МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. М.В. ЛОМОНОСОВА

Факультет
вычислительной математики
и кибернетики

В.Б. Кудрявцев, Э.Э. Гасанов, А.С. Подколзин

ВВЕДЕНИЕ В ТЕОРИЮ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Москва
2006

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М.В. ЛОМОНОСОВА

Факультет вычислительной математики и кибернетики

В.Б. Кудрявцев, Э.Э. Гасанов, А.С. Подколзин

ВВЕДЕНИЕ В ТЕОРИЮ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Учебное пособие по курсу
«Теория интеллектуальных систем»

МАКС ПРЕСС

МОСКВА – 2006

УДК 378(075.8):519.71
ББК 22.18я73
К88

Печатается по решению
Редакционно-издательского совета
факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова

Рецензенты:
Алексеев В.Б. профессор, д.ф.-м.н.
Козлов В.Н., профессор, д.ф.-м.н.

Кудрявцев В.Б., Гасанов Э.Э., Подколзин А.С.

К88 **Введение в теорию интеллектуальных систем**: Учебное пособие. – М.: Издательский отдел факультета ВМиК МГУ им. М.В. Ломоносова (лицензия ИД N 05899 от 24.09.2001 г.); МАКС Пресс, 2006. – 208 с.
ISBN 5-89407-272-7
ISBN 5-317-01743-2

Учебное пособие написано на основе специального курса «Теория интеллектуальных систем», читаемого на кафедре математической теории интеллектуальных систем механико-математического факультета МГУ им. М.В. Ломоносова. В книге дается представление об основных разделах теории интеллектуальных систем, таких как распознавание образов, теория баз данных и математическая логика.

Для студентов, аспирантов, специализирующихся в области математической кибернетики, дискретной математики и математической информатики.

УДК 378(075.8):519.71
ББК 22.18я73

ISBN 5-89407-272-7
ISBN 5-317-01743-2

Оглавление

Введение

Суть понятия управляющей системы составляют следующие компоненты: элементы, схемы элементов, их функционирование и среда, с которой взаимодействует схема.

Элементы имеют входные и выходные каналы, через которые осуществляется связь с ними. Из элементов соединением входов одних из них с выходами других строятся схемы. Они также имеют входы и выходы. Такая схема реализует во времени (для определенности — дискретном) некоторый рекуррентный процесс ее взаимодействия со средой. Он определяется рядом допущений.

Элемент имеет состояния, которые меняются во времени за счет влияния текущего значения его входов и состояний в предыдущий момент. Эти значения и состояния определяют значения входов элементов. Заданность состояний элементов схемы и значения ее выходов, таким образом, могут определять текущие значения выходов ее элементов, а, значит, и состояния элементов в следующий момент. Таким образом, среда, формируя значения входов схемы, определяет при заданных начальных состояниях элементов значения выходов элементов и схемы в текущий момент и значения состояний элементов в последующий момент. Цикл взаимодействия схемы со средой повторяется и т.д. Этот процесс называют функционированием схемы.

Под управляющей системой может пониматься описанная схема, функционирующая в среде при взаимодействии с последней.

Функционирование схемы называют также поведением управляющей системы в среде.

Важно отметить, что все преобразователи материальной и абстрактной информации с этой точки зрения являются управляющими системами и могут рассматриваться как конкретные их виды. Примерами подобных объектов являются электрические цепи, формулы, автоматы, живые клетки и т.д.

Из всех реальных видов управляющих систем, пожалуй, наиболее полно вобрала в себя черты управляющей системы модель автомата, поскольку она фактически получается путем лишь сужения множеств значений всех параметров управляющей системы до конечных множеств и уточнения видов схем до композиций элементов, вообще говоря, допускающих перекоммутацию; при этом автомат функционирует вообще говоря неограниченное время.

При аппроксимативном подходе к изучению управляющих систем, состоящем в допущении только конечных множеств значений всех характеристик этих систем, по существу, приходят к модели автомата, которая тем самым обретает свойство универсальности и потому имеет особую значимость в кибернетике.

Изучение свойств автоматов стало основным направлением в работе коллектива ученых факультета, которую образовали первый из авторов и его ученики. Были выделены и изучены основные среды, типы автоматов и виды поведения автоматов в средах, исследованы проблемы выразимости и полноты для автоматов, созданы методы оптимального синтеза помехоустойчивых автоматов, исследованы моделирующие возможности бесконечных автоматов, называемых клеточными автоматами и др.

Эти результаты вместе с достижениями Э. Мура, С. Клини, Д. Мак-Ноттона и др. сегодня составляют классическое содержание теории автоматов.

Одним из главных направлений этой теории является изучение поведения автоматов в средах. Идеология этого направления смыкается с тем, что изучается в теории интеллектуальных систем.

Так, автоматный анализ геометрических сред связан с распознаванием образов; анализ языковых сред — с языками, логическим выводом и решением задач; анализ смешанных языково-геометрических сред — с коллективным поведением и приняти-

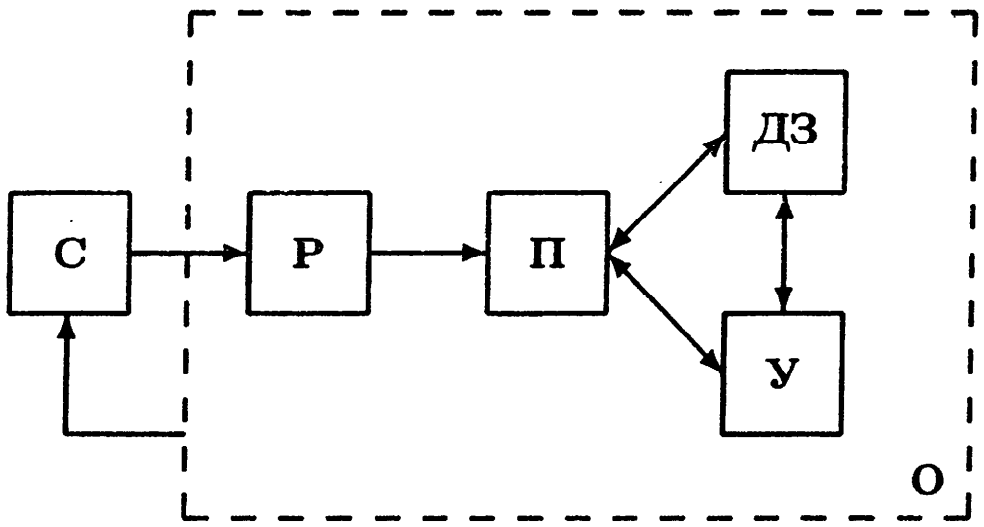

Рис. 1: Интеллектуальная система.

ем решений; вопросы синтеза автоматов с заданным поведением сопряжены с разработкой внутренних для автоматов архитектур баз данных и знаний, быстрого поиска информации и т.д.

При разработке теории автоматов первый из авторов и его ученики постоянно расширяли сферу интерпретаций автоматов, используя модели и соответствующие задачи из разных областей таких, как математика, физика, биология, психология, социология, техника и др. Это обогащало теорию автоматов и расширяло сферу ее применимости.

Как сопряженная с моделью автомата рассматривается и интеллектуальная система.

Очертим в достаточно общем виде распространенный вариант интеллектуальной системы, которую считаем системой типа Тьюринга, и изображаем, как на рисунке 1.

Имеется объект **О**, помещенный в среду **С**, с которой у него имеется двусторонняя связь. Он может воспринимать информацию, поступающую из среды, и влиять на нее, что изображено соответствующим направлением стрелок.

Входная информация из **С** поступает в **О** на блок распознавания **Р**, оттуда она направляется в блок оперативной памяти **П**, где подвергается анализу. При этом анализе используется блок **ДЗ** базы данных и знаний, играющий роль долгосрочной памяти, а сам процесс анализа регулируется управляющим бло-

ком У, который учитывает группу параметров, описывающих как внутренние характеристики объекта, так и состояние среды.

Базы данных и знаний вместе с блоком управления образуют "мозговой центр" системы. От достаточности заложенной в них информации и эффективности внутренних операторов зависят ее имитационные возможности.

Функционирование объекта в среде осуществляется во времени пошагово и оценивается серией внутренних и в общем случае внешних функционалов.

Последовательность значений этих функционалов считается характеризацией взаимодействия объекта и среды. По ней осуществляется оценка "разумности" поведения объекта, включая, в частности, заключение о том, сумел ли объект решить заданную задачу.

Конкретные интерпретации компонент, составляющих систему, приводят к конкретным видам интеллектуальных систем.

Примером такой системы является решатель математических задач. Он имеет в качестве среды класс задач по элементарной алгебре, тригонометрии и началам анализа. Процесс его работы составляет поиск решения предложенной задачи, а результатом этой работы являются ход решения задачи и ответ, если таковые достижимы решателем, и отказ от решения, если последнее невозможно для решателя.

Его базы данных и знаний включают список стандартных приемов тождественных преобразований выражений алгебры и тригонометрии, основных теорем из этих разделов, а также логических операций вывода.

Самым сложным здесь является блок управления, принцип работы которого состоит в эвристической оптимальности извлечения приемов из баз данных и знаний, обеспечивающий в определенном смысле градиентность последовательности примененных приемов, что существенно сокращает перебор вариантов вывода.

В этом блоке реализуется новая идея, позволившая обойти неэффективные попытки использовать для подобных целей логико-аксиоматический подход; упомяним в этой связи "General Problem Solver", "Mathematika" и др.

