$

Кратчайшее решение выделено на рис. 6 жирными стрелками. Оно будет получено дальнейшим продолжением поиска решений, пока не будут найдены все решения. Это продолжение поиска отмечено на рис. 6 пунктиром.

В отдельных случаях поиск в глубину может приводить к полному перебору, прежде чем будет построен решающий путь. Например, если в предыдущей задаче переупорядочивания кубиков с пространством состояний, изображенным на рис. 6, исходным состоянием является то же самое состояние, а целевым — другое состояние (см. рис. 7), то стратегия поиска в глубину приводит к почти полному перебору, прежде чем будет найдено решение. Кратчайшее решение в таком пространстве состояний будет найдено, действительно, полным перебором всех вершин и путей.

Поиск в глубину прост, легко программируется и наиболее адекватен рекурсивному стилю программирования, принятому в языке программирования Prolog. Последнее объясняется тем, что, обрабатывая цели, пролог-система просматривает альтернативы именно в глубину. Поиск в глубину (без механизма обнаружения циклов) может быть запрограммирован на языке Prolog следующим образом [9]:

решить (*B*, *[B]*):–*цель* (*B*).

решить (*B*, *[B | Реш1]*):–*после* (*B*, *B1*), *решить* (*B1*, *Реш1*).

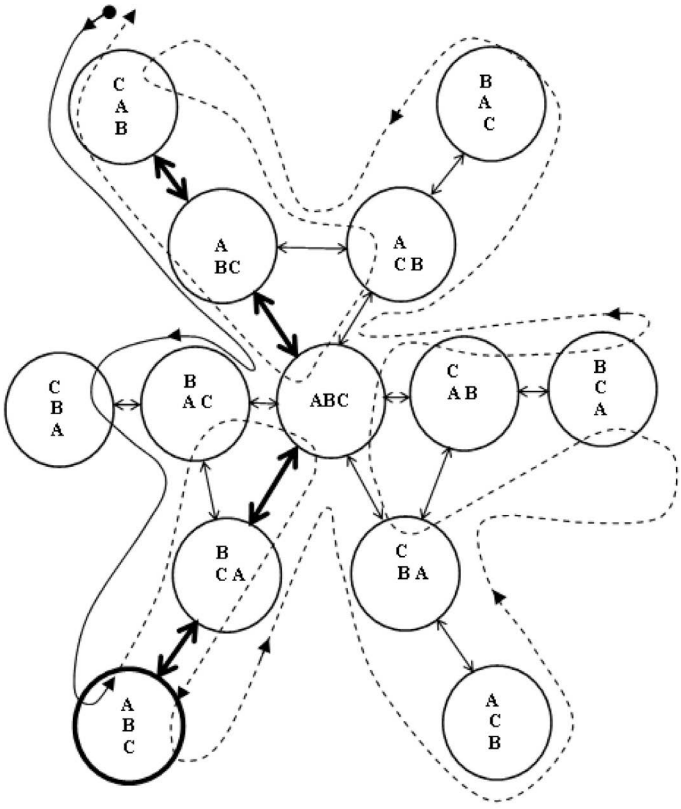


Рис. 6

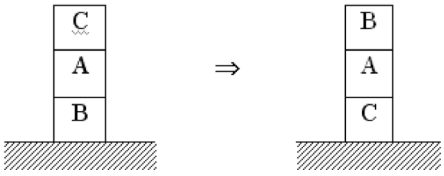


Рис. 7

Этот текст на языке Prolog означает: для того чтобы найти решающий путь *Реш* (последовательность вершин в виде прологовского списка [*B*, B_1, ..., B_k]) из начальной вершины *B* в некоторую целевую вершину B_k (целевые вершины *A* задаются фактами вида «*цель* (*A*).»), необходимо:

— если *B* — это целевая вершина, то положить *Реш* = [*B*],

— если для начальной вершины *B* существует вершина *B*1, в которую направлено ребро графа, выходящее из вершины *B*, то решающий путь *Реш* = [*B* | *Реш*1], где *Реш*1 — решающий путь, начинающийся из вершины *B*1 как из исходной вершины.

Факты вида «*после* (*B*, *B*1).» определяют ребра, направленные из вершины *B* в вершину *B*1. Предикат *решить* (*B*, *Реш*) описывает взаимосвязь между начальной вершиной *B* и решающим путем *Реш*, начинающимся из этой вершины. Прологовская запись *Реш*= [*B* | *Реш*1] означает, что *B* — голова списка *Реш*, а *Реш*1 — хвост этого списка.

3.3. Стратегия поиска в ширину

Обратимся к той же условной задаче, формализованной первым способом, что и при рассмотрении стратегии поиска в глубину. Вновь считаем, что построен ориентированный граф пространства состояний для этой задачи и каждому ребру этого графа приписано число 1 — «длина» этого ребра. Число 1 также интерпретируется как число операций перехода из одного состояния в другое.

Идея *алгоритма поиска в ширину* состоит в следующем: *начальная вершина графа принимается за начало решающего пути; а далее всегда*, *когда надлежит выбрать из нескольких альтернативных вершин ту*, *в которую следует перейти из текущей вершины для продолжения поиска решающего пути*, *нужно выбрать наименее «глубокую» из них*, *т. е. ту*, *которая расположена ближе других к начальной вершине*. Термины «наименее глубокая» и «ближе» здесь также следует понимать в смысле «длины» ребра. Заметим, в противоположность поиску в глубину стратегия поиска в ширину выбирает на каждом шаге из всех альтернативных вершин ближайшую к начальной вершине.

При поиске в ширину в случае, когда ближайших вершин оказывается несколько, можно выбрать любую из них, например, самую «левую». Здесь термин «левый» понимается так же, как и при описании стратегии поиска в глубину.

Если поиск приводит в тупик, т. е. к вершине, не являющейся целевой и не имеющей связей с более глубокими вершинами, то следует вернуться — выполнить откат — в предыдущую вершину и продолжить из нее поиск в ширину, выбирая следующую самую «левую» вершину.

Часто поиск в ширину называют полным перебором, потому что, имея тенденцию развиваться более в ширину, чем в глубину, данная стратегия поиска приводит к просмотру практически всех вершин пространства состояний, если в последнем целевые вершины являются самыми глубокими. На рис. 8 в качестве примера показано, в каком порядке алгоритм поиска в ширину проходит вершины пространства состояний.

Здесь вновь вершина *a* соответствует начальному состоянию, вершины *j* и *f* соответствуют конечным состояниям. Заметим, что первой конечной вершиной, встретившейся при построении решающего пути, является вершина *f*.

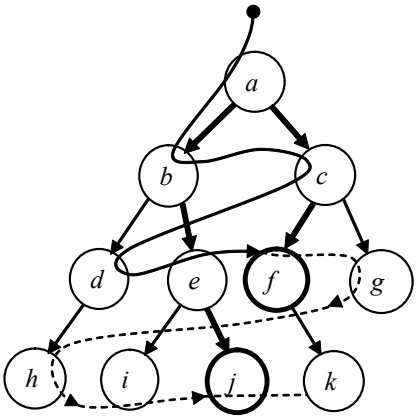


Рис. 8

Выделенный путь [*a*, *c*, *f*] — это первый найденный решающий путь, он же — кратчайший. Вторым решающим путем является путь [*a*, *b*, *e*, *j*], получающийся продолжением поиска в ширину. В этом примере второе решение получается практически полным перебором. Также полный перебор потребуется, если необходимо построить все решающие пути.

Поиск в ширину программируется сложнее, чем поиск в глубину, так как требует хранить информацию не только о всех альтернативных вершинах, которые пройдены в ходе поиска в ширину, но и о всех путях, приводящих к этим альтернативным вершинам. Это означает, что поиск в ширину требует хранения больших объемов информации, а следовательно, требует особой формы организации этой информации.

Во избежание зацикливания стратегия поиска в ширину должна тоже включать в себя механизм обнаружения циклов: ни одна из вершин уже построенных альтернативных путей не должна рассматриваться повторно в качестве альтернативы продолжения поиска в ширину.

Существенной особенностью стратегии поиска в ширину является тот факт, что она порождает решающие пути один за другим в порядке увеличения их длин, т. е. *самые короткие решения обнаруживаются первыми*. Это, очевидно, не верно для поиска в глубину.

Для обеих стратегий — поиск в глубину и поиск в ширину — в случае обширных пространств состояний существует опасность комбинаторного взрыва, приводящая к необходимости использования громадных объемов памяти компьютеров. Это ведет к усложнению программ, реализующих поиск в глубину и в ширину, в силу ограниченности памяти компьютеров. Комбинаторный взрыв часто можно избежать, использую эвристики.

3.4. Эвристический поиск

Стратегии поиска в глубину и в ширину целесообразно использовать для построения решающих путей в пространстве состояний тогда, когда поиск осуществляется без дополнительной информации о решаемой задаче. Иначе говоря, когда единственной информацией, имеющейся в распоряжении, являются списки вершин $N = \{n_1, n_2, ..., n_r\}$ и дуг $L = \{l_1, l_2, ..., l_s\}$ пространства состояний и больше ничего. Здесь $l_k = (n_{ik}, n_{jk})$ обозначает ребро, направленное из вершины n_{ik} в вершину n_{jk}.

Поиск в глубину и поиск в ширину является поиском, нацеленным на полный перебор альтернативных вершин путей, ведущих из начальной вершины. Очевидно, что полный перебор становится единственной возможностью найти решающий путь, когда поиск является неинформированным, т. е. когда о решаемой задаче известны только списки вершин N и дуг L.

Во многих случаях имеется возможность использовать некоторую дополнительную информацию, относящуюся к рассматриваемой задаче, чтобы содействовать сокращению поиска, т. е. сокращению времени и объема памяти, необходимых для выполнения поиска. Информацию такого рода (любая дополнительная к множествам N и L информация) называют *эвристической*. Стратегия поиска, использующая эвристическую информацию, называется *эвристическим поиском*. Процедура выбора альтернативных вершин, учитывающая эвристическую информацию о задаче, называется *эвристикой*. Часто эвристическую информацию находят, проводя углубленное исследование условий задачи, строя теорию решения соответствующих задач.

Дополнительная (эвристическая) информация о задаче в некоторых случаях может быть выражена численно в форме оценочной функции $f(n)$, определенной на вершинах пространства состояний, т. е. на множестве N. Оценочная функция $f(\cdot)$ вершине n пространства состояний сопоставляет действительное число $f(n)$, рассматриваемое как оценка перспективности (предпочтительности) продолжения поиска из вершины n. Чем меньше (а в некоторых задачах — чем больше) значение $f(n)$, тем предпочтительнее (перспективнее) продолжение поиска из вершины n. Вот почему эвристический поиск иногда называют *поиском с предпочтением*.

В некоторых случаях эвристическая информация о задаче выражается не количественно в форме числовой оценочной функции, а качественно в форме *отношения предпочтения*, которое определяется на множестве вершин пространства состояний N. Если две вершины n и n' являются кандидатами для продолжения поиска и находятся между собой в отношении предпочтения R, т. е. nRn',то более перспективной для достижения цели поиска следует считать вершину n.

Идея *эвристического поиска*, в какой бы форме — оценочной функции или отношения предпочтения — ни выражалась эвристическая информация, состоит в том, чтобы всегда продолжать поиск, начиная с наиболее перспективной вершины, выбранной из всего множества кандидатов. Точнее: *начальная вершина графа принимается за начало решающего пути; а далее всегда, когда надлежит выбрать из нескольких альтернативных вершин ту, в которую следует перейти из текущей вершины для продолжения поиска решающего пути, нужно выбрать наиболее перспективную из них, т. е. ту, для которой значение оценочной функции наименьшее* (*в некоторых задачах — наибольшее*), *или ту, которая предпочтительнее в смысле соответствующего отношения предпочтения.*

Алгоритм эвристического поиска во многом похож на алгоритм поиска в ширину. Существенное отличие состоит в том, что поиск в ширину из вершин-кандидатов всегда выбирает ту, которая наименее удалена от начальной вершины, тогда как эвристический поиск для каждого кандидата вычисляет оценку (сопоставляет с другими кандидатами в случае задания отношения предпочтения) и для продолжения выбирается кандидат с наилучшей оценкой (более предпочтительной). Оценочная функция обычно определяется так, чтобы наилучшей вершиной была та, для которой значение оценочной функции является наименьшим.

Существует целый ряд методов эвристического поиска, основанных на использовании оценочной функции и объединенных в одну группу, называемую *градиентными методами*. Иногда градиентные методы называют «взбирание на гору» («подъем в гору»). Суть градиентных методов поясним двумя примерами:

— альпинисту для скорейшего достижения вершин надо продвигаться к ней по самому крутому склону; если в какой-то момент это оказывается невозможным, то допускается минимально возможное отклонение, после чего альпинисту следует двигаться к цели из новой позиции, в которую привело его вынужденное отклонение;
— если мы на автомобиле хотим пересечь город в направлении с севера на юг за кратчайшее время, то стараемся сохранить это направление как можно дольше, а когда условия движения вынуждают нас изменить направление, то мы стараемся отклониться как можно меньше. Следовательно, глобальный экстремум в градиентных методах достигается через последовательность локальных экстремумов.

К градиентным методам поиска относятся симплекс-метод в линейном программировании, алгоритм *A**, методы последовательных приближений в численном анализе, минимаксные методы, альфа-бета алгоритм, динамическое программирование, метод ветвей и границ, метод разделения и оценки. В качестве иллюстрации эвристического поиска рассмотрим алгоритм *A** поиска решающего пути в пространстве состояний некоторой задачи.

3.5. Алгоритм *A**

Этот алгоритм является разновидностью эвристического поиска, когда эвристическая информация выражена в форме оценочной функции $f(\cdot)$, определенной на множестве вершин графа, представляющего пространство состояний. Пусть $N = \{n_1, n_2, ..., n_r\}$ — множество вершин, а $L = \{l_1, l_2, ..., l_s\}$ — множество дуг графа, представляющего собой пространство состояний решаемой задачи. Будем считать, что для любой дуги $l = (n_i, n_j) \in L$ определено число $c(n_i, n_j)$, которое назовем стоимостью этой дуги. Начальную вершину обозначим символом n_O, а целевую вершину — символом t или t_i, если имеется несколько целевых вершин.

Определим оценочную функцию $f(\cdot)$ таким образом, чтобы ее значение $f(n)$ для вершины n было оценкой суммы двух величин: минимальной стоимости пути от исходной вершины n_O к вершине n и минимальной стоимости пути от вершины n к некоторой целевой вершине t или t_i. Если первую из стоимостей обозначить $g^*(n)$, а вторую — $h^*(n)$, то наше предположение об определении оценочной функции $f(n)$ состоит в том, чтобы считать $f(n)$ некоторой оценкой для «идеальной» оценочной функции $f^*(n)=g^*(n) + h^*(n)$. Функция $f^*(n)$, следовательно, равна минимальной стоимости решающего пути при условии, что этот путь проходит через вершину n. Функции $g^*(n)$ и $h^*(n)$ можно определить через функцию $c(n_i, n_j)$ следующим образом:

$$g^*(n) = \min_{\{l_\alpha\}_{n_o}^{n}} \sum_{l_\alpha \in \{l_\alpha\}_{n_o}^{n}} c(n_{i_\alpha}, n_{j_\alpha}),$$

$$h^*(n) = \min_{t_i} \min_{\{l_\alpha\}_{n}^{t_i}} \sum_{l_\alpha \in \{l_\alpha\}_{n}^{t_i}} c(n_{i_\alpha}, n_{j_\alpha}).$$

Здесь в формуле для $g^*(n)$ минимум вычисляется по всем путям $\{l_\alpha\}_{n_o}^{n}$, начинающимся в начальной вершине n_O и заканчивающимся в вершине n, а в формуле $h^*(n)$ внешний минимум вычисляется по всем целевым вершинам t_i, а внутренний минимум — по всем путям $\{l_\alpha\}_{n}^{t_i}$, начинающимся в вершине n и заканчивающимся в целевой вершине t_i. Обозначение $\{l_\alpha\}_{m}^{n}$ используется для произвольного пути, начинающегося в вершине m и заканчивающегося в вершине n.

Использование этих двух формул для вычисления значений функций $g^*(n)$ и $h^*(n)$ в произвольной вершине n на практике, к сожалению, оказывается нецелесообразным из-за необходимости реализации полного перебора вершин и путей до выполнения самого эвристического поиска. Правда, в некоторых редких случаях они могут быть вычислены *a priori*, без выполнения полного перебора, исходя из условий решаемой задачи. В этих редких случаях в качестве оценочной функции $f(\cdot)$ следует использовать функ-

цию $f^*(\cdot)$. Тогда эвристический поиск обеспечивает раскрытие наименьшего числа вершин-кандидатов и наиболее быстро приводит к построению оптимального решающего пути.