Эта интеллектуальная система, созданная третьим из авторов, показала высокую эффективность, справляясь за секунды с большинством задач из известных учебников.

Решатель демонстрировался на международной выставке компьютеров и программных продуктов в г. Ганновере (ФРГ), на российских и международных конференциях и семинарах.

Список же конкретных интеллектуальных систем сегодня очень широк.

В соответствии со схемой интеллектуальной системы, изображенной на рисунке 1, в курсе будут рассмотрены 3 основные составляющие интеллектуальной системы. Первая глава, посвященная распознаванию образов, связана с блоком распознавания **Р**. Вторая глава посвящена теории хранения и поиска информации и связана в блоком **ДЗ** — баз данных и знаний. И наконец, последняя глава курса, связанная с блоком управления **У**, посвящена математической логике.

Предлагаемое изложение осуществлено авторами на основе курса лекций, читаемых ими на протяжении ряда лет студентам, специализирующимся по кафедре математической теории интеллектуальных систем (МаТИС) механико-математического факультета МГУ им. М.В.Ломоносова. При курсе работают спецсеминары, рассчитанные на студентов третьего курса кафедры МаТИС. Руководство этими спецсеминарами в разные годы осуществляли сотрудники кафедры МаТИС Алексеев Д.В., Алисейчик П.А., Мазуренко И.Л., Пантелеев П.А., Холоденко А.Б.

Нам приятно поблагодарить весь коллектив кафедры МаТИС, чье постоянное внимание, доброжелательная критика и советы способствовали появлению этой книги. Авторы выражают особую благодарность В.А.Носову, А.С.Строгалову и А.А.Часовских, которые принимали активное участие в разработке курса, и Алексееву Д.В., Алисейчику П.А., Мазуренко И.Л., Пантелееву П.А. и Холоденко А.Б. за помощь в составлении упражнений.

Глава 1

Распознавание образов

Образное восприятие мира — одно из свойств мозга, позволяющее правильно воспринимать информацию о внешнем мире. При этом мы всегда производим классификацию воспринимаемых ощущений, т. е. разбиваем их на группы похожих, но не тождественных образов. Образы обладают тем свойством, что ознакомление с некоторым конечным числом их реализаций дает возможность узнавать всех остальных представителей образа. Более того, разные люди, обучающиеся на различном материале наблюдений, одинаково распознают одни и те же объекты.

Процессу распознавания образов всегда предшествует процесс обучения, во время которого мы знакомимся с некоторым числом примеров, зная наперед об их принадлежности к каким-то образам. У человека процесс обучения заканчивается успешно. Все дети в школе могут научиться различать буквы. Следовательно, образы — это объективные характеристики внешнего мира, а не произвольные наборы изображений. Эту объективную характеристику образов и стремятся выявлять с помощью математических методов. Исследователи создают аппаратные и программные распознающие системы и называют распознаванием информационный процесс, реализуемый распознающей системой. На вход системы подается информация о предъявляемых объектах, на выходе системы отображается информация

о классах, к которым отнесены распознаваемые объекты.

Нам хотелось бы заменить человека, решающего эту задачу автоматической распознающей системой. Тогда круг задач, которые будут решаться с помощью распознающих систем, будет чрезвычайно широк. Сюда можно отнести задачи распознавания зрительных и слуховых образов, задачи выбора целесообразных действий руководителем предприятия, выбора оптимального управления технологическими, экономическими, транспортными или военными операциями.

Задача распознавания образов состоит в следующем. Дано множество M объектов, относительно которых производится распознавание. Известно, что множество M представимо в виде объединения подмножеств $K_1, \ldots, K_m$, называемых обычно классами. Задана частичная информация о классах $K_1, \ldots, K_m$, например, в виде "типичных" представителей классов, или в виде параметрического описания, где значения параметров неизвестны. Дано описание некоторого объекта из M, о котором, вообще говоря, неизвестно — к какому из классов он принадлежит. Необходимо определить, к какому из классов относится объект.

Основную рекомендуемую литературу к данной главе составляют [6, 8, 9, 25, 35].

1.1 Моделирование зрительного восприятия

В данном разделе предполагается, что объект представляет собой конечное множество точек на плоскости. Каждый из классов K_i, $i = 1, 2, \ldots, m$ определяется как множество всех объектов, полученных аффинными преобразованиями к некоторому эталонному объекту S_i, причем эталонные объекты различных классов аффинно к друг другу не сводимы. Множество M всех объектов определяется как объединение классов $K_1, \ldots, K_m$, т.е. в данном случае известно точное описание как множества M, так и классов $K_1, \ldots, K_m$. Надо для произвольно взятого объекта из M установить, к какому классу он относится.

1.1.1 Кодирование изображений

Поместим перед глазом некоторый объект. На сетчатке сформируется совокупность возбужденных рецепторов (см. рис. 1.1). Эта совокупность — и ничего более — есть тот первичный материал, который будет анализироваться мозгом в процессе распознавания. Можно сказать, что для последующих этапов распознавания эта совокупность возбужденных рецепторов и есть объект.

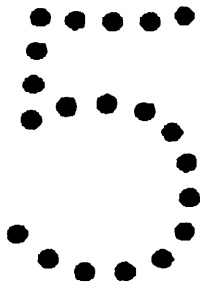

Рис. 1.1:

Если объект перед глазом заменить на другой, то совокупность возбужденных рецепторов изменится. Это и есть основа для того, чтобы в процессе последующего распознавания констатировать изменение объекта. Есть, однако, ситуации, когда объект как таковой не меняется, а меняется только его положение. Пример такой ситуации — сдвиг объекта перед глазом, что приводит к параллельному переносу изображения на сетчатке. При этом изображение на сетчатке осталось, в сущности, прежним, изменилось лишь его местоположение, однако первичный материал — совокупность возбужденных рецепторов — может, очевидно, существенно поменяться.

Вопрос: как осуществляется распознавание таких ситуаций?

В качестве простых преобразований изображений на сетчатке мы будем рассматривать сдвиг (параллельный перенос) изображения, поворот, подобное преобразование, сжатие, растяжение и, разумеется, все их сочетания (т.е. аффинные преобразования). Этот набор преобразований взят не произвольно, а потому, что эти преобразования получаются при определенных перемещениях объекта перед глазом.

Мы рассмотрим далее разные варианты преобразований изоб-

ражения на сетчатке и для каждого такого случая будем строить кодировку изображения, инвариантную к рассматриваемым преобразованиям. Распознавание будет основываться на этих инвариантах.

Пусть A — множество из n различных точек плоскости, т.е. $|A| = n$. Это множество в дальнейшем будем называть *изображением.*

Взаимно однозначную функцию $M_A : A \to \{1, 2, \ldots, n\}$ будем называть *функцией нумерации* изображения A. Поскольку M_A взаимно однозначная функция, то определена функция M_A^{-1}, которая номеру из $\{1, 2, \ldots, n\}$ сопоставляет точку из A.

Если a точка плоскости, то через $X(a)$, $Y(a)$ обозначим соответственно абсциссу и ординату точки a.

Вектор

$$K_{M_A} = (X(M_A^{-1}(1)), Y(M_A^{-1}(1)), \ldots, X(M_A^{-1}(n)), Y(M_A^{-1}(n)))$$

будем называть *вектором координат изображения A при нумерации M_A.*

Пусть $T : \mathbb{R}^{2n} \to \mathbb{R}^t$ — некоторая t-мерная вектор функция, называемая *кодирующей.* Пару $\langle M_A, T_A \rangle$, где $T_A = T(K_{M_A})$, будем называть *кодом изображения A при нумерации M_A для кодирующей функции T.* Число t, равное длине вектора T_A и размерности кодирующей функции T, назовем *сложностью кода* $\langle M_A, T_A \rangle$.

Два изображения A и B назовем *эквивалентными относительно кодирующей функции T,* если $|A| = |B|$ и существуют такие функции нумерации M_A и M_B, что $T(K_{M_A}) = T(K_{M_B})$.

Пусть Г — некоторое множество геометрических преобразований плоскости. В дальнейшем мы будем рассматривать следующие множества:

- Γ_1 — множество, состоящее из одного тождественного преобразования;
- Γ_2 — множество преобразований, получающихся с помощью любых комбинаций сдвига, поворота и преобразований симметрии;
- Γ_3 — множество преобразований подобия, т.е. преобразований, получающихся с помощью любых комбинаций

сдвига, поворота и преобразований симметрии и изменения в размерах (с сохранением подобия);

- Γ_4 — множество аффинных преобразований плоскости.

Скажем, что два изображения *эквивалентны относительно множества преобразований* Г, если одно может быть получено из другого с помощью преобразований из Г.

Скажем, что кодирующая функция T *правильная для множества геометрических преобразований* Г, если два изображения эквивалентны относительно кодирующей функции T тогда и только тогда, когда они эквивалентны относительно множества преобразований Г.

Таким образом, с помощью кода с правильной для множества геометрических преобразований Г кодирующей функцией может решаться задача распознавания эквивалентности изображений относительно этого множества преобразований.

Случай 1. Этот случай соответствует, в сущности, представлению того, что выше было названо первичным материалом. Полагаем, что перед глазом — плоское изображение, составленное из конечного числа точек (пример на рис. 2). Точки, из которых состоит изображение, будем считать, невелики по размерам и потому на проекцию каждой точки на сетчатку приходится один возбужденный рецептор. Изображение представляется совокупностью конкретных возбужденных рецепторов. Положение каждого такого рецептора известно. Его можно задавать по-разному. Будем полагать, что на сетчатке введена декартова система координат и положение каждого рецептора определяется его координатами. Отметим, что можно было бы взять и другие системы координат — суть дела это меняет незначительно.

Пусть дано изображение A. Перенумеруем его точки некоторым образом так, чтобы номера были попарно различны, т.е. зададим функцию нумерации M_A. Кодом K^1 изображения A (обозначение: K^1_A) назовем пару множеств $< M_A, T_A >$. Здесь T_A множество координат $(x, y)_n$ точек с указанием их номера, то есть, например, $(5, 3)_n$ означает, что у точки с номером n координатами являются пара $(5, 3)$. То сеть для кода K^1_A в каче-

стве кодирующей функции выступает функция $T : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ такая, что $T_A = T(K_{M_A}) = K_{M_A}$.

Перенумеровав по иному множество точек изображения A, получим другую пару $< M_A, T_A >$. Будем, однако, рассматривать кодировку с точностью до перенумерации и называть изображения, отличающиеся только нумерацией точек, эквивалентными (в смысле случая 1). Таким образом, если имеются изображение A с кодом $< M_A, T_A >$ и B с кодом $< M_B, T_B >$, то назовем A и B эквивалентными (в смысле случая 1), если существует такая нумерация точек изображения A, при которой его код есть $< \tilde{M}_A, \tilde{T}_A >$, и такая нумерация для B, при которой его код есть $< \tilde{M}_B, \tilde{T}_B >$, и при этом $\tilde{T}_A = \tilde{T}_B$.

Легко видеть, что код K_A^1 является правильным для множества преобразований Γ_1.

Случай 2. Пусть изображение перед глазом некоторым образом смещается и поворачивается. Пусть при этом плоскости изображения и сетчатки остаются параллельными (или, если они не параллельны, то угол между этими плоскостями остается неизменным) и расстояние между плоскостями не меняется. Очевидно, что этот случай можно трактовать и так, что изображение перед глазом неподвижно, а смещается и поворачивается сам глаз. Преобразования изображения на сетчатке в этом случае сводятся, очевидно, к тому, что оно смещается и поворачивается — других изменений нет. Ясно, что при этом меняется и координатный код изображения, поскольку меняются и координаты составляющих изображения точек.

При сдвиге и повороте меняется положение изображения на сетчатке, другими словами, положение изображения по отношению к осям системы координат. Взаиморасположение же точек изображения не меняется, что означает сохранение расстояний между ними. Именно сохранение расстояний между точками изображения при сдвиге и повороте положим в основу нового кода K_A^2. В коде $< M_A, T_A >$, как и ранее, M_A есть функция нумерации множества A. Множество T_A составляют все числа $r(q, m)$ (q и m — номера точек из $\{1, 2, \ldots, |A|\}$), являющиеся расстояниями между точками изображения, с указанием того, расстоянием между какими точками является данное число. То есть, если $|A| = n$, то кодирующая функция T такая,

что $T : \mathbb{R}^{2n} \to \mathbb{R}^t$, где $t = n(n-1)/2$ — число различных пар (q, m), $q < m$, и если нумеровать компоненты вектор-функции T парами (q, m), то (q, m)-я компонента равна

$$T_{q,m}(K_{M_A}) =$$

$$= \sqrt{(X(M_A^{-1}(q) - X(M_A^{-1}(m))^2 + (Y(M_A^{-1}(q) - Y(M_A^{-1}(m))^2}.$$

Аналогично тому, как это сделано выше, определим одинаковые с точностью до перенумерации точек коды. Изображения с такими кодами будем называть эквивалентными (в смысле случая 2). Можно показать, что два изображения эквивалентны тогда и только тогда, когда одно может быть получено из другого комбинацией сдвига, поворота и преобразования симметрии, т.е. код K_A^2 — правильный для множества преобразований Γ_2. Это можно рассматривать как частный случай утверждения, доказанного в [24].

Рассмотрим теперь, как соотносятся коды K_A^1 и K_A^2 и чем объясняется различие их свойств. Задавая изображение совокупностью координат всех его точек (код K_A^1), мы, в сущности, задаем нечто большее, чем собственно изображение — неявным образом в таком задании присутствует и внешняя по отношению к изображению система отсчета, то есть система координат с ее осями. Эта система в реальности должна быть "привязана" к каким-то дополнительным внешним точкам, например, к краям сетчатки. Действительно, посмотрим на код K_A^1 как на некий вариант кода K_A^2, то есть будем считать, что каждая точка в K_A^1 тоже задается совокупностью расстояний до других точек. В таком случае этими другими точками для произвольной точки a изображения будут точки x_a и y_a на осях координат, являющиеся проекциями точки a на оси. Код K_A^1 как бы предполагает наличие, помимо точек собственного изображения, еще и точек x_a и y_a для каждой точки a. Если присоединить эти точки к изображению, то очевидно, код K_A^2 уже не будет инвариантным к сдвигу и повороту первоначального изображения, поскольку при этом меняются расстояния от точек первоначального изображения до добавленных точек.

Основываясь на коде K_A^2, нетрудно осуществить распознавание сдвинутых, повернутых и симметрично преобразованных изображений. Пусть изображения A и B заданы координата-

ми своих точек, то есть заданы кодами K_A^1 и K_B^1. Мы предполагаем, что B — это сдвинутое, повернутое и симметрично преобразованное изображение A и намереваемся проверить это предположение. Определим коды K_A^2 и K_B^2 — это будут, соответственно, $< M_A, T_A >$ и $< M_B, T_B >$. Коды K_A^2 и K_B^2 очевидным образом строятся по кодам K_A^1 и K_B^1. Если изображения A и B эквивалентны, то коды K_A^2 и K_B^2 должны быть одинаковы с точностью до перенумерации точек. Остается только проверить это для кодов K_A^2 и K_B^2. Эта процедура может быть определена по-разному и основывается на очевидных свойствах кодов эквивалентных изображений. Так, мощности множеств T_A и T_B равны, если изображения A и B эквивалентны. Выделим, далее, в T_A подмножества, состоящие из равных по величине элементов и расположим эти подмножества в порядке возрастания этой величины, то есть $T_A = \{t_1^A, ..., t_m^A\}$. Очевидно, что $t_i^A \cap t_j^A = \emptyset$ для $i, j = 1, ..., m$ и $i \neq j$ $\cup_{i=1}^m t_i^A = T_A$, $\sum_{i=1}^m |t_i^A| = |T_A|$. Здесь $|t_i^A|$ и $|T_A|$ — мощности соответствующих множеств. Такое же представление в виде подмножеств и с такими же свойствами должно иметь место и для изображения B, то есть $T_B = \{t_1^B, ..., t_m^B\}$. Ясно, что при этом $|t_i^A| = |T_i^B|$ для всех $i = 1, ..., m$.

Очевидным образом возникающая "привязка" множеств t_i^A и t_i^B друг к другу дает возможность определить соответствие между тройкой точек в A и тройкой точек в B — соответствие между остальными точками определяется этим однозначно.

Отметим, что для изображения A, состоящего из n точек, сложность K_A^1 равна $2n$, а сложность K_A^2 равна $n(n-1)/2$.

Вместе с тем код K_A^2 явно избыточен. Это видно, например, из того, что если зафиксировать на плоскости положение трех точек изображения, то для определения положения остальных точек все элементы множества T_A, очевидно, не потребуются.

Случай 3. Пусть изображение перед глазом приближается или удаляется с сохранением параллельными плоскости изображения и плоскости сетчатки (или, если они не параллельны, с сохранением неизменным угла между ними). Это же можно трактовать и так, что изображение перед глазом неподвижно, а приближается сам глаз. В обоих случаях изображение на сетчатке увеличивается или уменьшается в размерах с сохранением подобия. Можно отметить, что такие же изменения на

сетчатке возникнут и в том случае, когда изображение не приближается и не удаляется, а увеличивается или уменьшается (с сохранением подобия). Если не располагать дополнительной информацией, то отличить эти два вида преобразования изображения перед глазом — изменение в размерах и изменение расстояния до глаза — нельзя, что и наблюдается в зрительных иллюзиях.

Коды K_A^2 K_B^2 изображения A при его приближении или удалении от глаза очевидно меняются, поскольку меняются и координаты точек его проекции на сетчатку, и расстояния между точками этой проекции. Соотносительные же размеры частей изображения при увеличении или уменьшении с сохранением подобия не меняются. Это и положим в основу кода K_A^3. Так же, как и в предыдущих случаях, перенумеруем множество точек изображения A и обозначим функцию нумерации через M_A. Далее зададим множество T_A численных значений отношений вида $r(m,l)/r(p,q)$, где $r(m,l)$ и $r(p,q)$ — расстояния между точками с номерами соответственно m и l, p и q, т.е.

$$r(p,q) =$$

$$= \sqrt{(X(M_A^{-1}(p) - X(M_A^{-1}(q))^2 + (Y(M_A^{-1}(p) - Y(M_A^{-1}(q))^2}.$$

Здесь m, l, p, q — номера из $\{1, 2, \ldots, |A|\}$, $m < l$, $p < q$. Для каждого числа из T_A полагаем известной соответствующую ему четверку номеров m, n, p, q. Код K_A^3 есть пара $< M_A, T_A >$. Аналогично тому, как это рассматривалось для предыдущих двух случаев, можно определить одинаковые с точностью до перенумерации точек коды. Изображения с такими кодами будем называть эквивалентными (в смысле случая 3). Можно показать, что два изображения эквивалентны в том и только в том случае, если на плоскости одно может быть получено из другого сдвигом, поворотом, изменением в размерах (с сохранением подобия), преобразованием симметрии или их комбинацией (то есть подобными преобразованиями), т.е. код K_A^3 — правильный для множества преобразований Γ_3. Соответствующее утверждение доказано в [33].