В общем случае оценочную функцию определим соотношением $f(n) = g(n) + h(n)$, где $g(n)$ является некоторой оценкой (приближением) функции $g^*(n)$, а $h(n)$ — оценкой (приближением) функции $h^*(n)$. В качестве оценки $g(n)$ можно выбрать суммарную стоимость дуг пути $\{l_\alpha\}_{n_o}^n$, ведущего из вершины n_O в вершину n, т. е. можно положить

$$g(n) = \sum_{l_\alpha \in \{l_\alpha\}_{n_o}^n} c(n_{i_\alpha}, n_{j_\alpha}).$$

В этом способе задания $g(n)$ предполагается, что путь $\{l_\alpha\}_{n_o}^n$ — это тот самый путь, который реально был построен в результате эвристического поиска. Эта величина $g(n)$ возможно не является наименьшей, так как $g(n) \geq g^*(n)$ по определению функции $g^*(n)$, но она вполне может быть принята в качестве оценки функции $g^*(n)$.

Что касается функции $h(n)$, являющейся оценкой функции $h^*(n)$, то именно для вычисления $h(n)$ следует воспользоваться той дополнительной информацией о задаче, которую называют эвристической информацией. По этой причине функция $h(n)$ называется *эвристической функцией*. Как правило, о значении $h(n)$ бывает сложно сказать что-либо определенное до окончания поиска, так как к моменту достижения вершины n в ходе эвристического поиска о путях, ведущих из вершины n, практически бывает ничего не известно. Способы вычисления эвристической функции $h(n)$ могут быть различными в зависимости от условий конкретной решаемой задачи.

Если в условиях задачи достаточно информации для вычисления *расстояния Хемминга* — «расстояния» между данной вершины n и целевой вершиной t, — то целесообразно расстояние Хемминга принять в качестве значений эвристической функции $h(n)$. Приведем два примера, поясняющих понятие «расстояние Хемминга».

В первом примере по карте автомобильных дорог определяется кратчайший путь следования автомобиля из пункта A в пункт B. За расстояние Хемминга здесь можно принять обычное расстояние по карте от пункта A до пункта B по прямой, соединяющей эти два пункта.

Во втором примере рассмотрим игру «Восемь», в которой требуется перейти от некоторого начального расположения фишек S_O к конечному расположению фишек S_t за наименьшее число ходов. За расстояние Хемминга здесь можно принять число фишек в текущем их расположении S, стоящих не на своих местах в сравнении с конечным расположением фишек S_t. Если в обоих этих примерах в качестве $h(n)$ принять расстояние Хемминга, то эвристический поиск пройдет успешно и кратчайший решающий путь будет най-

ден, если вообще существует хоть какой-нибудь решающий путь. Заметим, что в этих примерах эвристическая функция $h(n)$ оценивает идеальную эвристическую функцию $h^*(n)$ снизу, т. е. выполняется соотношение $h(n) \le h^*(n)$.

В общем случае расстояние Хемминга — это метрика в некотором метрическом пространстве. Для случая алгоритма A^* расстояние Хемминга — это метрика, определенная на множестве N всех вершин графа, представляющего пространство состояний, т. е. функция $d(n, m)$, где $n \in N$ и $m \in N$, обладающая свойствами:

1) $d(n, m) \ge 0$ для всех $n \in N$ и $m \in N$,
   $d(n, m) = 0$ тогда и только тогда, когда $n = m$,
2) $d(n, m) = d(m, n)$ для всех $n \in N$ и $m \in N$ (симметричность),
3) $d(n, m) \le d(n, p) + d(p, m)$ для всех $n, m, p \in N$ (неравенство треугольника).

Эвристический поиск, использующий оценочную функцию $f(n) = g(n) + h(n)$, в которой $h(n) \le h^*(n)$ для всех нецелевых вершин, называется *алгоритмом A**. Этот же поиск при отсутствии ограничения $h(n) \le h^*(n)$ называется *алгоритмом A*. В алгоритмах A и A^* функция $g(n)$ является некоторой оценкой функции $g^*(n)$. В обоих алгоритмах из множества альтернативных вершин-кандидатов для продолжения поиска выбирается та, для которой значение $f(n)$ является наименьшим.

Теорема о состоятельности алгоритма *A.** *Для любого пространства состояний алгоритм A* завершает работу, получая оптимальный решающий путь из начальной вершины в целевую при условии, что решающий путь существует.*

Так как эвристический поиск является разновидностью поиска в ширину, то алгоритм A^* обнаруживает кратчайшие решающие пути первыми. Этот факт подтверждается сразу как частный случай более общего утверждения (теоремы о состоятельности алгоритма A^*), если в алгоритме A^* положить $h(n) \equiv 0$.

3.6. Пример применения алгоритма *A**

Рассмотрим игру «Восемь», для которой следует из начального состояния S_O перейти в конечное состояние S_t (рис. 9). Каждый ход в этой игре можно интерпретировать как движение пустой клетки — цифры 0 в соответствующих матрицах.

$$S_o = \begin{pmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 0 & 5 \end{pmatrix} \Rightarrow S_t = \begin{pmatrix} 1 & 2 & 3 \\ 8 & 0 & 4 \\ 7 & 6 & 5 \end{pmatrix}.$$

Рис. 9

Целью этой игры является поиск кратчайшей последовательности движений 0, переводящей состояние S_O в конечное состояние S_t. Текущее состояние, полученное после нескольких ходов, обозначим S.

Определим оценочную функцию $f(S) = g(S) + h(S)$, положив $g(S)$ равным фактическому числу ходов, приведших к состоянию S из состояния S_O, а эвристическую функцию $h(S)$ положим равной числу фишек, расположенных не на своих местах в состоянии S в сравнении с целевым состоянием S_t. Из условия задачи и определения функции $h(S)$ следует неравенство $h(S) \le h^*(S)$ для всех нецелевых вершин. Дерево поиска, полученное с помощью алгоритма ***A****, приведено на рис. 10. На нем около каждого состояния справа внизу в квадратных скобках через запятую представлены соответствующие значения $g(S)$ и $h(S)$. Оптимальный решающий путь выделен жирными линиями. При данном начальном состоянии S_O потребуется самое меньшее пять движений пустой клетки–цифры 0 для получения целевого состояния S_t.

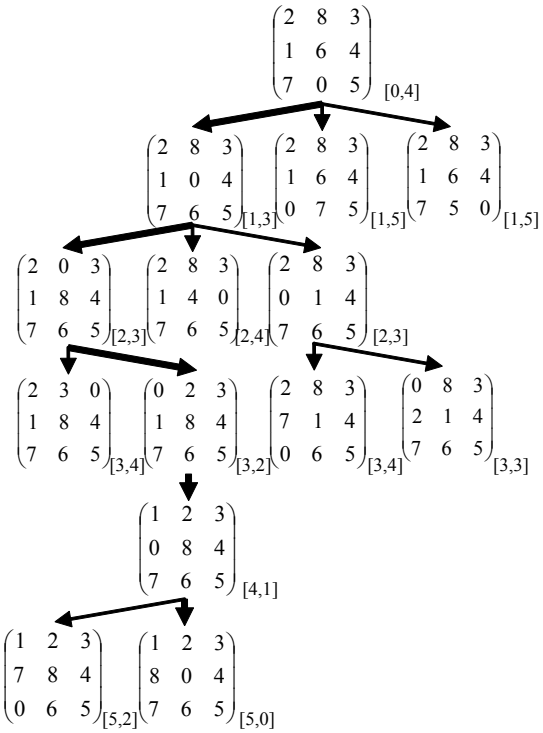


Рис. 10

Заметим, что алгоритм ***A**** с той же оценочной функцией для двух других начальных состояний (см. рис. 11) построит оптимальные решающие пути соответственно в 4 и 18 ходов.

$$S'_O = \begin{pmatrix} 1 & 3 & 4 \\ 8 & 0 & 2 \\ 7 & 6 & 5 \end{pmatrix}, \quad S''_O = \begin{pmatrix} 2 & 1 & 6 \\ 4 & 0 & 8 \\ 7 & 5 & 3 \end{pmatrix}.$$

Рис. 11

3.7. Сравнение вариантов алгоритма A^*

Рассмотрим два варианта алгоритма A^* для одного и того же пространства состояний некоторой задачи:

— алгоритм A_1^* с оценочной функцией
$f_1(n) = g_1(n) + h_1(n)$, $h_1(n) \le h^*(n)$,

— алгоритм A_2^* с оценочной функцией
$f_2(n) = g_2(n) + h_2(n)$, $h_2(n) \le h^*(n)$.

Выше отмечалось, что алгоритм A^* с идеальной оценочной функцией $f^*(n) = g^*(n) + h^*(n)$ обеспечивает на каждом шаге поиска раскрытие наименьшего числа вершин, кандидатов для продолжения поиска, и наиболее быстро строит оптимальный решающий путь. Заметим, что эвристической функцией в этом случае является идеальная эвристическая функция $h^*(n)$. Можно предположить, что чем точнее произвольная эвристическая функция $h(n)$ приближает снизу функцию $h^*(n)$, тем эффективнее (с меньшим числом вершин-кандидатов и быстрее) будет поиск оптимального решающего пути алгоритма A^*. Это предположение подтверждается на практике и в теории. В связи с этим введем определение

Определение. Алгоритм A_2^* называется *более информированным*, чем алгоритм A_1^*, если для всех нецелевых вершин выполняется неравенство $h_1(n) \le h_2(n)$.

Следующая теорема констатирует большую эффективность более информированного варианта алгоритма A^*.

Теорема. *Если A_1^* и A_2^* — два варианта алгоритма A^* для одного и того же пространства состояний некоторой задачи и алгоритм A_2^* более информирован, чем алгоритм A_1^*, то при наличии решающего пути к окончанию поиска каждая вершина, раскрытая алгоритмом A_2^*, будет раскрыта также и алгоритмом A_1^*.*

Именно в смысле этой теоремы алгоритм A_2^* является более эффективным алгоритмом, чем алгоритм A_1^*, так как вершин, раскрытых алгоритмом A_2^*, согласно теореме будет, вообще говоря, меньше, чем вершин, раскрытых алгоритмом A_1^*.

Примером, подтверждающим эту теорему, может служить игра «Восемь», если для алгоритма A_1^* принять $h_1(S) \equiv 0$, а для алгоритма A_2^* принять $h_2(S)$, равным числу фишек в состоянии S, стоящих не на своих местах в сравне-

нии с целевым состоянием S_t. Тогда алгоритм A_1* представляет собой обычный поиск в ширину. Это означает, что каждая вершина, раскрытая алгоритмом A_2*, оказывается обязательно раскрытой также алгоритмом A_1*, так как поиск в ширину раскрывает абсолютно все вершины пространства состояний, если отыскиваются все решающие пути.

3.8. Алгоритм программы GPS

General problem solver (GPS) — Общий решатель задач — одна из самых первых программ искусственного интеллекта. Она была предложена Алэном Ньюэлом, Клиффом Шоу и Гербертом Саймоном в конце 50-х годов XX в. и явилась одной из самых первых программ искусственного интеллекта. Программа GPS позволяет решать однотипным способом такие непохожие задачи, как вычисление неопределенного интеграла, логические головоломки, доказательство теорем средствами исчисления предикатов, грамматический анализ фразы.

В программе GPS реализована одна из разновидностей эвристического поиска решающего поддерева в пространстве подзадач. Это означает, что исходная задача формализуется вторым способом представления задач с помощью пространства подзадач и правил декомпозиции. При этом все задачи пространства подзадач могут быть только трех стандартных типов, описываемых ниже.

Программе GPS могут быть предложены для поиска решения только такие задачи, которые представляются в следующем виде: исходный объект, конечный объект и множество операторов, выполняющих преобразования объектов. Операторы дают возможность постепенными преобразованиями из исходного объекта получить конечный объект. Это и является целью решения задачи. Применение того или иного оператора к некоторому объекту диктуется потребностью уменьшить различия между преобразованным объектом и конечным объектом. *Различием* между двумя объектами считается любое свойство (набор свойств), присутствующее в одном из объектов и отсутствующее полностью или частично в другом.

Алгоритм программы GPS основан на идее сведения исходной задачи к подзадачам трех стандартных типов. Для удобства дальнейшего изложения введем следующие обозначения:

I — исходный объект, E — конечный объект,

O_1, O_2, ..., O_m — операторы преобразования объектов; запись $O_i(x)$ обозначает объект x', получаемый в результате преобразования объекта x с помощью оператора O_i, т. е. $x' = O_i(x)$.

$\Delta = (D_1, D_2, ..., D_n)$ — различие, по которому сравниваются объекты; D_j — j-й вид различия (j-е свойство); в дальнейшем запись $\Delta = \varnothing$ означает, что

между объектами нет различий, т. е. объекты совпадают, а запись означает, что между объектами имеются различия, т. е. объекты не совпадают.

Программа GPS строит *И-ИЛИ* дерево подзадач, которые могут быть трех стандартных типов:

$T(x, y)$ — преобразовать объект x в объект y,

$S(O, \Delta, x, y)$ — найти оператор O, уменьшающий различие Δ между объектами x и y,

$A(O, x, y)$ — применить оператор O к объекту x и сравнить $x' = O(x)$ с y.

Корнем *И-ИЛИ* дерева подзадач, т. е. начальной вершиной этого дерева является вершина, соответствующая исходной задаче $T(I, E)$. *И-ИЛИ* дерево подзадач строится программой GPS по следующим правилам декомпозиции:

1. Если текущей задачей является задача вида $T(x, y)$, то вычисляется различие Δ между объектами x и y. Если $\Delta = \varnothing$, то вершина, соответствующая задаче $T(x, y)$, является целевой (конечной) и x — конечный объект. Если $\Delta \neq \varnothing$, то программа GPS для задачи $T(x, y)$ создает подзадачу $S(Q, \Delta, x, y)$.

2. Если текущей задачей является задача вида $S(Q, \Delta, x, y)$, то для каждого из операторов $O_1, O_2, ..., O_m$ проверяется, можно ли его применить к объекту x. Если имеются операторы $O_{i_1}, O_{i_2}, ..., O_{i_k}$, которые можно применить к объекту x, то для каждого из них для задачи $S(Q, \Delta, x, y)$ создаются подзадачи $A(O_{i_1}, x), A(O_{i_2}, x), ..., A(O_{i_k}, x)$ соответственно. Тем самым вершина, соответствующая задаче $S(Q, \Delta, x, y)$, становится *ИЛИ*-вершиной.

3. Если текущей задачей является задача вида $A(O, x, y)$, то выполняется оператор преобразования O, т. е. находится объект $x' = O(x)$, и вычисляется различие Δ' между объектами x' и y. Затем Δ' и Δ (различие между объектами x и y) сравниваются. Если $\Delta' \leq \Delta$ (объект x' меньше отличается от объекта y, чем объект x от y), то для задачи $A(O, x, y)$ создается подзадача $T(x', y)$. Если $\Delta' > \Delta$, то вершина, соответствующая задаче $A(O, x, y)$, является тупиковой.

Программа GPS по существу является настраиваемой программной оболочкой, способной решать разнотипные задачи, такие, например, как синтаксический разбор предложений, символьное интегрирование, доказательство теорем и т. д. Настройка программы GPS на решение конкретной задачи осуществляется заданием Общему решателю задач (программе GPS) исходного объекта I, конечного объекта E, операторов преобразования объектов $O_1, O_2, ..., O_m$, а также, если возможно, таблицы зависимостей операторов O_i и конкретных видов различий (свойств D_j). Таблица зависимостей служит для ускорения поиска операторов, уменьшающих различие, и содержит для каждого оператора указания на те различия, которые могут быть уменьшены, если применяется соответствующий оператор. Таблица зависимостей может быть задана в следующей форме.

Таблица 1

Таблица зависимостей

	O_1	O_2	O_3	...
D_1	×	×		
D_2		×	×	
D_3				
D_4			×	
⋮				

Здесь для операторов O_1, O_2, ..., O_m знаком × помечены именно те конкретные виды различий (свойства D_j), которые уменьшаются, если применяется соответствующий оператор.