Рассмотрим теперь, как соотносятся коды K_A^2 и K_A^3 и чем объясняется различие их свойств. По коду K_A^2 можно, очевидно, построить код K_A^3. Обратное неверно. Код K_A^2 определяется

таким образом, чтобы в описании изображения не участвовала бы "внешняя" по отношению к изображению система координат. Однако нечто от этой внешней системы в определении кода K_A^2 осталось, а именно — единица измерения расстояний между точками изображения. Она, эта единица, предполагается для кода K_A^2 априори заданной. Когда же мы берем отношение $r(m,n)/r(p,q)$ (в коде K_A^3), эта единица измерения устраняется. Действительно, величина отношения $r(m,n)/r(p,q)$ будет одной и той же вне зависимости от того, в каких единицах измеряются расстояния $r(m,n)$ и $r(p,q)$. Отсюда и возникает возможность посредством K_A^3 описывать изображение безотносительно к его размерам.

Можно и несколько по иному интерпретировать код K_A^3. Возьмем в качестве единицы измерения расстояние между какой-либо парой точек на самом изображении. Пусть это будут точки с номерами p и q. Теперь для того, чтобы для произвольных точек m и n изображения получить расстояние между ними, выраженное в единицах, являющихся расстоянием между точками p и q, нужно, очевидно, проделать следующее. Взять произвольную единицу измерения и с ее помощью получить расстояния $r(m,n)$ и $r(p,q)$. Затем разделить $r(m,n)$ на $r(p,q)$, то есть получить $r(m,n)/r(p,q)$. Это и будет искомым числом. Если теперь поочередно считать единицей измерения расстояние между каждой парой точек в изображении, то мы и придем ко множеству T_A для кода K_A^3.

При восстановлении по коду достаточно задать конкретной величину расстояния между любой парой точек изображения. Тогда по элементам множества T_A можно получить значения расстояний для всех других пар точек (то есть, в сущности, получить код K_A^2), и затем построить изображение.

Процедуру распознавания изображений, полученных преобразованиями подобия, можно провести аналогично процедуре, описанной для случая 2.

Множество T_A имеет $(n(n-1)/2)(n(n-1)/2-1)$ элементов.

Вместе с тем код K_A^3, очевидно, избыточен.

Случай 4 (основной). Проведем в плоскости изображения перед глазом прямую (пересекающую изображение). Повернем изображение вокруг прямой на некоторый угол. Изображение

на сетчатке сожмется в направлении, перпендикулярном прямой (пример на рис. 1.2). Этот случай можно трактовать и так, что изображение перед глазом неподвижно, а поворачивается на некоторый угол плоскость сетчатки. Наконец, изображение перед глазом может действительно сжиматься по некоторому направлению. Во всех этих случаях преобразование изображения на сетчатке сведется к сжатию.

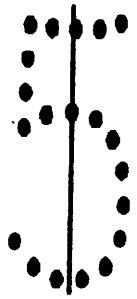

Рис. 1.2:

Если в исходном положении плоскость изображения непараллельна плоскости сетчатки и затем поворачивается, делаясь параллельной, то, очевидно, изображение на сетчатке при этом будет растягиваться в направлении, перпендикулярном оси поворота.

Еще раз подчеркнем, что только к сжатию или к растяжению преобразование изображения на сетчатке сводится в том случае, когда несущественна разница в расстояниях до глаза от разных частей изображения, возникающая после поворота вокруг оси. В целом эта особенность возникает как часть более широкой проблемы, отмеченной в исследованиях по машинному зрению (см., например, [58, 60]). Проекции тела на сетчатку рассматриваются обычно как параллельные проекции (с добавлением возможности преобразования подобия). Глазом же, можно считать с большим основанием, осуществляется центральная проекция. Если, однако, расстояние до объекта велико по сравнению с размерами самого объекта, то различия между центральной и параллельной проекциями невелики и ими можно пренебречь. Мы считаем, что находимся в границах применимости именно этого случая.

Коды K_A^1, K_A^2, K_A^3 изображения при сжатии и растяжении,

очевидно, меняются. Так, например, код K_A^3 меняется потому, что меняются соотносительные размеры частей изображения (не сохраняется подобие).

Назовем двумерным изображением конечное множество точек на плоскости. Перенумеруем некоторым образом точки изображения A, т.е. зададим функцию нумерации M_A. Пусть S_{mnu} и S_{kps} — площади треугольников с вершинами в тройках точек с номерами m, n, u и k, p, s и пусть $\rho_{mnu,kps} = S_{mnu}/S_{kps}$. Полагаем, что порядок номеров в тройках не важен, сами тройки различны и при $S_{kps} = 0$ значение $\rho_{mnu,kps}$ не определено. Множество индексированных чисел $\rho_{mnu,kps}$ для всех таких пар троек обозначим через T_A. Код K_A^4 изображения A — пара $< M_A, T_A >$. Изображения, все точки которых расположены на одной прямой, не рассматриваем, поскольку код для них не определен. Как и ранее, изображения A и B назовем эквивалентными в смысле случая 4 (далее — просто эквивалентными), если существуют такие нумерации M_A и M_B, что коды T_A и T_B равны. Ясно, что эквивалентность изображений содержательно означает одинаковость их кодов с точностью до перенумерации точек.

1.1.2 Теорема Козлова об аффинной эквивалентности изображений

Два изображения называем *аффинно эквивалентными* (*а-эквивалентными*), если они переводимы друг в друга аффинными преобразованиями.

Лемма 1. *Если два изображения а-эквивалентны, то они эквивалентны.*

Доказательство. Нужно показать, в сущности, что при аффинных преобразованиях численные значения элементов их множества T кода изображения не меняются. Каждый элемент в T определяется отношением площадей некоторых треугольников. При преобразованиях сдвига, поворота и симметрии эти площади не меняются, а значит не меняется и их отношение. Пусть теперь происходит сжатие (или растяжение) изображения по осям x и y с коэффициентами k_x и k_y. Известно [22], что

в этом случае площадь S' фигуры (или любой ее части) будет после преобразования равна Sk_xk_y (здесь S — площадь фигуры до преобразования). Отсюда следует, что если $\rho'_{muv,lpq}$ — соответствующий $\rho_{muv,lpq}$ элемент множества множества T после преобразования, то $S_{muv}k_xk_y/S_{lpq}k_xk_y = S_{muv}/S_{lpq} = \rho_{muv,lpq}$. Лемма доказана. □

Пусть изображение B состоит из точек, являющихся подмножеством точек изображения A с кодом $< M_A, T_A >$. Обозначим через M_B множество номеров тех и только тех точек из A, которые входят в изображение B. Соответственно через T_B обозначим множество всех тех элементов $\rho'_{muv,lpq}$ из T_A, для которых $m, u, v, l, p, q \in M_B$. Изображение B с кодом $< M_B, T_B >$ назовем частью изображения A. Любое изображение B', a-эквивалентное изображению B, назовем подизображением изображения A.

Будем называть изображение из четырех точек четырехточечником. Пусть в четырехточечнике A из точек a, b, c, d точки a, b и c расположены на одной прямой. Тогда $S_{abc} = 0$ и те элементы из T_A, для которых в соответствующем отношении S_{abc} входит в знаменатель, не определены, а те, для которых S'_{abc} входит в числитель (при не равном нулю знаменателе), равны нулю. Очевидно, можно утверждать обратное: если код $< M_A, T_A >$ обладает указанными свойствами, то в четырехточечнике A точки a, b и c расположены на одной прямой. В изображении A общего вида точки a, b, c, входящие в изображение, расположены на одной прямой в том и только в том случае, если для любой части изображения A, являющейся четырехточечником и включающей точки a, b, c, выполняется указанное свойство, определяющее расположенность точек a, b, c на одной прямой. В изображении A точки $a_1, ..., a_n (n \geq 3)$ расположены на одной прямой в том и только в том случае, когда любые три точки из $a_1, ..., a_n$ расположены на одной прямой.

Изображение назовем *плоским*, если все его точки не лежат на одной или двух параллельных прямых.

Лемма 2. *Если по коду плоского изображения выбраны какие-либо три его точки, не лежащие на одной прямой, и задано произвольное (но не на одной прямой) положение этих точек*

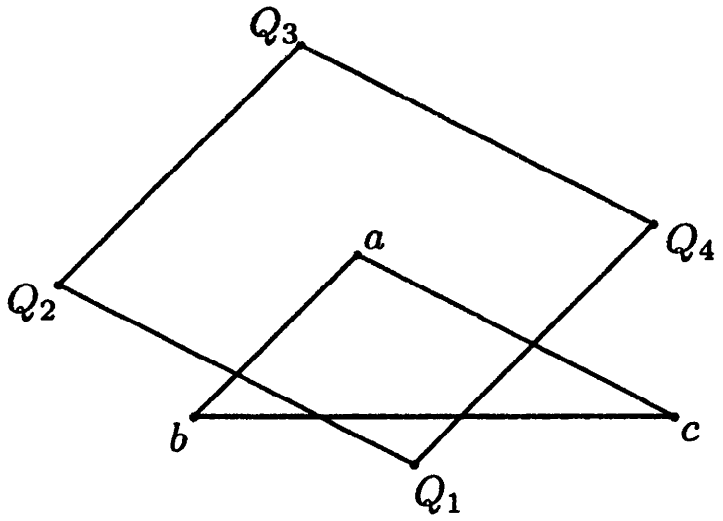

Рис. 1.3:

на плоскости, то положение остальных точек изображения определяется по коду однозначно.