3.9. Пример использования алгоритма программы GPS

Рассмотрим задачу из области формальной логики [6, С. 296–302]: найти последовательность разрешенных преобразований, позволяющих перейти от формулы $(R\to\neg P)\wedge(\neg R\to Q)$ (это и есть исходный объект I) к формуле $\neg(\neg P\wedge Q)$ (конечный объект E). Разрешенными преобразованиями будем считать следующие преобразования, представляющие собой, по сути, правила переписывания одних строк символов в другие:

$O_0)\ \ \neg\neg A \models\!\!\!\Rightarrow A$

$O_1)\ \ A\wedge B \models\!\!\!\Rightarrow B\wedge A$

$A\to B \models\!\!\!\Rightarrow B\wedge A$

$A\wedge B \Rightarrow B$

$O_2)\ \ A\to B \models\!\!\!\Rightarrow \neg B\to\neg A$

$O_3)\ \ A\wedge A \Leftrightarrow A$

$A\vee A \Leftrightarrow A$

$O_4)\ \ A\wedge(B\wedge C) \Leftrightarrow (A\wedge B)\wedge C$

$A\vee(B\vee C) \Leftrightarrow (A\vee B)\vee C$

$O_5)\ \ A\vee B \Leftrightarrow \neg(\neg A\wedge\neg B)$

$A\vee B \Leftrightarrow \neg(\neg A\wedge\neg B)$

$O_6)\ \ A\to B \Leftrightarrow \neg A\vee B$

$O_7)\ \ A\wedge(B\vee C) \Leftrightarrow (A\wedge B)\vee(A\wedge C)$

$A\vee(B\wedge C) \Leftrightarrow (A\vee B)\wedge(A\vee C)$

$O_8)\ \ A\wedge B \Rightarrow A$

$O_9)\ \ A \Rightarrow A\vee X$

$O_{10})\ \ \left.\begin{matrix} A \\ B \end{matrix}\right\} \Rightarrow A\wedge B$

$O_{11})\ \ \left.\begin{matrix} A \\ A\to B \end{matrix}\right\} \Rightarrow B$

$O_{12})\ \ \left.\begin{matrix} A\to B \\ B\to C \end{matrix}\right\} \Rightarrow A\to C$

Преобразования $O_0 - O_6$ разрешается использовать как для всей формулы в целом, так и для отдельных ее частей, например:

$$\neg\neg(\neg F \vee (G \to \neg H)) \models\!\Rightarrow (\neg F \vee (G \to \neg H)),$$
$$H \to (\neg\neg(G \wedge F) \to H)) \models\!\Rightarrow H \to ((G \wedge F) \to H)).$$

В первом случае O_0 применяются ко всей преобразуемой формуле, во втором — только к ее части, к подформуле $\neg\neg(G \wedge F)$.

Преобразования O_8–O_{12} разрешается применять только ко всей формуле в целом, например, допустимым является преобразование O_8 в следующих примерах:

$(F \vee G) \wedge H)) \Rightarrow (F \vee G)$,
$(F \vee G) \wedge H)) \Rightarrow H$.

Недопустимым является преобразование O_8 в примерах:

$(F \vee G) \wedge H))$ L $(F) \vee G$,
$(F \vee G) \wedge H))$ L $(G) \vee G$.

Это отмечено перечеркиванием стрелок. В преобразованиях O_8–O_{12} стрелка $\Rightarrow$ отличается от стрелки $\models\!\Rightarrow$ в преобразованиях O_0–O_2. Двойная стрелка $\Leftrightarrow$ в O_0–O_7 означает, что допустимы оба преобразования $C1 \models\!\Rightarrow C2$ и $C2 \models\!\Rightarrow C1$, например, преобразование O_6, имеющее вид $A \to B \Leftrightarrow \neg A \wedge B$, означает, что допустимы оба преобразования:

$$A \to B \models\!\Rightarrow \neg A \vee B,$$
$$\neg A \vee B \models\!\Rightarrow A \to B.$$

Определим те свойства (различия), по которым будем сравнивать формулы. Их шесть:

— увеличивающееся число букв (D_1);
— уменьшающееся число букв (D_2);
— число символов отрицания (D_3);
— число прочих связок (D_4);
— измененная расстановка скобок (D_5);
— измененный порядок букв (D_6);

Анализируя действия, которые оказывают на эти различия преобразования O_0–O_{12}, можно построить следующую таблицу зависимостей.

Таблица 2

Таблица зависимостей в задаче преобразования формул

	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}
D_1				×				×		×	×		
D_2				×				×	×			×	×
D_3	×		×			×	×						
D_4						×	×	×					
D_5					×			×					
D_6		×	×										

Начальная вершина дерева подзадач соответствует исходной задаче $T(I, E)$, или $T((R\to\neg P)\wedge(\neg R\to Q), \neg(\neg P\wedge Q))$.

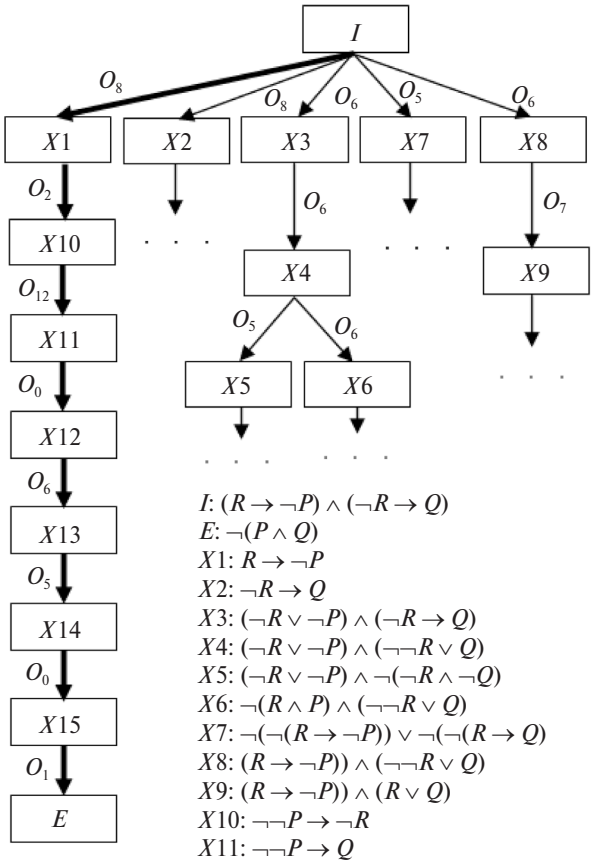


Рис. 12

Начальный фрагмент дерева подзадач для поставленной задачи представлен на рис. 12.

На нем в вершинах дерева обозначены формулы, а напротив ребер графа обозначены операторы преобразования, приводящие к соответствующим формулам. Выделенные ребра указывают на решающее поддерево. Многоточия соответствуют недостроенным частям дерева.

ГЛАВА 4
Формальные системы

4.1. Общее представление о формальной системе

Формальная система в математической логике — в самом общем понимании этого термина — это система обозначений, используемая для представления знаний и рассуждений какой-либо содержательной научной теории. Примерами формальных систем являются логика высказываний, логика предикатов, арифметика, теория вероятностей, теория групп и т. д. Любая формальная система основана на аксиоматическом методе и имеет свой формальный язык для представления знаний, для представления рассуждений в ней имеются аксиомы и правила вывода.

Под *аксиоматическим методом* понимается такой способ построения научной теории, при котором в основу теории кладутся некоторые исходные положения, называемые аксиомами, а все остальные утверждения теории получаются как логические следствия аксиом. Геометрия является исторически первой и вплоть до XIX в. была единственной аксиоматической теорией. Она дает наиболее наглядный образец формальной системы. Именно геометрия в начале XIX в. дала толчок развитию аксиоматического метода. Н. И. Лобачевский (1926 г.) и Я. Бояи (1929 г.) совершили открытие неевклидовой геометрии. Они установили, что, заменив привычный и, казалось бы, единственно «объективно-истинный» V постулат Евклида о параллельных его отрицанием, можно развивать чисто логическим путем геометрическую теорию, столь же стройную и богатую содержанием, как и геометрия Евклида. Этот факт привлек пристальное внимание математиков XIX в. к дедуктивному (от общего к частному) способу построения математических теорий, что привело к появлению новой проблематики, связанной с формальными системами.

Исследования аксиоматических (формальных) систем принесли свои плоды в виде логически безупречного построения элементарной геометрии (М. Паш, Дж. Пеано, Д. Гильберт) и в виде аксиоматизации арифметики (Дж. Пеано). Именно система аксиом, предложенная Д. Гильбертом, составляет основу современного школьного курса геометрии. Эта система со-

стоит из двадцати независимых аксиом, является полной системой аксиом и непротиворечива.

Аксиоматизация арифметики позволила свести решения вопроса о непротиворечивости многих математических теорий (евклидова геометрия, геометрия Лобачевского, геометрия Римана, проективная геометрия) к решению вопроса о непротиворечивости арифметики. В этом заключается особая роль арифметики среди остальных математических теорий. Заметим, что именно Дж. Пеано, аксиоматизируя арифметику, ввел в математическую логику обозначения, которые используются и поныне.

В работах Г. Фреге (1879 г.), Б. Рассела и А. Н. Уайхеда (1910) ставилась глобальная задача сведения всей математики к логике, т. е. к формальным системам. Д. Гильберт ставил перед собой задачу обоснования всей математики (решение вопроса о ее непротиворечивости) средствами аксиоматических систем. Позднее, благодаря результатам К. Гёделя (1930-е гг.), программа полной формализации (аксиоматизации) математики, предложенная Д. Гильбертом, потерпела крах. К такому итогу привели следующие доказанные Гёделем теоремы.

Первая теорема Гёделя о неполноте. *Любая непротиворечивая формальная система, содержащая минимум арифметики* (*операции сложения и умножения и кванторы* $\forall$ и $\exists$), *является неполной и непополнимой.*

Вторая теорема Гёделя о неполноте. *Если формализованная арифметика непротиворечива, то утверждение о ее непротиворечивости формализуемо средствами ее собственного языка, но это утверждение нельзя ни доказать, ни опровергнуть средствами самой формализованной арифметики.*

Если к двум теоремам Гёделя о неполноте добавить теорему Г. Генцена, доказанную им позднее (1960-е гг.), о том, что формализованная арифметика непротиворечива, то это позволит сделать два вывода. Во-первых (первая теорема Гёделя о неполноте), арифметику никогда нельзя аксиоматизировать «до конца», т. е. арифметика не полна при любой системе аксиом. Это означает, что при любой системе аксиом арифметики в ней будут содержаться утверждения, которые нельзя ни доказать, ни опровергнуть. Во-вторых (вторая теорема Гёделя о неполноте), непротиворечивость арифметики не может быть доказана средствами самой арифметики. Именно на пути построения мета-мета-...-теории арифметики Г. Генцен нашел доказательство непротиворечивости арифметики.

Завершая общую характеристику понятия «формальная система», отметим, что оно возникло в недрах формальной математической логики как средство изучения аксиоматического метода, впервые использованного в «Началах» Евклида при построении геометрии. Сама логика как наука о правильных рассуждениях, начиная с работ Аристотеля (IV в. до н. э.), зародилась и развивалась в рамках философии. И только в XIX в. с развитием методов формализации и символизации рассуждений появляется ма-

тематическая логика как отдельный раздел математики, изучающий формальные системы.

Точное определение формальной системы будет дано после рассмотрения основных идей аксиоматического метода на примере геометрии.

4.2. Аксиоматический метод в геометрии

Геометрия является классическим примером аксиоматической теории. Принципы построения именно этой теории легли в основу понятия «формальная система». Система аксиом геометрии, содержащаяся в «Началах» Евклида, была уточнена Д. Гильбертом в 1899 г. Предложенная им система аксиом евклидовой геометрии содержит 20 аксиом, которые разбиты на пять групп.

Первичными (неопределяемыми) понятиями в гильбертовой системе аксиом евклидовой геометрии являются объекты: точка, прямая и плоскость и отношения между ними, выражаемые словами: «принадлежит», «между», «конгруэнтен». По Гильберту, природа основных объектов и отношений между ними может быть какой угодно, лишь бы эти объекты и отношения удовлетворяли указанным аксиомам. Так же и в формальной системе: первичные обозначения не определяются, а задаются как конечное множество символов, называемое *алфавитом языка формальной системы*.

В геометрии наряду с первичными (неопределяемыми) понятиями имеются определяемые понятия (отрезок, луч, угол и др.). Их определения даются с помощью первичных понятий или с помощью ранее определенных понятий. Точно также и в формальной системе: из символов алфавита по определенным правилам образуются допустимые последовательности символов, называемые *формулами*, а иногда словами языка формальной системы. Правила образования формул (слов) называют *синтаксическими правилами*. Они образуют грамматику языка формальной системы.

Перечислим пять групп аксиом евклидовой геометрии.

I группа (8 аксиом) — аксиомы принадлежности (соединения), описывающие отношение «принадлежит», например:

I_1. Для любых двух точек существует прямая, проходящая через каждую из этих двух точек.

I_7. Если две плоскости имеют общую точку, то они имеют еще, по крайней мере, одну общую точку.

II группа (4 аксиомы) — аксиомы порядка, описывающие отношение «между», например:

II_2. Для любых двух точек A и B на прямой AB существует, по крайней мере, одна точка C, такая, что точка B лежит между точками A и C.

III группа (5 аксиом) — аксиомы конгруэнтности, описывающие отношение «конгруэнтен» (эквивалентен), например:

III_2. Если $A'B' \equiv AB$ и $A''B'' \equiv AB$, то $A'B' \equiv A''B''$.

IV группа (2 аксиомы) — аксиомы непрерывности Архимеда и Кантора, например:

IV_2 (аксиома Кантора). Для любой последовательности вложенных отрезков, длины которых стремятся к нулю, существует точка, принадлежащая всем отрезкам.

V группа (1 аксиома) — аксиома Евклида о параллельных прямых: через точку, не принадлежащую заданной прямой на плоскости, можно провести только одну прямую в этой плоскости, параллельную заданной прямой.

В геометрии аксиомы принимаются в качестве исходных утверждений, истинность которых не доказывается и не опровергается. Они играют роль посылок в рассуждениях, проводимых с использованием логических законов: закон силлогизма, метод от противного, метод математической индукции и т. д. Исходя из аксиом, доказываются другие утверждения (теоремы) евклидовой геометрии. Доказанные теоремы вместе с аксиомами используются далее для обоснования новых теорем. Таким образом, строится теория, которую называют евклидовой геометрией.

В формальных системах точно также. Как уже отмечалось, допустимые последовательности символов называют формулами. Некоторую группу формул называют *аксиомами*, которые используются для образования других формул, называемых, как в геометрии, *теоремами*. Правила образования теорем из аксиом и из ранее полученных теорем играют в формальной системе ту же роль, какую логические законы играют в геометрии. В формальной системе правила образования формул называются *правилами вывода*.

Очень важными свойствами всякой теории, в том числе и геометрии, являются *полнота* и *непротиворечивость теории*. Геометрия Евклида является полной и непротиворечивой теорией. Полнота евклидовой геометрии означает, что любое утверждение о геометрических объектах, либо противоположное ему утверждение, являются доказуемыми в ней. Непротиворечивость евклидовой геометрии означает, что в ней никогда нельзя доказать два противоречащих друг другу утверждения.

Завершая описание аксиоматического метода в геометрии, заметим, что в ходе построения евклидовой геометрии были реализованы следующие четыре принципа:

1. Введение первичных (неопределяемых) понятий и отношений.
2. Определение новых понятий и отношений через первичные или ранее определенные понятия и отношения.
3. Введение аксиом (исходных утверждений).
4. Использование логических законов для доказательства теорем исходя из аксиом или ранее доказанных теорем.

Эти четыре принципа полностью характеризуют сущность аксиоматического метода, положенного в основу понятия «формальная система».

4.3. Определение и свойства формальной системы

Формальная система представляет собой совокупность возможно абстрактных объектов (символов, обозначений), вообще говоря, никак не связанных с внешним миром. В этой совокупности имеются правила оперирования объектами (символами, обозначениями) в чисто синтаксической трактовке, изначально без какого бы то ни было смыслового содержания (семантики) этого оперирования.

Определение. *Формальной системой называется четверка объектов* $M = <T, P, A, B>$, *где:*

T *— конечный алфавит* (*конечное множество символов*)*;*

P *— множество синтаксических правил построения формул из символов алфавита;*

A *— конечное множество аксиом*, *представляющее собой некоторое подмножество множества формул;*

B *— конечное множество правил вывода*, *которые позволяют получать из некоторого конечного множества формул* $U_1, U_2, ..., U_p$ *другое множество формул* $W_1, W_2, ..., W_n$ *формальной системы и записываются в виде*

$$U_1, U_2, ..., U_p \rightarrow W_1, W_2, ..., W_n.$$

Здесь стрелка $\rightarrow$ читается как «влечет» или «следует», формулы $U_1, U_2, ..., U_p$ называются *посылками*, а каждая из формул $W_1, W_2, ..., W_n$ называется *заключением* правила вывода.

Формальную систему иногда называют *аксиоматикой*, или *формальной теорией*.

Алфавит, который заранее предполагается конечным, иногда называют *словарем*. При этом в нем, если необходимо, различают константы, переменные, пропозициональные, функциональные и предикатные символы, логические операторы, кванторы и т. д. Алфавит в формальной системе является аналогом множества первичных (неопределяемых) понятий и отношений в геометрии.

Синтаксические правила определяют допустимые в формальной системе последовательности символов алфавита. Образованные в соответствии с синтаксическими правилами последовательности символов называются *формулами* формальной системы. Тем самым множество синтаксических правил

определяет, по существу, грамматику формул. Проводя аналогию с естественным, например русским, языком, будем говорить, что алфавит и грамматика (множество синтаксических правил) формальной системы определяют ее *формальный язык*. Множество формул в формальной системе является аналогом множества определяемых понятий в геометрии.

Аксиомы формальной системы играют в ней ту же роль, какую в геометрии играют ее аксиомы, т. е. используются в качестве посылок в правилах вывода.

Правила вывода в формальной системе являются аналогом логических законов, используемых в геометрических рассуждениях. Результатом рассуждений в геометрии являются теоремы. В формальной системе правила вывода используются для проведения доказательств. *Формальное доказательство*, или просто *доказательство*, определяется как конечная последовательность формул $F_1, F_2, ..., F_r$ формальной системы, такая, что каждая формула F_i либо является аксиомой, либо при помощи одного из правил вывода выводима из предшествующих ей формул $F_1, F_2, ..., F_j$, где $j < i$. Некоторая формула называется *теоремой* и обозначается $\vdash F$, если существует доказательство $F_1, F_2, ..., F_r$, в котором она является последней, т. е. $F_r = F$. Отсюда ясно, что всякая аксиома является теоремой. Доказательства в формальной системе являются аналогами содержательных рассуждений в геометрии и служат для получения теорем. Отметим, что в следующей записи

$$\{\text{аксиомы}\} \subset \{\text{теоремы}\} \subset \{\text{формулы}\} \subset \{\text{последовательности символов алфавита}\}$$

все включения множеств являются строгими.

Различают два типа правил вывода:

а) *Правила продукций* (продукции, продукционные правила) применяются к формулам, рассматриваемым как единое целое. К частям формул правила продукции применять нельзя. Например, в формальной системе, описывающей родственные отношения между людьми, правила вывода

$$\begin{aligned} S(x,y) &\rightarrow F(y,x) \\ S(x,y)\ и\ S(y,z) &\rightarrow GS(x,z) \\ S(x,y)\ и\ D(z,y) &\rightarrow B(x,z) \end{aligned}$$

являются продукциями, где

$S(x, y)$ означает, что «x является сыном для y»,

$F(x, y)$ — «x является отцом для y»,

$GS(x, y)$ — «x является внуком для y»,

$D(x, y)$ — «x является дочерью для y»,

$B(x, y)$ — «x является братом для y».

б) *Правила переписывания* применяются как ко всей формуле в целом подобно правилу продукций, так и к отдельным частям формул, причем сами эти части являются формулами формальной системы. Например, в формальной системе, описывающей тождественные тригонометрические преобразования, правила вывода

(*а*) $\frac{a}{a} \models 1$,
(*б*) $\operatorname{tg} x \models \frac{\sin x}{\cos x}$

являются правилами переписывания. Здесь обычная стрелка $\rightarrow$ заменяется на специальную стрелку $\models$. Такое обозначение используется именно в правилах переписывания, для того чтобы отличать их от правил продукций, в которых используется обычная стрелка. Второе из указанных правил переписывания означает: где бы ни встретилась формула $\operatorname{tg} x$ внутри другой тригонометрической формулы или как отдельная формула, ее можно заменить дробью $\frac{\sin x}{\cos x}$, оставляя остальные выражения формулы без изменения. Применение обоих правил переписывания, приведенных выше в качестве примеров, позволяет упростить тригонометрическую формулу $\cos x \cdot \operatorname{tg} x$ в два этапа:

1) $\cos x \cdot \operatorname{tg} x \models \cos x \cdot \frac{\sin x}{\cos x} = \frac{\cos x}{\cos x} \cdot \sin x$ (правило *б*),
2) $\frac{\cos x}{\cos x} \cdot \sin x \models \sin x$ (правило *а*).

На практике формальные системы возникают из потребности представления знаний и рассуждений конкретных содержательных научных теорий. В этом случае формальные системы служат средством для более углубленного изучения математическими методами соответствующей научной теории. Однако вполне возможно определение формальной системы чисто абстрактно, безотносительно к некоторой содержательной теории. И только потом, когда формальная система определена и достаточно изучена, для нее может быть найдена подходящая содержательная интерпретация. Конкретная формальная система является тем более интересной, чем больше существует для нее различных интерпретаций в виде содержательных научных теорий. В таком случае наличие даже какого-то одного формального доказательства уже обеспечивает получение различных конкретных результатов во множестве содержательных научных теорий. В современной математике большой интерес вызывают формальные системы самого общего характера, которые исследуются в теории категорий и теории моделей.

Рассмотрим пример абстрактной формальной системы, для которой позднее будут даны две различные интерпретации.

Пример. Формальная система (*JP*).

1. Алфавит $\{a, b, \square\}$.

2. Формулы — это символ или какая-либо последовательность символов алфавита.

3. Одна аксиома: $a \square a$

4. Одно правило вывода: $c_1 \square c_2 \rightarrow bc_1 \square bc_2$

Здесь символы c_1 и c_2 не являются символами алфавита, а играют служебную роль. Символы c_1 и c_2 означают какие-то последовательности символов *a* или *b*.

Из определения данной формальной системы непосредственно вытекает, что ее теоремами являются только следующие формулы:

$$a \square a$$
$$ba \square ba$$
$$bba \square bba$$
$$bbba \square bbba$$
$$\ldots\ldots\ldots$$

Очевидно, что, например, формула $baab \square abba$ не является теоремой.

Когда формальная система $M = <T, P, A, B>$ определена, т. е. заданы алфавит *T*, грамматика *P*, аксиомы *A* и правила вывода *B*, представляется естественным ответить на следующие вопросы относительно системы *M*:

1. Есть ли аксиомы в *A*, которые могут быть выведены из остальных аксиом?

2. Есть ли противоречия в системе *M*, т. е. имеется ли формула *F*, что формула *F* и ее отрицание $\neg F$ обе доказуемы в *M*?

3. Верно ли, что для любой формулы *F* системы *M* доказуемой является либо сама формула *F*, либо ее отрицание $\neg F$?

4. Существует ли такая процедура, которая для каждой формулы *F* системы *M* позволяет установить за конечное число шагов, является ли она теоремой или нет?

5. Существует ли такая интерпретация формальной системы *M*, что все ее теоремы становятся истинными утверждениями в этой интерпретации?

При утвердительном ответе на первый вопрос говорят, что аксиомы формальной системы являются *зависимыми*, при отрицательном — *независимыми*. Во множестве зависимых аксиом доказуемую аксиому можно без ущерба удалить из списка аксиом.

При ответе «да» на второй вопрос формальная система называется *противоречивой*, в противном случае — *непротиворечивой*. Интерес представляют только непротиворечивые формальные системы, так как в противоречивой формальной системе доказуемыми оказываются любая формула и ее отрицание.

В зависимости от того, каким будет ответ на третий вопрос, говорят о полной или неполной формальной системе (теории). Если в системе любая формула либо сама система является доказуемой, либо доказуемо ее отрицание, то такая формальная система называется *полной*, иначе она называется *неполной*. В полной теории все формулы можно разбить на два непустых и непересекающихся класса: теоремы и не-теоремы (для них не существует доказательства). В неполной теории либо может встретиться формула, что она сама и ее отрицание не доказуемы, либо может встретиться формула, что она сама и ее отрицание доказуемы. Последнее означает противоречивость соответствующей формальной теории. Ценность полной теории заключается в том, что такая ситуация «неопределенности» — недоказуемости формулы и ее отрицания — ни для одной формулы не возникает.

Со свойством полноты теории тесно связано свойство разрешимости теории. Если для формальной теории существует процедура, позволяющая для любой формулы за конечное число шагов решить, является ли она теоремой или не-теоремой, то теория называется *разрешимой*, а сама такая процедура называется *процедурой решения*. При ответе на четвертый вопрос основная трудность заключается в отыскании процедуры решения. Такая процедура существует далеко не всегда, даже для таких простых и фундаментальных теорий, какой является исчисление предикатов первого порядка. Примером разрешимой формальной теории может служить система (*JP*). Для нее процедура решения может быть такой:

1) просматриваются слева направо символы анализируемой формулы и ведется подсчет символов *b* до символа *a*;
2) затем сразу за символом *a* должен следовать символ □;
3) после него должны следовать символы *b* в том же количестве, что и в начале формулы;
4) за ними должен следовать один символ *a*, после которого формула заканчивается.

Если эти четыре условия выполнены, то анализируемая формула теории является теоремой. В противном случае она является не-теоремой.

Наконец, в пятом вопросе выясняется, можно ли формальной теории придать такой содержательный смысл, чтобы ее теоремы соответствовали истинным утверждениям. Иначе говоря, имеется ли модель формальной теории. По существу содержательная модель представляет собой обычную выраженную естественным русским языком научную теорию в привычном понимании этого слова: со своими объектами исследования, методами и специфическими понятиями и идеями. Не следует, правда, считать, что моделью обязательно должна быть какая-нибудь известная содержательная научная теория. Роль модели одной формальной теории вполне может выполнять другая формальная теория или ее часть. Если утвердительно ответить на пятый вопрос, то это будет означать, что соответствующей формальной тео-

рии M сопоставляется некая другая (возможно также формальная) теория M', в которой имеет смысл говорить об истинности и ложности ее утверждений. В этом случае теорию M', какая бы она ни была, формальная или содержательная, называют *моделью* теории M. Так, обычная арифметика, известная из школьного курса, служит моделью для формальной арифметики. Другой интересный пример модели будет рассмотрен ниже. Он относится к содержательной интерпретации неевклидовой геометрии средствами евклидовой геометрии.

4.4. Определение понятия модели

Переход от формальной теории к ее модели является завершающей стадией решения некоторой проблемы математическими средствами. Сущность *математического подхода* к решению реальных задач, к какой бы отрасли знаний они не относились, заключается в следующем:

- сначала объект исследования или конкретная реальная проблема формализуется, т. е. подбирается формальная система, в которой утверждения (гипотезы) о том или ином решении поставленной задачи представляются в виде формул,
- затем формулы, соответствующие проверяемым гипотезам, анализируются на предмет их доказуемости средствами формальной системы; другими словами, обосновывается решение проблемы средствами формальной системы; если же решение, подлежащее обоснованию, заранее не известно, то оно отыскивается и обосновывается также средствами формальной системы,
- наконец, с помощью интерпретации символов формальной системы осуществляется обратный переход от формализованной постановки задачи к ее исходной постановке, имеющей отношение к реальному миру, т. е., строго говоря, строится ее реальная модель. Благодаря интерпретации доказуемых формул (теорем) формальной системы выясняется содержательный смысл истинного решения поставленной проблемы. Тем самым завершается процесс решения задачи математическими средствами.

При выяснении содержательного смысла формальной теории определяющими являются следующие два фактора:

- каким символам формальной системы что соответствует в реальной предметной области, к которой относится решаемая проблема,
- по каким правилам формулы формальной системы переводятся в утверждения о свойствах объектов предметной области.

В понятии «интерпретация» учитываются оба эти фактора. Пусть $M = <T, P, A, B>$ — некоторая формальная теория, а D — непустое множество объектов (реальных или абстрактных), имеющих отношение к решаемой задаче. Множество D называется *предметной областью*.

Определение. *Интерпретацией формальной теории M называется такое отображение* φ: $T \to D$, *при котором любой аксиоме теории M соответствует истинное утверждение о свойствах или взаимосвязях объектов из множества D, а каждой формуле F этой теории соответствует определенное истинное или ложное утверждение о свойствах или взаимосвязях объектов из множества D.*

Введем обозначения: 1 — значение истинности «истина», 0 — значение истинности «ложь». Запись $\varphi(F) = 1$ означает, что формула F интерпретируется с помощью интерпретации φ в истинное утверждение о свойствах или взаимосвязях объектов из множества D. Запись $\varphi(F) = 0$ означает, что формула F интерпретируется с помощью интерпретации φ в ложное утверждение о свойствах или взаимосвязях объектов из множества D. Если $\mathcal{F}$ — какое-нибудь множество формул, то запись $\varphi(\mathcal{F}) = 1$ обозначает тот факт, что для любой формулы $F \in \mathcal{F}$ выполняется $\varphi(F) = 1$. Аналогично определяется равенство $\varphi(\mathcal{F}) = 0$. Интерпретацию φ теории M называют также *моделью* этой теории.

Рассмотрим пример модели неевклидовой геометрии на плоскости, которая создана немецким математиком Ф. Клейном и для которой предметной областью является евклидова геометрия на плоскости. Евклидова геометрия рассматривается как формальная система с гильбертовой системой аксиом. Неевклидова геометрия также рассматривается как формальная система, единственное (!) отличие которой от евклидовой геометрии заключается в замене евклидовой аксиомы о параллельных на аксиому о параллельных Лобачевского или на аксиому о параллельных Римана. В первом случае неевклидову геометрию называют геометрией Лобачевского, во втором — геометрией Римана. В геометрии Римана аксиома о параллельных формулируется следующим образом.

Аксиома о параллельных Римана. *Если в плоскости заданы прямая и точка вне нее, то в этой плоскости не существует ни одной прямой, проходящей через заданную точку параллельно заданной прямой.*

Из нее можно сделать вывод о том, что в геометрии Римана параллельных прямых нет вообще.

Ниже строится модель геометрии Лобачевского на плоскости. В ней аксиома о параллельных формулируется так.

Аксиома о параллельных Лобачевского. *Если в плоскости заданы прямая и точка вне нее, то в этой плоскости существует более одной прямой, проходящей через заданную точку параллельно заданной прямой.*

Из нее можно сделать вывод о том, что в геометрии Лобачевского на плоскости для произвольной прямой можно провести через заданную точку,

не принадлежащую этой прямой, бесконечно много прямых, параллельных данной прямой.

Сущность модели геометрии Лобачевского состоит в следующем. Пусть C — некоторый круг, расположенный в евклидовой плоскости. В геометрии Лобачевского плоскостью назовем внутренность этого круга (рис. 13), т. е. точки, лежащие на окружности, этой плоскости не принадлежат. Тот факт, что точки окружности не принадлежат плоскости Лобачевского (кругу), отмечен на рис. 13 пунктиром. Прямыми назовем хорды этого круга, причем концы хорд, естественно, не рассматриваются, т. е. не принадлежат этим прямым в плоскости Лобачевского. Этот факт отмечен выделением концов хорд. Наконец, точками назовем внутренние точки круга. Для таких точек, прямых и плоскости выполняются все аксиомы геометрии Лобачевского на плоскости, в том числе аксиома о параллельных Лобачевского.

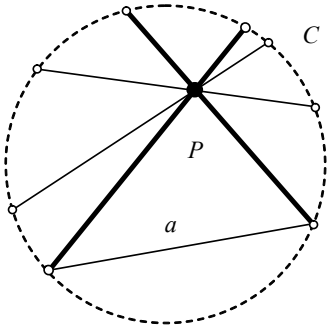


Рис. 13

Действительно, через точку P, не принадлежащую прямой a, проходит бесконечно много прямых, не имеющих общих точек с заданной прямой a. Выделенные прямые — это крайние положения параллельных прямых.

4.5. Свойства формальных теорий. Понятие *метатеории*

Формальные теории служат средством представления знаний и рассуждений содержательных научных теорий и используются для решения математическими методами проблем, возникающих в этих теориях. Смысл математического подхода к решению конкретных проблем обычных научных теорий, как отмечалось выше, заключается в определении подходящей формальной теории, в которой исходная проблема, пройдя этап формализации, решается внутренними ограниченными средствами этой теории. Формальные теории позволяют сделать исходную проблему более «прозрачной», т. е. «увидеть» внутреннюю структуру и взаимосвязи элементов изучаемого объ-

екта. Ограниченность средств формальной теории, вообще говоря, облегчает поиск решения формализованной проблемы. Интерпретация результатов решения позволяет вернуться к исходной предметной области, т. е. узнать содержательный смысл найденного решения исходной проблемы. Таким образом, формальная теория является средством изучения объектов некоторой предметной области.

Но объектом изучения может быть сама формальная теория. Именно так происходит, когда для некоторой формальной теории M устанавливается, обладает ли она свойствами полноты, непротиворечивости, разрешимости, зависимы ли ее аксиомы. Для решения этих проблем может быть построена другая формальная теория M', которая называется *метатеорией* теории M. Теоремы метатеории M' называются *метатеоремами*. Теоремы Гёделя о неполноте, приведенные выше, являются по существу метатеоремами. Алфавит метатеории M' и ее грамматика образуют *метаязык*. Запись теорем Гёделя о неполноте и их доказательства осуществляется на обычном русском языке, который в данном случае играет роль метаязыка. Почти все сказанное выше о формальных системах, по сути, является метатеорией для любой формальной теории.