Доказательство. Пусть наш код $< M_A, T_A >$ является кодом некоторого плоского изображения A. Пусть m, p, q — три выбранные точки, не лежащие на одной прямой, и a', b', c' их исходное положение. Пусть s — четвертая точка изображения, которая вместе точками m, p, q образует плоское изображение, т.е. если d' — исходное положение точки s, то точки a', b', c' и d' не лежат на одной или двух параллельных прямых.

Предположим, что задано новое положение a, b, c точек m, p, q. Для определения нового положения точки s воспользуемся процедурой (стандартной) построения по коду четырехточечника четвертой точки при известных положениях остальных трех. Полагаем, что точки a, b, c разные, лежат не на одной прямой, в остальном же их положение можно задавать произвольным образом. Точку, положение которой определяется, обозначим через x.

Поскольку площадь треугольника abc известна, то, используя $S_{abx}/S_{abc}, S_{acx}/S_{abc}, S_{bcx}/S_{abc}$, находим площади S_{abx}, S_{acx}, S_{bcx}. Для треугольника abc длины его сторон известны, поэтому можно определить длины высот h_{ab}, h_{ac}, h_{bc}, опущенных из точки x на стороны соответственно ab, ac и bc. (Одна из этих высот может оказаться нулевой по длине, две быть нулевыми, очевидно, не могут). Таким образом, задача сводится к тому, чтобы по заданному треугольнику abc построить точку x та-

кую, что высоты из этой точки на стороны ab, ac и bc треугольника есть соответственно h_{ab}, h_{ac} и h_{bc}. Проводим две прямые, параллельные стороне ab по разные стороны и отстоящие от нее на величину h_{ab}, и две прямые, параллельные стороне ac по разные ее стороны и отстоящие от нее на величину h_{ac}. Возникает параллелограмм, вершины которого обозначены на рис. 4 как Q_1, Q_2, Q_3 и Q_4. По построению, точка x может располагаться только на вершинах этого параллелограмма. Для окончательного определения положения точки x нужно провести две прямые, параллельные оставшейся стороне треугольника abc (стороне bc на рис. 1.3) и отстоящие от нее на величину h_{bc}. Пересечение этих прямых с вершинами параллелограмма даст возможные положения точки x.

Покажем, что одно такое пересечение гарантированно существует.

Рассмотрим аффинное преобразование, которое переводит точки a', b', c' в точки a, b и c. Пусть это аффинное преобразование переводит точку d' в точку d. Согласно лемме 1 это аффинное преобразование не меняет кода и, значит, что точка d является гарантированно существующим положением, точки x. Отметим, что так как аффинное преобразование переводит параллельные прямые в параллельные, то точки a, b, c и d не лежат на одной или двух параллельных прямых.

Остается вопрос, единственным ли может получиться положение восстанавливаемой точки.

Предположим, что для точки x возможны два разных положения x' и x'', одно их которых гарантированная точка d.

1. Предположим, что точки x' и x'' совпадают с Q_1 и Q_2, которые поэтому должны быть равноудалены от стороны bc. Следовательно, отрезок bc должен делить отрезок Q_1Q_2 пополам, то есть точка b должна лежать на отрезке Q_1Q_2. В этом случае a, b, c, x' и x'' расположены на двух параллельных отрезках: отрезке ac и отрезке Q_1Q_2, и, значит, на двух параллельных отрезках лежат точки a, b, c и d — пришли к противоречию.

2. Аналогичным образом приходим к противоречию в предположении, что x' и x'' совпадают с Q_1 и Q_4.

Если x' и x'' совпадают с Q_1 и Q_3, то отрезок Q_1Q_3 должен делиться стороной bc пополам. Однако серединой диагона-

ли Q_1Q_3 должен делиться стороной bc пополам. Однако серединой диагонали Q_1Q_3 параллелограмма является, по построению, точка a.

Если x' и x'' совпадают с Q_2 и Q_3, то отрезок Q_2Q_3 должен быть либо параллелен отрезку bc, либо делиться этим отрезком или его продолжением пополам. И то, и другое противоречиво по построению. Аналогичное имеет место и при совпадении x' и x'' с Q_3 и Q_4.

3. Пусть, наконец, x' и x'' совпадают с Q_2 и Q_4. Тогда диагональ Q_2Q_4 должна быть параллельной стороне bc, что противоречит тому, что точки a, b, c и d не лежат на двух параллельных прямых.

Тем самым мы доказали, что положение точки x определяется однозначно и совпадает с точкой d при условии, что точки m, p, q и s не лежат на двух параллельных прямых.

Разобьем множество точек изображения A, отличных от точек m, p, q, на два класса A_1 и A_2. В множество A_1 включим точки s такие, что m, p, q и s не лежат на двух параллельных прямых, а класс A_2 составят оставшиеся точки.

Как мы показали выше, положение каждой точки их A_1 определяется однозначно положением точек m, p, q, причем так как изображение A плоское, то множество A_1 — непустое.

Рассмотрим произвольную точку s из множества A_2. Возьмем произвольную точку r из множества A_1, положение которой уже определено. Точки m, p, q, s и r образуют плоское изображение, значит из четырех m, p, q и r можно выбрать три, что они вместе с точкой s образуют плоское изображение, т.е. не лежат на двух параллельных прямых. Тогда по этим трем точка положение точки s определяется однозначно.

Лемма доказана. □

Теорема 1. *Два плоских изображения эквивалентны точно тогда, когда они a-эквивалентны.*

Доказательство. В одну сторону теорема 1 следует из леммы 1.

Покажем теперь, что если плоские изображения A и B эквивалентны, то они a-эквивалентны. Выберем на A три точки

a, b, c, не лежащие на одной прямой. Пусть следующей из определения эквивалентности биекцией точкам a, b, c сопоставляются точки соответственно a', b', c' изображения B. Аффинным преобразованием изображения B совместим его точки a', b', c' с точками соответственно a, b, c изображения A. Поскольку положение остальных точек изображения B, при заданных точках a', b', c' определяется, согласно лемме 2, однозначно, то, следовательно, они совпадут с точками изображения A.

Теорема доказана. □

1.1.3 Распознавание и восстановление объемных изображений

Назовем *трехмерным изображением* A или *телом* конечное множество точек в трехмерном евклидовом пространстве. Занумеруем попарно различными номерами точки изображения A, т.е. зададим функцию нумерации M_A. Пусть V_{mnuv} и V_{kspq} — объемы тетраэдров с вершинами в четверках точек с номерами m, n, u, v и k, s, p, q и пусть $\rho_{mnuv,kpsq} = V_{mnuv}/V_{kspq}$. Полагаем, что порядок в четверках не важен, сами четверки различны и для случая , когда $V_{kspq} = 0$, $\rho_{mnuv,kpsq}$ не определено. Множество индексированных чисел $\rho_{mnuv,kpsq}$ для всех таких пар четверок обозначим через T_A. *Кодом тела* назовем пару $< M_A, T_A >$. Тела A и B назовем *эквивалентными*, если существуют такие функции нумерации M_A и M_B, что в кодовых множествах T_A и T_B элементы, соответствующие одинаковым индексам совпадают.

Тела, все точки которых расположены в одной плоскости, называем двумерными и для них рассматриваемый код не определен.

Тела называем *аффинно эквивалентными* (*а-эквивалентными*), если они переводятся друг в друга аффинными преобразованиями.

Трехмерное изображение назовем *объемным*, если все его точки не лежат в одной плоскости или в двух параллельных плоскостях.

Приведем без доказательства следующее утверждение.

Теорема 2. *Два объемных изображения эквивалентны точно тогда, когда они аффинно эквивалентны.*

Теперь обратимся к вопросу восстановления трехмерного тела по его двумерным проекциям.

Рассмотрим тело T и прямую, называемую *направлением проекции*. Направления проекции назовем *разными*, если они не параллельны. Проведем через каждую точку тела T прямые, параллельные направлению проекции α и называемые *лучами*. Полагаем α таким, что на каждом луче находится только одна точка тела. Назовем плоскость, пересекающую лучи, *плоскостью проекции*, изображение, образованное точками пересечения лучей с плоскостью проекции — *проекцией тела* (на данную плоскость по данному направлению). Рассматриваем проекции тела T по разным направлениям и на разные плоскости. Взаимно однозначное соответствие между точками двух изображений назовем их *разметкой*. Соответствующие друг другу точки будем обозначать одной буквой (с разными индексами). Ясно, что описанным выше устанавливается взаимно однозначное соответствие между точками тела T и точками проекций S_i $(i = 1, 2, \ldots)$. Если a — точка тела T, то точку проекции S_i, лежащую с ней на одном луче, обозначим через a_i и будем называть *проекцией точки* a. Это устанавливает и взаимно однозначное соответствие между точками проекций S_i и S_j: соответствующие друг другу точки являются проекциями одной и той же точки тела T. Размеченные изображения A и B назовем *a'-эквивалентными*, если можно перевести их одно в другое аффинными преобразованиями так, что совместятся соответствующие друг другу точки (обозначение: $A \approx B$). В противном случае A и B назовем *a'-разными* (обозначение: $A \not\approx B$). Часть изображения A, состоящую из его точек $a, b, \ldots, v$ будем обозначать как $A(a, b, \ldots, v)$. Три точки проекции, не лежащие на одной прямой, назовем *гранью*, определяемую этими точками плоскость — *плоскостью грани*. Три точки проекции, не лежащие на одной прямой, назовем *треугольником*. Оговорим обозначения для специального случая. Изображение A, состоящее из точек $a_i, b_i, c_i, \ldots u_i$ и прямой L_i, будем обозначать через $A(a_i, b_i, c_i, \ldots u_i, L_i)$. Аналогично, изображение B, состоящее из точек $a_j, b_j, c_j, \ldots u_j$ и прямой L_j, будем обозна-

чать через $B(a_j, b_j, c_j, \ldots u_j, L_j)$. Назовем изображения A и B *a'-эквивалентными*, если их можно перевести одно в другое аффинными преобразованиями так, что точки $a_i, b_i, c_i, \ldots u_i$ совместятся с точками соответственно $a_j, b_j, c_j, \ldots u_j$ и прямая L_i совместится с прямой L_j.