В качестве примеров метатеорем приведем известные в математической логике теоремы ограничения Гёделя, Тарского и Чёрча, которые произвели огромное впечатление одновременно и на математиков, и на философов.

Теорема Гёделя. *Существуют формальные системы, в которых имеются формулы F такие, что ни F, ни ее отрицание $\neg F$ не являются доказуемыми.*

Эта теорема констатирует, что существуют неполные формальные теории. Например, формальная арифметика, так как свойство ее непротиворечивости выразимо формулой формальной арифметики, которую нельзя ни доказать, ни опровергнуть, не выходя за пределы этой теории.

Теорема Тарского. *Существуют формальные системы, в которых во всякой их интерпретации найдутся выражения, истинные, но недоказуемые.*

Более кратко иногда эта теорема формулируется так: понятие истинности не формализуемо.

Теорема Чёрча. *Существуют формальные системы, для которых отсутствует алгоритм, позволяющий установить, является ли формула теоремой или не-теоремой.*

Другими словами, эта теорема утверждает, что существуют неразрешимые формальные теории, например, исчисление предикатов первого порядка.

Понятие *алгоритма*, использованное в теореме Чёрча, принадлежит к числу основных первоначальных понятий математики, не допускающих определения в терминах более простых понятий. Говоря точнее, понятие *алгоритма*, как и другое основное понятие математики «множество», является неопределяемым.

4.6. Понятие *алгоритма* и *разрешимости теории*

Использование понятия *алгоритма* в математике основано на интуитивном представлении об алгоритме и тезисе Чёрча.

Интуитивное определение алгоритма. *Алгоритм* (*алгорифм*) — *это точное предписание*, *которое задает вычислительный или иной процесс*, *начинающийся с произвольного исходного данного* (*из некоторой совокупности возможных для данного алгоритма исходных данных*) *и направленный на получение полностью определяемого этим исходным данным результата*.

Алгоритмами являются, например, известные из начальной школы правила сложения, вычитания, умножения и деления столбиком.

Возможные уточнения понятия *алгоритма* практически оказываются сужением интуитивного понятия *алгоритма*. К таким уточненным понятиям относится понятие *алгоритма*, основанное на понятии *машины Тьюринга*, или нормальные алгоритмы Маркова. Эти два понятия имеют точные определения и, как выяснилось, являются эквивалентными между собой. Тезис Чёрча постулирует эквивалентность интуитивного и точного определений алгоритма.

Тезис Чёрча. *Интуитивное понятие алгоритма эквивалентно точному понятию алгоритма*.

Тезис Черча не может быть строго доказан, так как в его формулировке участвует неточное понятие *интуитивное понятие алгоритма*. Этот тезис является естественно-научным фактом, подтверждаемым опытом, накопленным в математике за всю ее историю. Все известные в математике примеры алгоритмов удовлетворяют ему. На понятии *алгоритма* основано понятие *разрешимости формальной теории*.

Определение. *Формальная теория называется разрешимой*, *если существует алгоритм*, *который для любой формулы данной теории определяет*, *является ли она теоремой или не-теоремой*.

Примером разрешимой теории является формальная система (*JP*). Для нее разрешающий алгоритм был приведен выше. Примером неразрешимой теории является логика предикатов первого порядка. Для логики высказываний справедлива теорема Э. Поста.

Теорема Э. Поста. *Формула логики высказываний доказуема тогда и только тогда*, *когда она является тождественно-истинной формулой*.

Это означает, что логика высказываний, которая входит составной частью в логику предикатов, является разрешимой теорией.

4.7. Доказуемость и истинность

Пусть M — некоторая формальная теория, а φ — произвольная модель этой теории. Тогда если F — формула теории M, то априори эта формула либо доказуема (является теоремой), либо не доказуема (не-теорема). Интерпретация $\varphi(F)$ этой формулы является либо истиной, т. е. $\varphi(F) = 1$, либо ложью, т. е. $\varphi(F) = 0$. Так что для произвольной формулы F теории M и произвольной модели φ имеются четыре варианта взаимоотношений доказуемости и истинности:

(T_0): F — теорема и $\varphi(F)$ — ложное утверждение,
(T_1): F — теорема и $\varphi(F)$ — истинное утверждение,
($\overline{T}_0$): F — не-теорема и $\varphi(F)$ — ложное утверждение,
($\overline{T}_1$): F — не-теорема и $\varphi(F)$ — истинное утверждение.

Для произвольной формальной теории M и любой ее модели φ символом T_0 будем обозначать класс формул теории M, удовлетворяющих условию (T_0), символом T_1 будем обозначать класс формул теории M, удовлетворяющих условию (T_1), символом $\overline{T}_0$ — класс формул, удовлетворяющих условию ($\overline{T}_0$), и $\overline{T}_1$ — класс формул, удовлетворяющих условию ($\overline{T}_1$). Тогда множество $\mathcal{F}_M$ всех формул теории M, вообще говоря, представляет собой объединение этих четырех классов формул

$$\mathcal{F}_M = T_0 \cup T_1 \cup \overline{T}_0 \cup \overline{T}_1.$$

Наиболее предпочтительным является выполнение условий

$$\mathcal{F}_M = T_1 \cup \overline{T}_0, \quad T_0 = \varnothing, \quad \overline{T}_1 = \varnothing,$$

которые означают, что все теоремы интерпретируются моделью φ как истинные утверждения, а не-теоремы — как ложные. В этом случае можно сказать, что формальная теория M адекватно формализует предметную область D модели φ, для которой выполняются эти условия. Если $T_0 \neq \varnothing$, то в теории M имеются теоремы F, для которых $\varphi(F)$ — ложное утверждение. Такая ситуация, $\varphi(F) = 0$, указывает на необходимость корректировки теории M для того, чтобы в ней все теоремы интерпретировались истинными утверждениями. Если $\overline{T}_1 \neq \varnothing$, то в теории M имеются не-теоремы F, которые все же являются истинными утверждениями. Ситуация $\overline{T}_1 \neq \varnothing$ может означать необходимость дополнения теории M новыми аксиомами. Еще более неприятным является тот факт (теорема Тарского), что имеются формальные системы, в которых класс $\overline{T}_1$ не является пустым ни при какой интерпретации.

Приведем пример ситуации $\overline{T}_1 \neq \varnothing$ для формальной системы (*JP*), т. е. подберем для этой формальной системы интерпретацию (модель) φ, для которой имеются не-теоремы, интерпретируемые как истинные утверждения. Рассмотрим модель φ, в которой символам алфавита {*a*, *b*, □} формальной системы (*JP*) приписаны значения:

a – утверждение «Сократ смертен»,

b – фраза «Неверно, что»; обозначает отрицание утверждения,

□ – фраза «эквивалентно тому что»; обозначает эквивалентность утверждений.

Тогда аксиома *a*□*a* интерпретируется так:

«Сократ смертен эквивалентно тому, *что Сократ смертен»*,

а теорема *ba*□*ba* означает

«Сократ бессмертен эквивалентно тому, что Сократ бессмертен»,

если считать, что фразы «Неверно, что Сократ смертен» и «Сократ бессмертен» имеют идентичный смысл.

Но для формулы *bba*□*a*, не являющейся теоремой формальной системы (*JP*), ее интерпретация является истинным утверждением:

«Неверно, что Сократ бессмертен эквивалентно тому,
что Сократ смертен».

Так что формула $F = bba\square a$ является не-теоремой, интерпретируемой как истинное утверждение, т. е. $F \in \overline{T}_1$.

Рассмотрим другую модель формальной системы (*JP*):

a – это число ноль, т. е. вместо *a* будем писать само число 0,

b – знак, указывающий на последующее натуральное число, т. е. запись *ba* читается как «натуральное число, следующее за числом 0»; короче говоря, запись *ba* обозначает число 1; запись *bba* обозначает число 2 и т. д.,

□ – знак равенства двух натуральных чисел, т. е. вместо знака □ будем писать знак =.

Для этой интерпретации выполняются указанные выше условия:

$$\mathcal{F}_M = T_1 \cup \overline{T}_0, \quad T_0 = \varnothing, \quad \overline{T}_1 = \varnothing.$$

Действительно, аксиома *a*□*a* означает 0 = 0. Теорема, например, *bbba*□*bbba* означает 3 = 3. А вот не-теорема *bba*□*a* означает неверное равенство 2 = 0.

ГЛАВА 5
Примеры формальных систем

5.1. Исчисление высказываний

Эта формальная система называется еще *логикой высказываний*, или *пропозициональной логикой*. Формальные системы, как определено выше, представляют собой систему четырех объектов $M =< T, P, A, B >$: T — алфавит, P — множество синтаксических правил, A — множество аксиом, B — множество правил вывода. Следовательно, задать формальную систему — это задать эти четыре объекта.

5.1.1. Определение исчисления высказываний

Логика высказываний определяется следующим образом.

1. *Алфавит*

— *пропозициональные символы* — это буквы латинского алфавита (прописные и строчные) или те же буквы с индексами, например: A_1, A_1, ..., P, Q, a, b, x_1, x_2, ...; эти символы используются для обозначения высказываний, т. е. утверждений о свойствах или взаимосвязях объектов предметной области;

— *логические связки*:
отрицание $\neg$ (читается «не», «отрицание», «неверно, что»); используется для обозначения частицы «не» и всех ее синонимов,
импликация $\rightarrow$ (читается «если ..., то ...», «влечет», «следует», «импликация», «имплицирует»); используется для обозначения словосочетаний «если ..., то ...», «потому что», «так как» и т. п.;

— круглые скобки (и); используются для правильного написания и правильной интерпретации формул.

2. *Синтаксические правила*

— любой пропозициональный символ есть формула;

— если m есть формула, то (m) тоже является формулой;

— если m, m_1 и m_2 являются формулами, то выражения $\neg m$ и $m_1 \rightarrow m_2$ тоже являются формулами.

3. *Аксиомы*

($A1$) $m_1 \rightarrow (m_1 \rightarrow m_2)$,

($A2$) $(m_1 \rightarrow (m_2 \rightarrow m_3)) \rightarrow ((m_1 \rightarrow m_2) \rightarrow (m_1 \rightarrow m_3))$,

($A3$) $(\neg m_2 \rightarrow \neg m_1) \rightarrow (m_1 \rightarrow m_2)$.

В этих аксиомах символы m_1, m_2 и m_3 обозначают произвольные формулы.

4. *Правило вывода* (правило *modus ponens*, или *правило отделения*)

(MP) $\dfrac{m_1 \ , \ m_1 \rightarrow m_2}{m_2}$.

Правило modus ponens означает: если имеются две формулы m_1 и $m_1 \rightarrow m_2$, то можно записать формулу m_2. Другими словами, формула m_2 является логическим выводом из формул m_1 и $m_1 \rightarrow m_2$.

Данное определение довольно экономно, так как использует лишь две связки ($\neg$ и $\rightarrow$) и всего три аксиомы. Однако существует еще более экономное определение этой формальной системы, использующее одну связку (штрих Шеффера) и одну аксиому. Об этом подробнее будет идти речь ниже.

В любой формальной системе, если необходимо, можно определять новые связки и правила образования формул на основе заданных в определении формальной системы. Например, в логике высказываний, как правило, в алфавит вводятся новые логические связки $\wedge$ (конъюнкция), $\vee$ (дизъюнкция) и $\equiv$ (эквивалентность), которые определяются через заданные связки и следующим образом:

$a \wedge b$ обозначает формулу $\neg(a \rightarrow \neg b)$,

$a \vee b$ обозначает формулу $(\neg a \rightarrow b)$,

$a \equiv b$ обозначает формулу $(a \rightarrow b) \wedge (b \rightarrow a)$.

Введение новых логических связок в алфавит логики высказываний сопровождается дополнением списка синтаксических правил новым правилом:

— если m_1 и m_2 являются формулами, то выражения $m_1 \wedge m_2$, $m_1 \vee m_2$ и $m_1 \equiv m_2$ тоже являются формулами.

К правилу вывода (MP) при этом добавляются следующие правила переписывания:

$a \wedge b \models\!\!\Rightarrow \neg(a \rightarrow \neg b)$,

$\neg(a \rightarrow \neg b) \models\!\!\Rightarrow a \wedge b$,

$a \vee b \models\!\!\Rightarrow (\neg a \rightarrow b)$,

$(\neg a \rightarrow b) \models\!\!\Rightarrow a \vee b$,

$a \equiv b \models\!\!\Rightarrow (a \rightarrow b) \wedge (b \rightarrow a)$,

$(a \rightarrow b) \wedge (b \rightarrow a) \models\!\!\Rightarrow a \equiv b$.

Правила переписывания применяются как ко всей формуле в целом, так и к отдельным ее частям. Символ $\models\!\!\Rightarrow$ указывает на то, что соответствующее правило вывода является именно правилом переписывания. Правило вывода (MP) применяется только к формулам в целом, к частям формул его применять нельзя.

Так что логику высказываний можно было сразу определить со всеми пятью связками и с теми дополнениями к списку синтаксических правил и к списку правил вывода, о которых только что шла речь. При этом оба определения приводят к одному и тому же множеству теорем.

К этому же множеству теорем приводит еще более экономное определение логики высказываний, основанное на единственной логической связке $|$ (штрих Шеффера). Логика высказываний в некотором смысле эквивалентна формальной системе, в которой алфавит, синтаксические правила, аксиомы и правила вывода задаются следующим образом.

1. *Алфавит*
   - *пропозициональные символы* — это буквы латинского алфавита (прописные и строчные) или те же буквы с индексами, например: A_1, A_1, ..., P, Q, a, b, x_1, x_2, ...; эти символы используются для обозначения высказываний, т. е. утверждений о свойствах или взаимосвязях объектов предметной области;
   - *логическая связка*: $|$ (штрих Шеффера);
   - круглые скобки (и); используются для правильного написания и правильной интерпретации формул.
2. *Синтаксические правила*
   - любой пропозициональный символ есть формула;
   - если m есть формула, то (m) тоже является формулой;
   - если m_1 и m_2 являются формулами, то выражение $m_1|m_2$ тоже является формулой.
3. *Аксиома*

   $(p|(q|r))|(s|(s|s))|((t|q)|((p|t)|(p|q)))$.

   Здесь символы p, q, r, s и t обозначают произвольные формулы.
4. *Правило вывода*

$$\frac{m_1\ ,\ m_1\big|(m_2\big|m_3)}{m_3}.$$

Эквивалентность описанной формальной системы логике высказываний следует понимать в том смысле, что если во всех теоремах этой формальной системы записи $a|b$ заменить формулой $(\neg a \vee \neg b)$, то полученное множество формул полностью исчерпывает все теоремы логики высказываний.

Этот факт был установлен Ж. Нико в 1917 г. Заметим, что пять логических связок логики высказываний могут быть выражены через штрих Шеффера:

$\neg p$ соответствует формуле $p|p$,

$p \vee q$ соответствует формуле $((p|p)|(q|q))$,

$p \wedge q$ соответствует формуле $((p|q)|(p|q))$,

$p \to q$ соответствует формуле $(p|(q|q))$,

$p \equiv q$ соответствует $((p|(q|q))|(q|(p|p)))|(((p|(q|q))|(q|(p|p)))$.

Имеется еще один экономный вариант определения логики высказываний в сравнении с данным в начале. Он был предложен Я. Лукасевичем и Б. Собочинским и основан на тех же двух связках $\neg$ и $\rightarrow$, но на одной единственной аксиоме при использовании правила вывода modus ponens. Этот вариант определения приводит к тому же множеству формул, что и первоначальное определение логики высказываний.

Введенные выше три связки $\wedge$, $\vee$ и $\equiv$ обогащают логику высказываний, предоставляя дополнительные средства формализации знаний об объектах предметной области. В частности, можно отметить, что три закона логики из большого списка законов, сформулированных Аристотелем, в обогащенной новыми тремя связками логике высказываний выражаются строго доказуемыми формулами (теоремами):

— закон тождества $p \rightarrow p$,

— закон исключенного третьего $p \vee \neg p$,

— закон противоречия $\neg(p \wedge \neg p)$.

В качестве примера доказательства в исчислении высказываний рассмотрим доказательство закона тождества $p \rightarrow p$.

1. $p \rightarrow ((p \rightarrow p) \rightarrow p)$ аксиома $(A1)$ при $m_1 = p$, $m_2 = (p \rightarrow p)$
2. $(p \rightarrow ((p \rightarrow p) \rightarrow p)) \rightarrow ((p \rightarrow (p \rightarrow p)) \rightarrow (p \rightarrow p))$ аксиома $(A2)$ при $m_1 = m_3 = p$, $m_2 = (p \rightarrow p)$
3. $(p \rightarrow (p \rightarrow p)) \rightarrow (p \rightarrow p)$ правило (MP), примененное к формулам (1) и (2)
4. $p \rightarrow (p \rightarrow p)$ аксиома $(A1)$ при $m_1 = m_2 = p$
5. $p \rightarrow p$ правило (MP), примененное к формулам (3) и (4)

Пусть в алфавит логики высказываний введены связки $\wedge$, $\vee$ и $\equiv$, как это описано выше. Некоторые теоремы логики высказываний, играющие в ней важную роль, называют *законами логики высказываний*. Перечислим их.

I. Законы коммутативности

$$(a \vee b) \equiv (b \wedge a),$$
$$(a \wedge b) \equiv (b \wedge a).$$

II. Законы ассоциативности

$$(a \vee (b \vee c)) \equiv ((a \vee b) \vee c),$$
$$(a \wedge (b \wedge c)) \equiv ((a \wedge b) \wedge c).$$

III. Законы дистрибутивности

$$(a \vee (b \wedge c)) \equiv ((a \vee b) \wedge (a \vee c)),$$
$$(a \wedge (b \vee c)) \equiv ((a \wedge b) \vee (a \wedge c)).$$

IV. Законы идемпотентности

$$(a \vee a) \equiv a,$$
$$(a \wedge a) \equiv a.$$

V. Законы де Моргана (законы двойственности)

$$\neg(a \vee b) \equiv (\neg a \wedge \neg b),$$
$$\neg(a \wedge b) \equiv (\neg a \vee \neg b).$$

VI. Закон двойного отрицания

$$\neg\neg a \equiv a.$$

VII. Закон исключенного третьего

$$a \vee \neg a.$$

VIII. Закон противоречия

$$\neg(a \wedge \neg a).$$

В законах логики высказываний символами a, b и c обозначены произвольные формулы, поэтому законы рассматриваются как схемы теорем: подстановка в закон любых формул вместо символов a, b и c приводит к формуле, которая сама является теоремой.

5.1.2. Конъюнктивная и дизъюнктивная нормальные формы

Каждую формулу логики высказываний можно преобразовать в эквивалентную формулу некоторого стандартного вида, в которой используются только связки $\neg$, $\vee$ и $\wedge$. Познакомимся с двумя стандартными видами формул: с конъюнктивной нормальной формой (КНФ) и с дизъюнктивной нормальной формой (ДНФ). Под эквивалентностью двух формул здесь подразумевается их синтаксическая эквивалентность, под преобразованием понимается построение вывода одной формулы из другой. Определим точно эти понятия.

Определение. Формула A называется *синтаксическим следствием* (говорят еще *синтаксически выводима из*) множества формул Γ, что обозначается $\Gamma \vdash A$, если имеется такая последовательность формул $A_1, A_2, \ldots, A_n$, что A_n есть A и для любого $i \in \{1, 2, \ldots, n\}$ формула A_i есть либо аксиома, либо одна из формул множества Γ, либо получается по одному из правил вывода из предыдущих формул $A_1, A_2, \ldots, A_{i-1}$. Последовательность формул $A_1, A_2, \ldots, A_n$ называется *доказательством*, или *выводом*, формулы A из множества Γ. Сами формулы, входящие во множество Γ, называются *посылками*, или *гипотезами*.

Определение. Если $\varnothing \vdash A$, то формула A называется *теоремой* и обозначается $\vdash A$.

Определение. Формула A называется *синтаксически эквивалентной* формуле B, если $A \vdash B$ и $B \vdash A$ справедливо одновременно.

Оказывается, что синтаксическая эквивалентность формул A и B означает $\vdash (A \equiv B)$, т. е. эквивалентность $A \equiv B$ является теоремой. Поэтому понятия синтаксическая эквивалентность и эквивалентность являются синонимами. Это позволяет избегать построения вывода для получения из формулы A ее КНФ или ДНФ. Для того чтобы выполнить преобразование формулы A в КНФ или ДНФ, достаточно использовать законы логики высказываний, строя цепочки эквивалентностей подобно тому, как в алгебре строятся цепочки равенств в ходе алгебраических преобразований. Следующее утверж-

дение по существу является метатеоремой логики высказываний, обосновывающей указанной способ получения КНФ и ДНФ.

Теорема. *Формулы A и B синтаксически эквивалентны, т. е. $A \vdash B$ и $B \vdash A$ справедливы одновременно тогда и только тогда, когда* $\vdash (A \equiv B)$.

Это утверждение является следствием теоремы дедукции, доказанной Ж. Эрбраном (1930 г.).

Теорема дедукции. *Пусть Г — множество формул, A и B — некоторые формулы. Если $\Gamma \vdash \{A\} \vdash B$, то $\Gamma \vdash (A \to B)$. В частности, если $A \vdash B$, то $\vdash (A \to B)$.*

Приведение формул к КНФ и ДНФ проводится с использованием законов логики высказываний и следующих теорем:

$$(a \to b) \equiv (\neg a \vee b),$$
$$(a \equiv b) \equiv ((a \to b) \wedge (b \to a)).$$

Здесь a и b — произвольные формулы.

Определение. Формула вида $a_1 \vee a_2 \vee \ldots \vee a_n \vee \neg b_1 \vee \neg b_2 \vee \ldots \vee \neg b_m$ называется *дизъюнктом*, или *предложением*, а формула вида $a_1 \wedge a_2 \wedge \ldots \wedge a_n \wedge \neg b_1 \wedge \neg b_2 \wedge \ldots \wedge \neg b_m$ называется *конъюнктом*, где $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m$ — пропозициональные символы. При этом $a_1, a_2, \ldots, a_n$ называются *положительными*, а $\neg b_1, \neg b_2, \ldots, \neg b_m$ *отрицательными* членами дизъюнкта или конъюнкта.

У дизъюнктов и конъюнктов могут полностью отсутствовать положительные или отрицательные члены. Это означает, что формулы

$$a_1 \vee a_2 \vee \ldots \vee a_n,$$
$$\neg b_1 \vee \neg b_2 \vee \ldots \vee \neg b_m,$$

в частности, формулы a_1 и $\neg b_1$ тоже являются дизъюнктами. Аналогично формулы

$$a_1 \wedge a_2 \wedge \ldots \wedge a_n,$$
$$\neg b_1 \wedge \neg b_2 \wedge \ldots \wedge \neg b_m,$$

в частности, формулы a_1 и $\neg b_1$ тоже являются конъюнктами.

Определение. *Конъюнктивной нормальной формой* (КНФ) называется формула, представляющая собой конъюнкцию конечного числа дизъюнктов.

Другими словами: КНФ — это конъюнкция предложений. В общем виде КНФ можно представить формулой

$$D_1 \wedge D_2 \wedge \ldots \wedge D_k,$$

где $D_1, D_2, \ldots, D_k$ — предложения (дизъюнкты).

Определение. *Дизъюнктивной нормальной формой* (ДНФ) называется формула, представляющая собой дизъюнкцию конечного числа конъюнктов.

В общем виде ДНФ можно представить формулой

$$C_1 \vee C_2 \vee \ldots \vee C_p,$$

в которой C_1, C_2, ..., C_k — конъюнкты. Вполне возможно, что ДНФ состоит из единственного конъюнкта; точно также КНФ может состоять из единственного дизъюнкта.

Теорема. *Любая формула логики высказываний эквивалентна некоторой КНФ и некоторой ДНФ.*

Справедливость этой теоремы установить несложно. Следующий алгоритм преобразования произвольной формулы в эквивалентную ей КНФ по существу является обоснованием этой теоремы. Тот же алгоритм, как позднее будет показано, можно использовать для преобразования произвольной формулы в эквивалентную ей ДНФ.

5.1.3. Алгоритм преобразования формулы в КНФ и ДНФ

Пусть дана формула A, подлежащая преобразованию в КНФ. Если A — это пропозициональный символ p, либо его отрицание $\neg p$, то ее КНФ состоит из единственного дизъюнкта, каковым является само p, либо $\neg p$. Если же это не так, то надлежит выполнить следующие действия.

1. Исключение из A связок $\rightarrow$ и $\equiv$, используя теоремы:

$$(a \rightarrow b) \equiv (a \vee b),$$
$$(a \equiv b) \equiv ((a \rightarrow b) \wedge (b \rightarrow a)).$$

2. Внесение связки $\neg$ внутрь скобок везде, где это возможно, применяя законы де Моргана:

$$\neg(a \vee b) \equiv (\neg a \wedge \neg b),$$
$$\neg(a \wedge b) \equiv (\neg a \vee \neg b).$$

В результате этих действий связка $\neg$ будет расставлена в формуле A только перед пропозициональными символами или перед их отрицаниями. Вследствие этого могут появиться выражения вида $\neg\neg p$.

3. Удаление двойных отрицаний в соответствии с законом двойного отрицания

$$\neg\neg a \equiv a.$$

4. Применение закона дистрибутивности

$$(a\vee(b\wedge c))\equiv((a\vee b)\wedge(a\vee c))$$

необходимое число раз, пока не будет получена КНФ.

Для получения ДНФ этим же алгоритмом нужно на этапе 4 применять второй из законов дистрибутивности

$$(a\wedge(b\vee c))\equiv((a\wedge b)\vee(a\wedge c))$$

необходимое число раз, пока не будет получена ДНФ.

Пример. Приведем к КНФ следующую формулу:

$$(p\to(q\to r))\to((p\wedge s)\to r).$$

1. Исключение импликаций

$$\neg(\neg p\vee(\neg q\vee r))\vee(\neg(p\wedge s)\vee r).$$

2. Внесение связки $\neg$ внутрь скобок

$$(\neg\neg p\vee(\neg\neg q\wedge\neg r))\vee((\neg p\vee\neg s)\vee r).$$

3. Удаление двойных отрицаний

$$(p\wedge q\wedge\neg r)\vee(\neg p\vee\neg s\vee r).$$

4. Применение закона дистрибутивности $(a\vee(b\wedge c))\vee((a\vee b)\wedge(a\vee c))$

$$(p\vee\neg p\vee\neg s\vee r)\vee(q\vee\neg p\vee\neg s\vee r)\vee(\neg r\vee\neg p\vee\neg s\vee r).$$

Следовательно, исходная формула эквивалентна КНФ $D_1\wedge D_2\wedge D_3$, где
D_1: $p\vee\neg p\vee\neg s\vee r$,
D_2: $q\vee\neg p\vee\neg s\vee r$,
D_3: $\neg r\vee\neg p\vee\neg s\vee r$.

Приведение к ДНФ той же формулы выполняется точно также, и видно, что уже на этапе 3 искомая ДНФ построена, т. е. исходная формула эквивалентна ДНФ $C_1\vee C_2\vee C_3\vee C_4$, где
C_1: $p\wedge q\wedge\neg r$.
C_2: $\neg p)$.
C_3: $\neg s$.
C_4: r.

Конъюнктивная нормальная форма играет важную роль в обработке знаний на ЭВМ: дизъюнкты, входящие в КНФ, являются посылками в принципе резолюции, используемом в качестве единственного правила вывода в механизме вывода языков логического программирования. Например, синтаксической основой языка программирования PROLOG являются предложения Хорна, а его логической основой является принцип резолюции.

5.1.4. Интерпретация логики высказываний

В соответствии с определением интерпретации (модели) φ она является отображением, сопоставляющим символам алфавита T объекты предметной области D, и переводит формулы, в том числе пропозициональные символы, в истинные или ложные утверждения о свойствах объектов предметной области. Поскольку в математической логике имеют значение только истинность или ложность утверждений о свойствах предметной области, то отображение φ: $T \to D$ удобнее рассматривать как отображение $\varphi: \mathcal{F} \to \{0, 1\}$, переводящее множество $\mathcal{F}$ всех формул логики высказываний на множество {0, 1} значений истинности, где 0 — это символ значения истинности «ложь», а 1 — символ значения истинности «истина».

Понятно, что для задания конкретной модели $\varphi: \mathcal{F} \to \{0, 1\}$ достаточно определить φ на пропозициональных символах и задать способ вычисления значений истинности $\varphi(a \to b)$ и $\varphi(\neg a)$, если заданы значения истинности для $\varphi(a)$ и $\varphi(b)$. Таблицы истинности для связок $\to$ и $\neg$ задают способ вычисления значений истинности.

Определение. *Интерпретацией* (*моделью*) логики высказываний называется такое отображение $\varphi: \mathcal{F} \to \{0, 1\}$ множества $\mathcal{F}$ всех формул логики высказываний на множество {0, 1} значений истинности, по которому каждому пропозициональному символу соответствует определенное значение истинности 1 (истина) или 0 (ложь), а каждой формуле вида $a \to b$ или вида $\neg a$ сопоставляется 1 или 0 в зависимости от значений истинности для формул a и b в соответствии с *таблицами истинности*:

a	b	$a \to b$
1	1	1
1	0	0
0	1	1
0	0	1

a	$\neg a$
0	1
1	0

Вместо a и $\neg a$ в шапке таблицы истинности для связки $\neg$ следовало бы писать $\varphi(a)$ и $\varphi(\neg a)$ соответственно. Аналогично, в таблице истинности для связки $\to$ следовало бы писать $\varphi(a)$, $\varphi(b)$ и $\varphi(a \to b)$, а не a, b и $a \to b$ соответствен-

но. Тем не менее, для краткости записи сохраним в этих таблицах и во всех других таблицах истинности использованную сокращенную форму записи.

Из определений связок $\wedge$, $\vee$ и $\equiv$ следует, что их таблицы истинности имеют вид:

a	b	$a \wedge b$
1	1	1
1	0	0
0	1	0
0	0	0

a	b	$a \vee b$
1	1	1
1	0	1
0	1	1
0	0	0

a	b	$a \equiv b$
1	1	1
1	0	0
0	1	0
0	0	1

Определение. Формула A логики высказываний называется *тавтологией*, или *тождественно-истинной формулой*, если для любой модели φ выполняется $\varphi(A) = 1$.

Другими словами, формула A является тавтологией, если для любого набора значений истинности составляющих формулу A пропозициональных символов эта формула согласно таблицам истинности всегда принимает значение истинности 1 (истина).

Упомянутая ранее теорема Э. Поста формулируется так.

Теорема Э. Поста. *Для того чтобы формула логики высказываний была теоремой, необходимо и достаточно, чтобы она являлась тавтологией.*

Эта теорема, по сути, утвердительно отвечает на вопрос, является ли логика высказываний разрешимой формальной теорией. Эта же теорема позволяет установить непротиворечивость и полноту логики высказываний. Так что справедлива следующая теорема.

Теорема. *Логика высказываний является разрешимой, непротиворечивой и полной формальной теорией.*

Этот результат представляет особый интерес, так как он указывает на адекватность логики высказываний классическим логическим рассуждениям. Он утверждает, что мышление и его выражение с помощью языка формализуемы, по крайней мере, частично (ибо не учитывается внутренняя структура утверждений), и представимы в краткой форме с помощью чисто символьных схем и представлений.

5.2. Исчисление предикатов первого порядка

5.2.1. Определение логики предикатов

Формальная система, называемая исчислением предикатов первого порядка, или логикой предикатов первого порядка, является расширением логики высказываний. Логика предикатов дает возможность более детально

рассматривать, а следовательно, более точно формализовать знания и рассуждения о свойствах объектов предметной области.

В логике высказываний пропозициональные символы служат для обозначения простых высказываний (утверждений) о предметной области, представляющих собой с точки зрения русского языка простые (с одной грамматической основой) повествовательные предложения. Логические связки логики высказываний обозначают следующие союзы или словосочетания и их синонимы:

$\neg$ — «не», «неверно, что»,

$\vee$ — «или», «либо»,

$\wedge$ — «и», «а», «но»,

$\rightarrow$ — «если», «при условии, что»,

$\equiv$ — «равносильно», «эквивалентно», «тогда и только тогда, когда».

В силу этого формулы логики высказываний обозначают простые или сложные (с несколькими грамматическими основами) повествовательные предложения, в которых выражены свойства предметной области. Исчисление высказываний не «проникает» внутрь простого высказывания, т. е. не учитывает его внутренней структуры.

Логика предикатов позволяет учесть внутреннюю структуру простых высказываний, обеспечивая тем самым более точное представление знаний и рассуждений о свойствах предметной области. Рассмотрим простое высказывание «5 является целым положительным числом». В логике высказываний оно может быть обозначено единственным пропозициональным символом p. В логике предикатов это простое высказывание может быть представлено формулой $I(5) \wedge P(5)$, в которой использованы одноместные предикаты:

$I(x)$ — означает «x — целое число»,

$P(x)$ — означает «x — положительное число».