Имея тело T, заданное направление проекции α и меняя плоскости проекции, можно получить некоторое множество $\{S\}$ проекций. Все проекции из $\{S\}$ будут попарно a'-эквивалентными. С другой стороны, тело T — не единственное, проецированием которого можно получить множество $\{S\}$ проекций. Таким будет, например, тело T', полученное заменой каждой точки тела T, находящейся на луче α_x проецирования, на какую-то другую точку x' на том же луче. В частном и вырожденном случае все точки тела T' могут находиться в одной плоскости. Из этого следует, что имея одну или несколько проекций из множества $\{S\}$, нельзя восстановить тело T. Проекции из $\{S\}$ можно интерпретировать как изображения тела в одном ракурсе. Следовательно, для того, чтобы восстановить тело, нужно иметь более чем одну проекцию, причем в разных ракурсах (то есть при разных направлениях проекции).

Теорема 3. *Если S_1 и S_2 суть a'-разные проекции тела T, то существует процедура, которая по проекциям S_1 и S_2 построит тело T' такое, что тела T и T' — a'-эквивалентны.*

Опишем процедуру, существование которой утверждается в теореме 3, без доказательства ее корректности.

Пусть S_1 и S_2 — a'-разные проекции тела T, $a_1b_1c_1$ на S_1 и $a_2b_2c_2$ на S_2 — треугольники. Будем говорить, что точка $D-1$ на S_1 *лежит в плоскости треугольника* $a_1b_1c_1$, если

$$S_1(a_1, b_1, c_2, d_1) \approx S_2(a_2, b_2, c_2, d_2).$$

В этом случае и d_2 лежит в плоскости треугольника $a_2b_2c_2$. В противном случае говорим, что d_1 *лежит вне плоскости треугольника* $a_1b_1c_1$ (d_2 лежит вне плоскости треугольника $a_2b_2c_2$).

Пусть S_1 и S_2 — a'-разные проекции тела T. Рассмотрим на S_1 и S_2 четверку троек точек $((a_1b_1c_1)(a_2b_2c_2)(c_1e_1d_1)(c_2e_2d_2))$ такую, что $a_1b_1c_1$ и $a_2b_2c_2$ — треугольники, точки d_1 и e_1 лежат вне плоскости треугольника $a_1b_1c_1$ (и, соответственно, точки d_2 и e_2 лежат вне плоскости треугольника $a_2b_2c_2$) и из троек

$c_1e_1d_1$ и $c_2d_2e_2$ хотя бы одна треугольник. Такую четверку назовем *правильной*. Ее точки на S_1 и S_2 являются проекциями точек a, b, c, d, e тела T, причем abc и cde — грани, и плоскости этих граней разные. Обозначим через L линию пересечения этих плоскостей, через L_1 и L_2 — проекцию этой линии на S_1 и $S-2$. Очевидно, что L_1 и L_2 проходят через точки соответственно c_1 и c_2. Для определения положения прямых L_1 и L_2, тем самым, достаточно определить еще по одной точке, лежащих на них. Опишем алгоритм определения по правильной четверке $((a_1b_1c_1)(a_2b_2c_2)(c_1e_1d_1)(c_2d_2e_2))$ линий L_1 и L_2.

1. Положим, что только одна из троек $c_1e_1d_1$ и $c_2d_2e_2$ — треугольник, и пусть, для определенности, это тройка $c_2d_2e_2$. Следовательно, c_1, d_1 и e_1 лежат на одной прямой и, значит, направление проекции α_1 параллельно плоскости грани cde. В этом случае точки c_1, d_1 и e_1 лежат на прямой L_1, что и определяет ее положение.

Прямая L в теле T лежит в плоскости треугольника abc, следовательно $S_1(a_1, b_1, c_1, L_1) \approx S_2(a_2, b_2, c_2, L_2)$. Совместим аффинными преобразованиями изображения S_1 точки a_1, b_1 и c_1 с точками соответственно a_2, b_2 и c_2 на S_2. Прямая L_1 при этом совместится с L_2, что и определяет положение L_2 на S_2.

2. Положим теперь, что тройки $c_1e_1d_1$ и $c_2d_2e_2$ — треугольники.

Если какая-либо из точек a_1 и b_1 лежит в плоскости треугольника $c_1e_1d_1$, то прямая L_1 проходит через эту точку и точку c_1. Положение L_2 тоже определено тем, что она проходит через соответствующие точки на S_2.

Пусть теперь точки a_1 и b_1 лежат вне плоскости треугольника $c_1e_1d_1$.

Если в теле T отрезок de параллелен плоскости грани abc и отрезок ab параллелен плоскости грани cde, то оба отрезка должны быть параллельны прямой L и, следовательно, параллельны между собой. В этом случае отрезки a_1b_1 и d_1e_1 на $S-1$ и a_2b_2 и d_2c_2 на $S-2$ параллельны между собой. Их направления и определяют прямые L_1 и L_2.

Положим теперь, что либо отрезок de не параллелен грани abc, либо отрезок ab не параллелен грани cde. Пусть, для определенности, отрезок de не параллелен грани abc. Обозна-

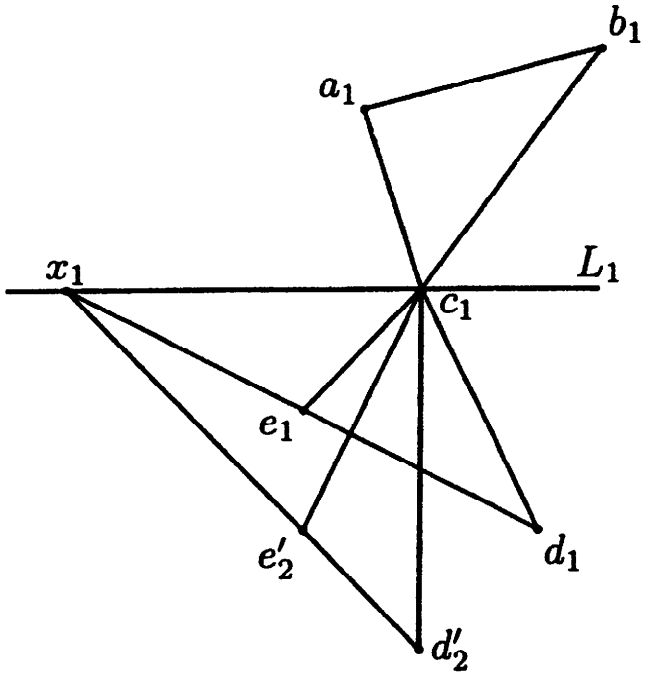

Рис. 1.4:

чим через x точку пересечения отрезка de ил его продолжения с прямой L, и соответственно через x_1 и x_2 — соответствующие точки на S_1 и S_2. Точки a, b, c и x лежат в теле T в одной плоскости, следовательно, $S_1(a_1, b_1, c_1, x_1) \approx S_2(a_2, b_2, c_2, x_2)$. Точки c, d, e и x лежат в теле T в одной плоскости, и, значит, $S_1(c_1, d_1, e_1, x_1) \approx S_2(c_2, d_2, e_2, x_2)$. Совместим аффинными преобразованиями изображения S_2 его точки a_2, b_2 и c_2 с точками соответственно a_1, b_1 и c_1 на S_1. При этом точка x_2 совместится с точкой x_1, а точки d_2 и e_2 займут положения, которые обозначим через d'_2 и e'_2 (рис. 1.4). Точка x_1 лежит на отрезке e_1d_1 или его продолжении. Точка x_2 — на отрезке e_2d_2 или его продолжении. После совмещения x_1 с x_2 точка x_1 должна, тем самым, быть пересечением отрезков e_1d_1 и $e'_2d'_2$ или их продолжений. Это и определяет положение прямой L_1 на S_1, а значит и L_2 на S_2.

Пусть S_1 и S_2 — a'-разные проекции тела T. Опишем процедуру построения по S_1 и S_2 некоторого тела T', a'-эквивалентного телу T. Пусть $a_1b_1c_1$ на S_1 и $a_2b_2c_2$ на S_2 — треугольники и точка e_1 лежит вне плоскости треугольника $a_1b_1c_1$. Выберем некоторую прямую α' (не параллельную плоскости изображения S_1) в качестве направления проекции. Проведем через точ-

ки a_1, b_1, c_1 и e_1 лучи, параллельные α', и на каждом луче возьмем по точке соответственно a', b', c', e' так, чтобы они не лежали в одной плоскости. Пусть теперь d_1 — произвольная точка на S_1. Построим соответствующую ей точку d' тела T'.