Другое простое высказывание «квадрат любого действительного числа положителен» в логике высказываний может быть обозначено единственным пропозициональным символом q. В логике предикатов оно может быть представлено формулой

$$\forall x(R(x) \rightarrow Q(s(x))),$$

в которой

$\forall$ — квантор общности, обозначающий фразы: «все», «любой», «произвольный», «для всех» и т. п.,

$R(x)$ — предикат, означающий «x — действительное число»,

$Q(x)$ — предикат, означающий «x — неотрицательное число»,

$s(x)$ — функция, определяемая равенством $s(x) = x^2$.

Наряду с квантором общности в логике предикатов используется квантор существования $\exists$, который служит для обозначения фраз и словосочетаний: «существует», «имеется», «какой-нибудь», «для некоторого» и т. п.

Таким образом, наряду с пропозициональными символами, связками и скобками, входящими в алфавит логики высказываний, исчисление предикатов использует символы констант, переменных, символы предикатов, функций и кванторы. Более точно логика предикатов первого порядка как формальная система определяется следующим образом.

1. *Алфавит*
   $a, b, c, \ldots$ — константы,
   $x, y, z, \ldots$ — переменные,
   $f, g, h, \ldots$ — функциональные символы; каждый символ характеризуется некоторым натуральным числом m — число аргументов символа; это число называют *арность*, или *местность* символа,
   $P, Q, R, \ldots$ — предикатные символы; каждый символ также характеризуется некоторым натуральным числом n — число аргументов символа; это число называют *арность*, или *местность*, символа,
   $p, q, r, \ldots$ — пропозициональные символы (символы высказываний),
   $\neg, \rightarrow$ — логические связки (отрицание, импликация),
   $\forall$ — квантор общности,
   0, 1 — символы значений истинности: 0 — «ложь» и 1 — «истина»
   (,) — скобки; используются для правильной записи и интерпретации формул.
2. *Синтаксические правила*

— символы 0 и 1 являются формулами,
— любой пропозициональный символ есть формула,
— если A есть формула, то (A) тоже является формулой,
— если A, A_1 и A_2 являются формулами, то выражения $\neg A$, $A_1 \wedge A_2$, $A_1 \vee A_2$, $A_1 \rightarrow A_2$ и $A_1 \equiv A_2$ тоже являются формулами,
— если P — n-местный предикатный символ, $x_1, x_2, \ldots, x_n$ — переменные, то выражение $P(x_1, x_2, \ldots, x_n)$ является формулой,
— если $A(x_1, x_2, \ldots, x_n)$ — формула, в которой отсутствуют кванторы, то выражение $\forall x_1 A(x_1, x_2, \ldots, x_n)$ является формулой, при этом в ней x_1 называется *связанной переменной*, а остальные переменные называются *свободными*.

3. *Аксиомы*
   $(A1)\ A \rightarrow (A \rightarrow B)$
   $(A2)\ (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
   $(A3)\ (\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$
   $(A4)\ (\forall t B(t)) \rightarrow B(u)$, где u — переменная и $B(t)$ не содержит u в качестве связанной переменной,
   $(A5)\ (\forall t (A \rightarrow B)) \rightarrow (A \rightarrow \forall t (B))$, где A и B являются формулами, а t не является свободной переменной в формуле A.
   Заметим, что первые три аксиомы такие же, как в логике высказываний.

4. *Правила вывода*

$(MP)\ \dfrac{A,\ A \to B}{B}$ — правило *modus ponens*,

$(GN)\ \dfrac{A}{\forall t(A)}$ — правило обобщения, где t — свободная переменная в A.

5.2.2. Описание алфавита логики предикатов

В алфавите логики предикатов *константы* служат для обозначения конкретных значений переменных, например: для обозначения точки на плоскости с координатами (0;0) или отрезка на плоскости с концами в точках (–1;2) и (3;7), или прямоугольного треугольника со сторонами длиной 3, 4 и 5 см.

Переменные служат для обозначения объектов предметной области. Например, в геометрии переменные могут обозначать точки, отрезки, треугольники и т. д.

Функциональные символы используются для обозначения правил образования одних объектов предметной области из других. Например, функциональный символ sin обозначает правило вычисления нового числа $\sin(x)$ по величине угла x. Функциональным символом f, например, можно обозначить правило, ставящее в соответствие каждому человеку t его отца. В таком случае запись $f(t)$ обозначает отца человека t. Обратим внимание, что для функциональных символов всегда используются строчные латинские буквы в отличие от предикатных символов, для которых всегда используются прописные (заглавные) буквы латинского алфавита.

Предикатные символы применяются для обозначения свойств объектов предметной области или для обозначения отношений между этими объектами. Например, если предикатный символ P обозначает свойство «быть дипломатом», то предикат $P(x)$ означает «x является дипломатом». Или, если Q обозначает отношение «проживать в одном городе», то предикат $Q(u, v)$ означает «некто u проживает в одном городе с v».

Пропозициональные символы служат для обозначения высказываний (простейших утверждений о свойствах и взаимосвязях конкретных объектов предметной области). Высказывания не содержат переменных. Поэтому можно считать, что пропозициональные символы служат для обозначения знаний о конкретных объектах предметной области. Если в предикатах

$P(x)$ — «x является дипломатом»

и $Q(u, v)$ — «некто u проживает в одном городе с v»

подставить константу a = «Андрей» вместо переменной x, подставить константы b = «Алексей» и c = «Александр» вместо переменных u и v соответственно, то получатся высказывания $P(a)$ («Андрей является дипломатом») и $Q(b, c)$ («Алексей проживает в одном городе с Александром»), которые мож-

но обозначить пропозициональными символами соответственно p = «Андрей является дипломатом» и q = «Алексей проживает в одном городе с Александром». Заметим, что здесь под именем «Андрей» подразумеваются не все люди с этим именем, а только какой-то один из них. Так же и с другими именами.

Логические связки обозначают операции, позволяющие из одних утверждений об объектах предметной области конструировать новые, более сложные, утверждения с использованием отрицательной частицы «не», союзов «и», «или», «если... то...» и словосочетания «тогда и только тогда, когда». Например, признак подобия треугольников:

два треугольника подобны тогда и только тогда, когда два угла одного треугольника соответственно равны двум углам другого треугольника —

образован из двух утверждений

p = «два треугольника подобны»

и q = «два угла одного треугольника соответственно равны двум углам другого треугольника»

с использованием словосочетания «тогда и только тогда, когда». Заменяя это словосочетание логической связкой $\equiv$ (эквивалентность), признак подобия треугольников можно упрощенно представить формулой логики предикатов $p \equiv q$.

Квантор общности $\forall$ и *квантор существования* $\exists$ используются для обозначения фраз «для всех» и «существует» соответственно и их синонимов, связывающих переменные в утверждениях об объектах предметной области. Например, в таких:

Все ели — зеленые.

Среди всех чисел имеются неотрицательные числа.

Любая акция является ценной бумагой.

Последнее утверждение можно записать в виде формулы логики предикатов $\forall x\,(A(x) \to C(x))$, если предикат $A(x)$ означает «x является акцией», а предикат $C(x)$ означает «x — ценная бумага».

Наконец, символ 0 (*значение истинности «ложь»*) и символ 1 (*значение истинности «истина»*) служат для обозначения соответственно любого ложного и любого истинного утверждения об объектах предметной области.

В логику предикатов введем новые логические связки $\wedge$ (конъюнкция), $\vee$ (дизъюнкция), $\equiv$ (эквивалентность) точно так же, как это делается в логике высказываний. Квантор существования $\exists$ вводится в логику предикатов через квантор общности так: формула $\exists t B(t)$ служит для обозначения формулы $\neg(\forall t \neg B\,(t))$.

В названии формальной системы «логика предикатов первого порядка» фраза «...первого порядка» означает, что квантором общности можно связывать только переменные. Квантор общности нельзя применять к предикатным символам. В исчислении предикатов второго порядка квантор общности допускается применять также к предикатным символам. В исчислении

предикатов более высокого порядка квантор общности может применяться к предикатным символам, обозначающим предикаты, аргументами которых являются другие предикаты, и т. д.

5.2.3. Синтаксис логики предикатов

Перейдем теперь к описанию синтаксиса, т. е. правил образования «слов» формального языка логики предикатов. Проводя аналогию с русским языком, скажем, что формальный язык логики предикатов имеет только две «части речи»: термы и атомарные формулы. В связи с введением дополнительных логических связок список синтаксических правил следует дополнить следующими определениями термов, атомарных формул и формул.

Определение. Если символом V обозначить множество всех переменных $x, y, z, \ldots$ и констант $a, b, c, \ldots$, то множество T всех термов определяется как наименьшее из множеств, удовлетворяющих условиям:

(τ_1) $V \in T$ (т. е. все переменные и константы являются термами),

(τ_2) если $t_1, t_2, \ldots, t_n \in T$ и f — n-местный функциональный символ, то $f(t_1, t_2, \ldots, t_n) \in T$.

Термы языка предназначены для обозначения терминов, понятий и конкретных объектов или групп объектов предметной области. Можно сказать, что в формальном языке логики предикатов одни термы играют ту же роль, что существительные и местоимения в русском языке, а другие термы — роль определений одних понятий через другие понятия. Термы можно интерпретировать как наименования объектов, входящих в какую-нибудь предметную область. Так, в биологии объекты: ель, сосна, климат, хвоя, корень — в формальном языке логики предикатов представляются некоторыми термами. Такое истолкование термов относится в первую очередь к переменным и константам (см. (τ_1) в определении). Другой вид термов можно интерпретировать как наименования объектов, определяемых через другие объекты предметной области или конструируемых по другим объектам, входящим в предметную область (см. (τ_2) в определении). Например, такими объектами в геометрии являются *отрезки*, *ломаные*, *треугольники* и т. д., определяемые через простейшие объекты: *точки*, *прямые* и *плоскости*.

Другая «часть речи» формального языка логики предикатов — *атомарные формулы* — служит для обозначения простейших знаний об объектах предметной области. В русском языке знания о предметной области записываются в форме простых (с одной грамматической основой) или сложных (с двумя или более грамматическими основами) повествовательных предложений, в которых выражаются свойства или отношения между объектами предметной области. Теперь можно сказать более точно: атомарные формулы формального языка логики предикатов служат для представления тех знаний

об объектах предметной области, которые выражаются в форме простых, с одной грамматической основой, повествовательных предложений русского языка. Например, знания, взятые из биологии:

ель — зеленое растение,
тропический климат влажен,
сосна является растением,
хвоя сосны длиннее, чем у ели,

— можно представить некоторыми атомарными формулами языка.

Определение. *Атомарной формулой* называется:

(α_1) любой пропозициональный символ p,

(α_2) если $t_1, t_2, ..., t_n$ — термы и P — n-местный предикатный символ, то $P(t_1, t_2, ..., t_n)$ является атомарной формулой.

По существу, правила (α_1), (α_2), (τ_1), (τ_2) являются правилами образования «слов» формального языка логики предикатов. Из слов в русском языке образуются предложения. Похожее происходит и в формальном языке логики предикатов. Роль предложений в нем играют *формулы*.

Формулы служат для представления знаний об объектах предметной области, которые на естественном языке записываются как простые или сложные повествовательные предложения. В этих предложениях высказываются утверждения о свойствах и взаимосвязях объектов предметной области.

Определение. *Формулой* называется:

(φ_1) любая атомарная формула, символ лжи и символ истины,

(φ_2) если A и B — формулы, а x — переменная, то выражения $\neg A$, $A \vee B$, $A \wedge B$, $A \rightarrow B$, $A \equiv B$, $\forall x A$, $\exists x A$ являются формулами.

Определения понятий «терм», «атомарная формула» и «формула», по сути, являются синтаксическими правилами. Охарактеризуем теперь семантику формального языка логики предикатов, опуская строгие определения.

5.2.4. Семантика логики предикатов

Под *семантикой* языка логики предикатов подразумеваются правила перевода термов и формул этого языка на содержательный, например русский, язык или на другой формальный язык. Благодаря такому переводу термам ставятся в соответствие определенные объекты предметной области, а формулам ставятся в соответствие утверждения об объектах предметной области. Тем самым выясняется, какие формулы формального языка логики предикатов соответствуют истинным утверждениям, а какие формулы — ложным утверждениям об объектах предметной области.

Некоторые теоремы логики предикатов, играющие в ней важную роль, называют *законами логики предикатов*. Ясно, что законами логики предикатов являются перечисленные выше законы I–VIII логики высказываний. До-

полним их теми законами логики предикатов, которые позволяют все кванторы произвольной формулы выносить в начало этой формулы.

IX. $(\neg(\forall x A\,(x))) \equiv (\exists x\,(\neg A\,(x)))$,
$(\neg(\exists x A\,(x))) \equiv (\forall x(\neg A\,(x)))$.

X. $((\forall x A\,(x))\vee B) \equiv (\forall y\,(A\,(y)\vee B))$,
$((\exists x A\,(x))\vee B) \equiv (\exists y\,(A\,(y)\vee B))$,
$((\forall x A\,(x))\wedge B) \equiv (\forall y\,(A\,(y)\wedge B))$,
$((\exists x A\,(x))\wedge B) \equiv (\exists y\,(A\,(y)\wedge B))$.
Здесь считается, что переменная y не содержится в формуле B.

XI. $((\forall x A\,(x)) \vee (\forall x B\,(x))) \equiv (\forall x\forall y\,(A(x)\vee B\,(y)))$, где переменная y не содержится ни в формуле $A\,(x)$, ни в формуле $B\,(x)$,
$((\exists x A\,(x)) \vee (\exists x B\,(x))) \equiv (\exists x\,(A\,(x)\vee B\,(x)))$,
$((\forall x A\,(x)) \wedge (\forall x B\,(x))) \equiv (\forall x\,(A(x)\wedge B\,(x)))$,
$((\exists x A\,(x)) \wedge (\exists x B\,(x))) \equiv (\exists x\exists y\,(A(x)\wedge B\,(y)))$, где переменная y не содержится ни в формуле $A\,(x)$, ни в формуле $B\,(x)$.

Эти законы позволяют любую формулу логики предикатов преобразовать к следующему виду:

$$(Q_1x_1)\,(Q_2x_2)\,\ldots\,(Q_nx_n)F\,(x_1, x_2, \ldots, x_n),$$

где любое Q_i — это либо квантор общности, либо квантор существования, а $F\,(x_1, x_2, \ldots, x_n)$ — формула, не содержащая кванторов. В формуле такого вида можно введением *сколемовских констант* и *функций* удалить все кванторы существования. При этом следует пользоваться следующими законами логики предикатов:

XII. $(\exists x)\,(Q_1y_1)\,\ldots\,(Q_my_m)F\,(y_1, y_2, \ldots, y_m) \equiv (Q_1y_1)\,\ldots\,(Q_my_m)F\,(a, y_1, \ldots, y_m)$, где $Q_1, \ldots, Q_m$ — произвольные кванторы, а символ константы a вводится в алфавит в качестве именно той константы, которая существует в соответствии с выражением $(\exists x)$, стоящим в начале формулы; константу a называют *сколемовской константой*.

XIII. $(\forall x_1)\,(\forall x_2)\ldots\,(\forall x_k)\,(\exists y)\,(Q_1z_1)\ldots\,(Q_mz_m)F\,(x_1, x_2, \ldots, x_k, y, z_1, \ldots, z_m) \equiv$
$\equiv (\forall x_1)\,(\forall x_2)\ldots\,(\forall x_k)\,(Q_1z_1)\ldots\,(Q_mz_m)F\,(x_1, x_2, \ldots, x_k, f(x_1, x_2, \ldots, x_k), z_1, \ldots, z_m)$, где $Q_1, \ldots, Q_m$ — произвольные кванторы, а k-местный функциональный символ f вводится в алфавит в качестве именно того функционального символа, который обозначает функцию, соответствующую выражению $(\forall x_1)\,(\forall x_2)\ldots\,(\forall x_k)\,(\exists y)$, стоящему в начале формулы; эту функцию $f\,(x_1, x_2, \ldots, x_k)$ называют *сколемовской функцией*.

XIV. $\exists x A(x) \equiv A(a)$, где a — сколемовская константа.

Законы XI–XIV обеспечивают преобразование любой формулы логики предикатов к *сколемовской форме*. Так называют формулу вида $\forall x_1\forall x_2\ldots\,\forall x_n A$, в которой формула A вообще не содержит кванторов, а содержит переменные, константы, сколемовские константы и сколемовские функции. Ско-

лемовская форма $\forall x_1 \forall x_2 ... \forall x_n A$, в которой формула A представляет собой КНФ, называется *клаузальной формой*. Клаузальная форма имеет следующий общий вид:

$$\forall x_1 \forall x_2 ... \forall x_n (D_1 \wedge D_2 \wedge ... \wedge D_m).$$

Здесь D_1, D_2, ..., D_m — дизъюнкты, которые называют *клаузами*, или *предложениями*.