1. Пусть d_1 лежит в плоскости треугольника $a_1b_1c_1$. Совместим аффинными преобразованиями точки a_1, b_1 и c_1 с соответственно точками a', b' и c'. Точка, в которую перейдет d_1 при этом преобразовании, и есть d'.

2. Пусть d_1 лежит вне плоскости треугольника $a_1b_1c_1$.

Пусть хотя бы одна из троек $c_1d_1e_1$ и $c_2d_2e_2$ — треугольник. Тогда $((a_1b_1c_1)(a_2b_2c_2)(c_1e_1d_1)(c_2e_2d_2))$ — правильная четверка. Построим по S_1 и S_2 прямые L_1 и L_2 на них как было описано выше. Аффинными преобразованиями изображения S_1 совместим точки a_1, b_1 и c_1 с точками соответственно a', b' и c'. Прямую, в которую преобразуется L_1, обозначим через L'. Пусть, для определенности, $c_2d_2e_2$ — треугольник. S_2 Аффинными преобразованиями изображения (с прямой L_2) совместим L_2 с L' и точку e_2 с точкой e'. Точка, в которую перейдет при этом d_2, есть d'.

Пусть теперь и $c_1d_1e_1$ и $c_2d_2e_2$ — не треугольники. Возможны две ситуации.

2а). Точки c, d и e в теле T лежат на одной прямой. На S_1 и S_2 это проявляется тем, что в каждой из четверок a_1, c_1, e_1, d_1 и b_1, c_1, e_1, d_1 точки лежат в одной плоскости. В свою очередь, точки a_1, c_1, e_1 и d_1 лежат в одной плоскости, если при $a_1c_1e_1$ и $a_2c_2e_2$ — треугольниках $S_1(a_1, c_1, e_1, d_1) \approx S_2(a_2, c_2, e_2, d_2)$. Если же, например, $a_1c_1e_1$ — не треугольник (то есть направление проекции α_1 параллельно плоскости грани ace, то все четыре точки a_1, c_1, e_1, d_1 должны лежать на одной прямой. Аналогично определяется принадлежность одной плоскости и точек b_1, c_1, e_1 и d_1.

Аффинным преобразованием изображения S_1 совмещаем точки c_1 и e_1 с точками c' и e'. Точка, в которую переходит d_1, есть d'.

2б). Точки c, d и e в теле T образуют грань, и направления α_1 и α_2 оба параллельны этой грани. Если точка d лежит на одной прямой с точками a и e или b и e, то построение d' сводится к пункту 2а. Если ade и bde грани, то, например, четверка

$((a_1b_1c_1)(a_2b_2c_2)(a_1e_1d_1)(a_2e_2d_2))$ — правильная, и построение точки d' сводится к началу пункта 2.

Описание процедуры завершено.

Упражнения

1.1. Существует ли линейный по сложности правильный код для множества геометрических преобразований Γ_2?

1.2. Пусть M_A — функция нумерации изображения A, $|A| = n$,

$$r_{i,j} = \sqrt{(X(M_A^{-1}(i) - X(M_A^{-1}(j))^2 + (Y(M_A^{-1}(i) - Y(M_A^{-1}(j))^2},$$

$$T_A = \{r_{1,1}, r_{1,3}, r_{2,3}, r_{1,4}, r_{2,4}, r_{3,4}, \ldots, r_{1,n}, r_{2,n}, r_{3,n}\}.$$

Будет ли код $< M_A, T_A >$ правильным для множества преобразований Γ_2?

1.3. Будет ли правильным для множества преобразований Γ_3 код, полученный из кода, описанного в упражнении 1.2, делением каждого элемента на $r_{1,2}$?

1.4. Пусть K_A^3, K_B^3 — описанные ранее коды изображений A и B. Оценить сверху сложность алгоритма, выясняющего эквивалентность этих изображений.

1.5. Являются ли аффинно эквивалентными а) два ромба; б) два параллелограмма; в) две трапеции?

1.6. Даны два треугольника ΔABC и $\Delta A_1B_1C_1$. Докажите, что существует единственное аффинное преобразование, которое переводит точку A в A_1, B — в B_1, C — в C_1.

1.7. Докажите, что любое аффинное преобразование плоскости можно представить в виде композиции двух растяжений и преобразования подобия.

1.8. Докажите, что если аффинное преобразование φ переводит некоторую окружность в себя, то φ — либо поворот, либо симметрия.

1.9. На плоскости даны 3 вектора a, b, c, причем $\alpha a + \beta b + \gamma c = 0$. Докажите, что эти векторы аффинным преобразованием можно перевести в векторы равной длины тогда и только тогда, когда из отрезков с длинами $|\alpha|$, $|\beta|$, $|\gamma|$ можно составить треугольник.

1.10. Пусть φ — взаимно однозначное отображение плоскости в себя. Предположим, что оно обладает следующим свойством: если три точки лежат на одной прямой, то их образы тоже лежат на одной прямой. Докажите, что тогда φ — аффинное преобразование.

1.11. Придумайте правильный код для множества преобразований Γ_4, который имеет сложность по порядку меньшую, чем сложность кода K_A^4.

1.12. Дан выпуклый N-угольник, где $N < 100$, и множество $A = \{A_1, \ldots, A_{100}\}$ некоторых точек этого многоугольника. Известно, что все N вершин N-угольника принадлежат множеству A, и что никакие 3 точки не лежат на одной прямой, а никакие 4 точки не лежат на двух параллельных прямых. Разрешается задавать вопросы типа: чему равна площадь $\Delta A_i B_j C_k$? Докажите, что 300 вопросов достаточно, чтобы выяснить, какие точки являются вершинами многоугольника, и чтобы найти площадь многоугольника.

1.2 Алгебро-геометрические методы распознавания

Одним из основных подходов в распознавании образов является геометрический подход, при котором исследуемые объекты представляются векторами в евклидовом пространстве. Мы будем отождествлять вектор с объектом и предполагать, что разным объектам соответствуют разные вектора. Каждый из объектов относится к одному из конечного числа классов, объединяющих объекты с некоторой общностью свойств. В простейшем случае множества точек, соответствующие разным классам, не пересекаются. Однако, из-за неполноты описания, искажений при наблюдении, классы как множества векторов могут пересекаться. Задачей распознавания является отнесение неизвестного вектора к одному из рассматриваемых классов, классификация производится по принципу попадания конца вектора в область решения, аппроксимирующую определенным образом собственную область данного класса. Исходя из условия однозначности классификации объектов, области, соответствующие разным классам, должны определяться как непересекающиеся. В случае, если конец вектора не попадает ни в одну область решения, считается, что объект неопознан.

При выборе формы областей решения целесообразно стремиться к простым областям решения и простым решающим поверхностям, это удобно не только с точки зрения реализации на ЭВМ, а также дает уверенность в правильности обучающей вы-

борки. Кроме того, при детальном учете индивидуальных особенностей объектов обучающей выборки, когда качество распознавания именно обучающей выборки котируется очень высоко, область решения просто сужена до самой обучающей выборки, и может оказаться, что объекты контрольной выборки распознаются плохо. Все эти соображения обуславливают широкое применение линейных и кусочно-линейных алгоритмов распознавания, когда в качестве разделяющих поверхностей используются гиперплоскости или их куски.

В простейшем случае областями являются два полупространства, лежащие по разные стороны от одной разделяющей гиперплоскости, называемой решающей. Использование линейных методов целесообразно как с точки зрения простоты их реализации, так и с точки зрения широты приложений. С помощью линейных методов разделяющая поверхность получается не хуже, чем в методах минимизации Евклидова расстояния или в корреляционных методах. Они инвариантны относительно многократных линейных преобразований пространства наблюдений, а такие преобразования применяются в предварительной обработке для улучшения качества распознавания. В линейных методах такие преобразования не нужны. Области, для которых существует разделяющая их гиперплоскость, называются линейно разделимыми, а если такой плоскости нет, то они называются линейно неразделимыми.

При распознавании объектов с помощью линейных методов решающая поверхность строится так, что области решения для любой пары классов отделяются друг от друга плоскостью. Многоальтернативная задача при этом сводится к двухальтернативной задаче распознавания, т.е. распознавания объектов из двух классов. Обратный переход от двухальтернативной задачи к многоальтернативной выполняется путем группировки двухальтернативных задач. Рассмотрим конкретные линейные методы распознавания.

1.2.1 Некоторые эвристические методы распознавания

Метод эталонов

Метод применим в случае, когда точки каждого класса расположены рядом друг с другом, а центры сгущения этих точек отстоят друг от друга на сравнительно большом расстоянии. Для каждого класса по обучающей выборке строится эталон $\bar{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)$, имеющий значения признаков

$$x_i^0 = \frac{1}{k} \sum_{j=1}^{k} x_i^j,$$

где $\{x^j = (x_1^j, x_2^j, \ldots, x_n^j) : j = 1, 2, \ldots k\}$ — множество объектов данного образа в обучающей выборке. Эталон — это усредненный по обучающей выборке абстрактный объект, он может не совпадать ни с одним объектом генеральной совокупности.

Принадлежность объекта $\bar{x}$ к тому или иному образу определяется по расстоянию до эталонов всех образов, и система относит $\bar{x}$ к тому образу, расстояние до эталона которого минимально. Расстояние измеряется в той метрике, которая введена для решения задачи распознавания. Как правило — это метрика Евклида, метрика "городских кварталов" или метрика Хэмминга.