Пример. Преобразуем формулу
$\forall x ((A(x) \wedge \neg B(x)) \rightarrow \exists y (C(x, y) \wedge D(y)))$ в клаузальную форму.

1. $\forall x (\neg(A(x) \wedge \neg B(x)) \wedge \exists y (C(x, y) \wedge D(y)))$ получается с помощью теоремы $(m_1 \rightarrow m_2) \equiv (\neg m_1 \vee m_2)$.
2. $\forall x ((\neg A(x) \vee \neg\neg B(x)) \vee \exists y (C(x, y) \wedge D(y)))$ следует из закона двойственности для конъюнкции.
3. $\forall x (\neg A(x) \vee B(x) \vee \exists y (C(x, y) \wedge D(y)))$ вытекает из закона двойного отрицания.
4. $\forall x \exists z (\neg A(x) \vee B(x) \vee (C(x, z) \wedge D(z)))$ вытекает из закона X.
5. $\forall x (\neg A(x) \vee B(x) \vee (C(x, f(x)) \wedge D(f(x))))$ следует из закона XIII, здесь $f(x)$ — сколемовская функция. Эта формула является сколемовской формой.
6. $\forall x ((\neg A(x) \vee B(x) \vee C(x, f(x))) \wedge (\neg A(x) \vee B(x) \vee D(f(x))))$ следует из закона дистрибутивности. Эта формула является клаузальной формой.

Именно клаузальные формы используются в языке логического программирования PROLOG для представления знаний и рассуждений. Клаузы в этом языке, прежде чем будут представлены в форме прологовских предложений, преобразуются в хорновские предложения. *Хорновские предложения* — это клаузы, не содержащие положительных членов вообще либо содержащие только один положительный член. Хорновские предложения (дизъюнкты, клаузы) имеют одни из следующих трех видов:

$\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_n \vee A$ — *правило*,
$\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_n$ — *вопрос*,
A — *факт*.

Эти виды хорновских предложений на языке PROLOG записываются в форме соответственно:

$$A :- A_1, A_2, ..., A_n.$$
$$A_1, A_2, ..., A_n.$$
$$A.$$

Характеризуя исчисление предикатов в целом, отметим, что эта формальная система является непротиворечивой и неразрешимой. Что касается полноты, то для логики предикатов справедлива следующая теорема.

Теорема Гёделя о полноте. *В исчислении предикатов первого порядка все теоремы являются тавтологиями.*

Это означает, что в исчислении предикатов первого порядка все теоремы являются истинными во всех интерпретациях.

5.3. Формальная арифметика

Формальная арифметика разработана Пеано и является расширением исчисления предикатов первого порядка. В формальной арифметике по сравнению с исчислением предикатов дополнительно вводятся одна константа, три функциональных символа, один предикатный символ, одно правило построения формул и девять аксиом.

1. *Дополнительные символы алфавита*
   0 — константа нуль,
   σ — функциональный одноместный символ, означающий «непосредственно следующий за», например, σ (0) — это 1, σ (σ (0)) — это 2 и т. д.
   + — функциональный двухместный символ «плюс»; соответствует операции сложения; вместо + (x, y) принято записывать $x+y$,
   · — функциональный двухместный символ «произведение»; соответствует операции умножения; вместо · (x, y) принято записывать $x \cdot y$,
   = — двухместный предикатный символ «равно», используемый только для сравнения чисел и играющий особую роль среди других предикатных символов; вместо = (x, y) принято записывать $x = y$.
2. *Дополнительное правило построения формул*

— если t_1, t_2 — два числа, построенных с помощью числа 0 и операций σ, + и ·, то выражение $t_1 = t_2$ является формулой.

3. *Дополнительные аксиомы*
   ($A6$) $\forall x(x + 0 = x)$
   ($A7$) $\forall x(x \cdot 0 = 0)$
   ($A8$) $\forall x\neg(\sigma(x) = 0)$
   ($A9$) $\forall x \approx y(x + \sigma(y) = \sigma(x + y))$
   ($A10$) $\forall x \approx y(x \cdot \sigma(y) = x \cdot y + x)$
   ($A11$) $\forall x \approx y((\sigma(x) = \sigma(y)) \rightarrow (x = y))$
   ($A12$) $\forall x \approx y((x = y) \rightarrow (\sigma(x) = \sigma(y)))$
   ($A13$) $\forall x \approx y \approx z\ ((x = y) \rightarrow ((x = z) \rightarrow (y = z)))$
   ($A14$) $(A(0) \wedge (\forall u(A(u) \rightarrow A(\sigma(u)))) \rightarrow \approx uA(u)$
   Здесь x, y, z, u — числа, построенные с помощью числа 0 и операций σ, + и ·, а A (u) — произвольный одноместный предикат с числовым аргументом.

4. *Правила вывода* в формальной арифметике те же (*MP*) и (*GN*), что и в логике предикатов.

Заметим, что в последней (*A14*) аксиоме формализовано рекуррентное рассуждение, называемое *принципом математической индукции*. Из аксиом формальной арифметики вытекают такие, например, свойства, как коммутативный и ассоциативный законы умножения и сложения, а также дистрибутивный закон умножения относительно сложения:

$$x + y = y + x,$$
$$x + (y + z) = (x + y) + z,$$
$$x \cdot y = y \cdot x,$$
$$x \cdot (y \cdot z) = (x \cdot y) \cdot z,$$
$$x \cdot (y + z) = x \cdot y + x \cdot z.$$

В формальной арифметике доказуемы все известные в элементарной арифметике утверждения и теоремы. В целом формальная арифметика является непротиворечивой, неразрешимой и неполной формальной теорией. Последнее означает, что в ней имеются формулы, для которых нельзя доказать ни саму эту формулу, ни ее отрицание. Причем, оказывается, что если A — именно такая формула, то даже добавление A в список аксиом формальной арифметики не делает эту теорию полной. Другими словами, формальная арифметика является неполной и непополнимой теорией.

5.4. Продукционные системы

Продукцией называется выражение вида

$$(i);\ Q;\ P;\ A \to B;\ N,$$

в котором i — имя продукции, Q — указатель сферы применения продукции, P — предисловие, т. е. условие применения продукции, $A \to B$ — ядро продукции, N — постусловие, т. е. те действия, которые необходимо выполнить, если ядро продукции было активизировано. Некоторые элементы продукций, за исключением ядра, могут не использоваться в продукционных системах. Определяемая ниже формальная система служит для представления именно таких продукционных систем. Точнее, тех из них, в которых продукции состоят из единственного элемента — ядра продукции $A \to B$. Эту формальную систему называют *формальной продукционной системой*, или формальной системой продукций.

Формальная продукционная система является расширением другой формальной системы — логики высказываний, в которую вводятся дополнительные аксиомы (для каждого факта и каждой продукции по одной аксиоме). *Формальная система продукций* определяется следующим образом.

1. *Алфавит* наряду со связками $\neg$, $\rightarrow$ и скобками (,) содержит ровно столько пропозициональных символов, сколько различных посылок и заключений имеется в продукциях формализуемой продукционной системы.
2. *Синтаксические правила*, т. е. правила построения формул, те же, что и в логике высказываний.
3. *Аксиомы* включают все аксиомы логики высказываний, а также аксиомы:
   $(F_1)\ p_1 \quad (R_1)\ q_1 \rightarrow r_1$
   $(F_2)\ p_2 \quad (R_2)\ q_2 \rightarrow r_2$
   ....
   $(F_n)\ p_n \quad (R_m)\ q_m \rightarrow r_m$,
   соответствующие всем фактам и правилам, имеющимся в продукционной системе. Здесь p_i — обозначение i-го факта базы данных продукционной системы, а q_j и r_j — обозначения посылки и заключения j-й продукции базы знаний продукционной системы.
4. *Правило вывода* то же, что в логике высказываний — modus ponens.

Возможность такой формализации продукционной системы указывает на возможность представления и реализации целей продукционной системы средствами языка логического программирования PROLOG.

Библиографический список

1. Стерлинг Л. Искусство программирования на языке Пролог: пер. с англ. / Л. Стерлинг, Э. Шапиро. — М.: Мир, 1990.
2. Марселлус Д. Программирование экспертных систем на ТУРБО-ПРОЛОГЕ / Д. Марселлус. — М.: Финансы и статистика, 1994.
3. Амамия М. Архитектура ЭВМ и искусственный интеллект: пер. с япон. / М. Амамия, Ю. Танака. — М.: Мир, 1993.
4. Ковальски Р. Логика в решении проблем / Р. Ковальски. — М.: Наука, 1990.
5. Левин Р. Практическое введение в технологию искусственного интеллекта и экспертных систем с иллюстрациями на Бейсике / Р. Левин, Д. Дранг, Б. Эделсон. — М.: Финансы и статистика, 1990.
6. Логический подход к искусственному интеллекту: от классической логики к логическому программированию: пер. с фр./А. Тейз, П. Грибомон, Ж. Луи, Д. Снийерс, П. Водон, П. Гоше, Э. Грегуар, Э. Санчес, Ф. Дельсарт. — М.: Мир, 1990.
7. Ин Ц. Использование Турбо-Пролога: пер. с англ. / Ц. Ин, Д. Соломон. — М.: Мир, 1993.
8. Лорьер Ж.-Л. Системы искусственного интеллекта: пер. с фр. / Ж.-Л. Лорьер. — М.: Мир, 1991.
9. Братко И. Программирование на языке Пролог для искусственного интеллекта: пер. с англ. / И. Братко. — М.: Мир, 1990.

Рекомендуемая литература

1. Толковый словарь по искусственному интеллекту / Авторы-составители: А. Н. Аверкин, М. Г. Гаазе-Рапопорт, Д. А. Поспелов. — М.: Радио и связь, 1992.
2. Адаменко А. Н. Логическое программирование и Visual Prolog / А. Н. Адаменко, А. М. Кучуков. — Санкт-Петербург: BHV–СПб, 2003.
3. Будущее искусственного интеллекта: сборник статей / Редакторы-составители: К. Е. Левитин, Д.А Поспелов. — М.: Наука, 1991.

4. Герман О. В. Введение в теорию экспертных систем и обработку знаний: учебное пособие для вузов по специальности «Автоматизированные системы обработки информации и управления» / О. В. Герман. — Мн.: Дизайн ПРО, 1995.
5. Грановская Р. М. Интуиция и искусственный интеллект / Р. М. Грановская, И. Я. Березная. — Л.: Изд-во ЛГУ, 1991.
6. Зенкин А. А. Когнитивная компьютерная графика / А. А. Зенкин. — М.: Наука, 1991.
7. Искусственный интеллект: В 3 кн. Кн. 1. Системы общения и экспертные системы: справочник / под ред. Э. В. Попова. — М.: Радио и связь, 1990.
8. Искусственный интеллект: В 3 кн. Кн. 2. Модели и методы: справочник / под ред. Д. А. Поспелова. — М.: Радио и связь, 1990.
9. Искусственный интеллект: В 3 кн. Кн. 3. Программные и аппаратные средства: справочник / под ред. В. Н. Захарова, В. Ф. Хорошевского. — М.: Радио и связь, 1990.
10. Ларичев О. И. Теория и методы принятия решений, а также Хроника событий в Волшебных странах: учебник / О. И. Ларичев. 2-е изд., перераб. и доп. — М.: Логос, 2002.
11. Макаллистер Дж. Искусственный интеллект и Пролог на микроЭВМ: пер. с англ. / Дж. Макаллистер; под ред. М. В. Сергиевского. — М.: Машиностроение, 1990.
12. Малпас Дж. Реляционный язык Пролог и его применение: пер. с англ. / Дж. Малпас. — Новосибирск: Наука, 1990.
13. Мински М. Фреймы для представления знаний / М. Мински. — М.: Энергия, 1979.
14. Нейлор К. Как построить свою экспертную систему / К. Нейлор. — М.: Энергоатомиздат, 1991.
15. Нильсон Н. Принципы искусственного интеллекта: пер. с англ. / Н. Нильсон. — М.: Радио и связь, 1985.
16. Осуга С. Обработка знаний / С. Осуга. — М.: Мир, 1989.
17. Представление и использование знаний: пер. с япон. / под ред. Х. Уэно, М. Исидзука. — М.: Мир, 1990.
18. Приобретение знаний: пер. с япон. / под ред. С. Осуга, Ю. Саэки. — М.: Мир, 1990.
19. Проектирование экономических экспертных систем: учеб. пособие / под ред. А. М. Романова — М.: Компьютер; ЮНИТИ, 1996.
20. Таунсенд К. Проектирование и программная реализация эспертных систем на персональных ЭВМ / К. Таунсенд, Д. Фохт. — М.: Финансы и статистика, 1990.

21. Тельнов Ю. Ф. Интеллектуальные информационные системы в экономике: учебное пособие. Серия «Информатизация России на пороге XXI века» / Ю. Ф. Тельнов. — М.: СИНТЕГ, 1998.
22. Уинстон П. Искусственный интеллект: пер. с англ. / П. Уинстон. — М.: Мир, 1980.
23. Уотермен Д. Руководство по экспертным системам / Д. Уотермен. — М.: Мир, 1989.
24. Чень Ч. Математическая логика и автоматическое доказательство теорем / Ч. Чень. — М.: Наука, 1983.
25. Частиков А. П. Разработка экспертных систем. Среда CLIPS / А. П. Частиков, Т. А. Гаврилова, Д. А. Белов. — СПб.: БХВ-Петербург, 2003.
26. Черноруцкий И. Г. Методы принятия решений / И. Г. Черноруцкий. — СПб.: БХВ-Петербург, 2005.
27. Элти Дж. Экспертные системы: концепции и примеры / Дж. Элти, М. Кумбс. — М.: Финансы и статистика, 1987.
28. Янсон А. Турбо-Пролог в сжатом изложении: пер. с нем. / А. Янсон. — М.: Мир, 1991.

Приложение

Вопросы к итоговому испытанию

1. Понятие «искусственный интеллект», предмет и методы исследования в искусственном интеллекте.
2. Основные задачи искусственного интеллекта.
3. Направления исследований в искусственном интеллекте.
4. Основные понятия искусственного интеллекта.
5. Понятие «знание», виды знаний.
6. Формализованное представление задачи с помощью пространства состояний.
7. Формализованное представление задачи с помощью пространства подзадач.
8. Графическое представление пространств поиска: пространства состояний и пространства подзадач.
9. Стратегия поиска в глубину.
10. Стратегия поиска в ширину.
11. Эвристический поиск. Алгоритм *A**.
12. Сравнение вариантов алгоритма *A**.
13. Эвристический поиск. Алгоритм программы *GPS*.
14. Аксиоматический метод в геометрии.
15. Определение и свойства формальной системы.
16. Свойства формальных теорий. Понятие метатеории.
17. Понятие алгоритма и разрешимости теории.
18. Доказуемость и истинность: соотношение между ними.
19. Определение исчисления высказываний.
20. Конъюнктивная и дизъюнктивная нормальные формы формул логики высказываний.
21. Алгоритм преобразования формулы логики высказываний в КНФ и ДНФ.
22. Интерпретация логики высказываний. Таблицы истинности.
23. Определение логики предикатов первого порядка.
24. Алфавит логики предикатов первого порядка.
25. Синтаксис логики предикатов первого порядка.

26. Семантика логики предикатов первого порядка.
27. Законы логики предикатов первого порядка.
28. Сколемовские и клаузальные формы формул логики предикатов.
29. Алгоритм преобразования формулы логики предикатов в клаузальную форму.
30. Хорновские дизъюнкты и предложения языка программирования PROLOG.

Учебное издание

Иванов Владимир Михайлович

ИНТЕЛЛЕКТУАЛЬНЫЕ
СИСТЕМЫ

Редактор Н. П. Кубыщенко
Верстка О. П. Игнатьевой

Подписано в печать 26.01.2015. Формат 70×100/16.
Бумага писчая. Плоская печать. Гарнитура Newton.
Уч.-изд. л. 4,2. Усл. печ. л. 7,4. Тираж 100 экз.
Заказ 16.

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ
620049, Екатеринбург, ул. С. Ковалевской, 5
Тел.: 8 (343) 375-48-25, 375-46-85, 374-19-41
E-mail: rio@urfu.ru

Отпечатано в Издательско-полиграфическом центре УрФУ
620075, Екатеринбург, ул. Тургенева, 4
Тел.: 8 (343) 350-56-64, 350-90-13
Факс 8 (343) 358-93-06
E-mail: press-urfu@mail.ru