Метод дробящихся эталонов

Если метод эталонов работает не удовлетворительно, то его ошибки можно исправить с помощью дополнительных эталонов. Процесс обучения состоит в следующем. На первом этапе в обучающей выборке "охватывают" все объекты каждого класса гиперсферой возможно меньшего радиуса. Сделать это можно, например, так. Строится эталон каждого класса. Вычисляется расстояние от эталона до всех объектов данного класса, входящих в обучающую выборку. Выбирается максимальное из этих расстояний r_{max}. Строится гиперсфера с центром в эталоне и радиусом $R = r_{max} + \varepsilon$. Она охватывает все объекты данного класса. Такая процедура проводится для всех классов (обра-

зов). На рисунке 1.5 приведен пример двух образов в двухмерном признаковом пространстве.

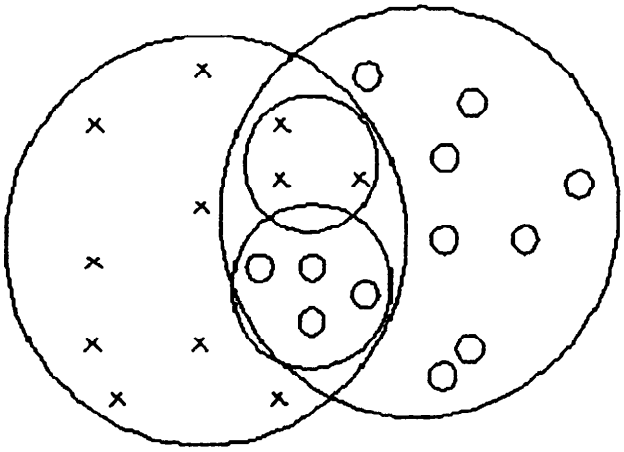

Рис. 1.5: Метод дробящихся эталонов.

Если гиперсферы различных образов пересекаются и в области перекрытия оказываются объекты более чем одного образа, то для них строятся гиперсферы второго уровня, затем третьего и т.д. до тех пор, пока области не окажутся непересекающимися, либо в области пересечения будут присутствовать объекты только одного образа.

Распознавание осуществляется следующим образом. Определяется местонахождение объекта относительно гиперсфер первого уровня. При попадании объекта в гиперсферу, соответствующую одному и только одному образу, процедура распознавания прекращается. Если же объект оказался в области перекрытия гиперсфер, которая при обучении содержала объекты более чем одного образа, то переходим к гиперсферам второго уровня и проводим действия такие же, как для гиперсфер первого уровня. Этот процесс продолжается до тех пор, пока принадлежность неизвестного объекта тому или иному образу не определится однозначно. Правда, это событие может и не наступить. В частности, неизвестный объект может не попасть ни в одну из гиперсфер какого-либо уровня. В этих случаях "учитель" должен включить в решающие правила соответствующие действия. Например, система может либо отказаться от решения об однозначном отнесении объекта к какому-либо образу, либо использовать критерий минимума расстояния до этало-

нов данного или предшествующего уровня и т.п. Какой из этих приемов эффективнее, сказать трудно, т.к. метод дробящихся эталонов носит в основном эмпирический характер.

Метод ближайших соседей

Обучение состоит в запоминании всех объектов обучающей выборки. При распознавании система относит объект $\bar{x}$ к тому образу, чей "представитель" оказался ближе всех к $\bar{x}$. Модификацией метода является правило большинства ближайших соседей. Оно работает так: строится окрестность радиуса R с центром в $\bar{x}$ и распознавание осуществляется по большинству "представителей" какого-либо образа, оказавшихся внутри окрестности. Сложность состоит в том, чтобы правильно выбрать размер окрестности, он должен быть достаточно большим, чтобы в нее попало относительно большое число "представителей" разных образов, и достаточно маленьким, чтобы не работать со всей обучающей выборкой. Метод ближайших соседей требует хранения всей обучающей выборки, дает хорошие результаты, но требует большого времени распознавания. В ряде задач, таких как распознавание речи, рукописных тестов и фотографий, сравнение двух "представителей" — достаточно медленная операция, а принимать решение нужно в реальное время.

Для сокращения числа запоминаемых объектов можно применять комбинированные решающие правила, например сочетание метода дробящихся эталонов и ближайших соседей. В этом случае запоминанию подлежат те объекты обучающей выборки, которые находятся вблизи разделяющей образы границы.

Метод потенциальных функций

Название метода связано со следующей аналогией. Представим себе, что в точки $\bar{x}_j$ обучающей выборки помещены заряды $+q_j$ для объектов из класса K_1, и $-q_j$ — для объектов из класса K_2. Определим потенциал точки $\bar{x}$, создаваемый единичным зарядом, находящимся в точке 0 как $P(\bar{x})$, а общий потенциал, создаваемый всеми зарядами, как их сумму $g(\bar{x}) =$

$\sum_{j=1}^{n} q_j P(\bar{x} - \bar{x}_j)$. Здесь $P(\bar{x})$ — потенциальная функция, которая монотонно убывает до нуля с увеличением $|\bar{x}|$. Простейший случай, когда

$$P(\bar{x}) = \begin{cases} 0 & \text{при } |\bar{x}| > 1, \\ 1 - |\bar{x}|, & \text{иначе} \end{cases},$$

изображен на рисунке 1.6.

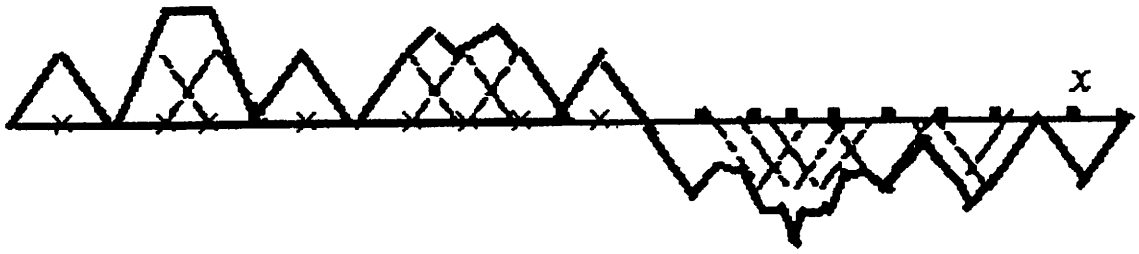

Рис. 1.6: Метод потенциальных функций.

На рисунке 1.6 пунктиром изображены потенциальные функции, порожденные одиночным объектом, а сплошной линией — суммарная потенциальная функция.

Функция электростатического потенциала используется в качестве решающего правила. Если потенциал $g(\bar{x})$ положителен, то $\bar{x}$ относят к классу K_1, если отрицателен, — то к K_2. При большом объеме обучающей выборки эти вычисления достаточно громоздки, и удобнее оценивать границу, разделяющую образы.

В зависимости от вида потенциальных функций возможны следующие случаи. Если $P(\bar{x})$ быстро убывают с ростом расстояния, можно добиться безошибочного разделения обучающих выборок. Однако при этом возникают трудности при распознавании неопознанных объектов (снижается достоверность принимаемого решения, возрастает зона неопределенности). При слишком "пологих" потенциальных функциях увеличивается количество ошибок распознавания, в том числе и на обучающих объектах.

1.2.2 Модель персептрона Розенблатта

В середине 50-х годов была предложена модель нейрона (см Рис. 1.7) — живой клетки которая, различая по заряду цитоплазмы, может находиться в двух состояниях: покоя и возбуждения.

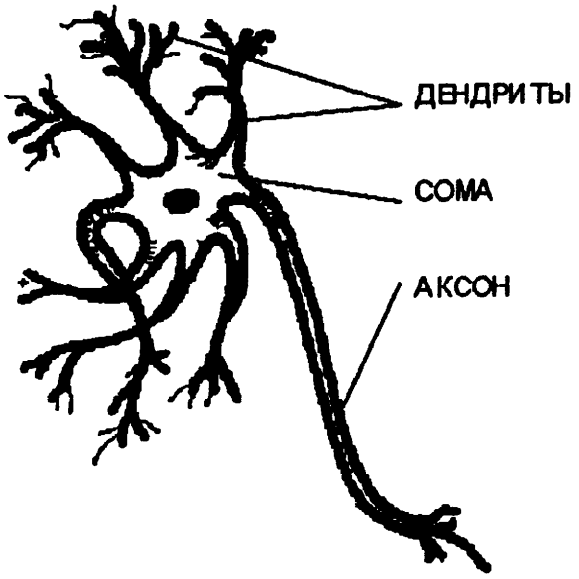

Рис. 1.7: Общая схема строения биологического нейрона.

Аксоны одних клеток соединены с дендритами других. Текущее состояние нейрона зависит от состояния аксонов соединенных с ним других нейронов и чувствительности дендритов.

Основываясь на этой модели, нейрона в 1957 году Розенблатт (F.Rosenblatt) предложил модель персептрона (PERCEPTRON), одну из первых искусственных сетей, способных к перцепции (восприятию) и формированию реакции на воспринятый стимул.

Это физическое устройство, состоящее из трех слоев:

- рецепторный слой (20 × 20 фотоэлементов);
- передающий слой (512 нейронов, каждый имеет по 10 входов, случайным образом соединенных с элементами рецепторного слоя), причем для каждого $j = 1, 2, \ldots, 512$ имеет место

$$y_j = \begin{cases} 1, \sum_{i=1}^{10} x_{ji} \geqslant 5 \\ 0, \text{ иначе} \end{cases},$$

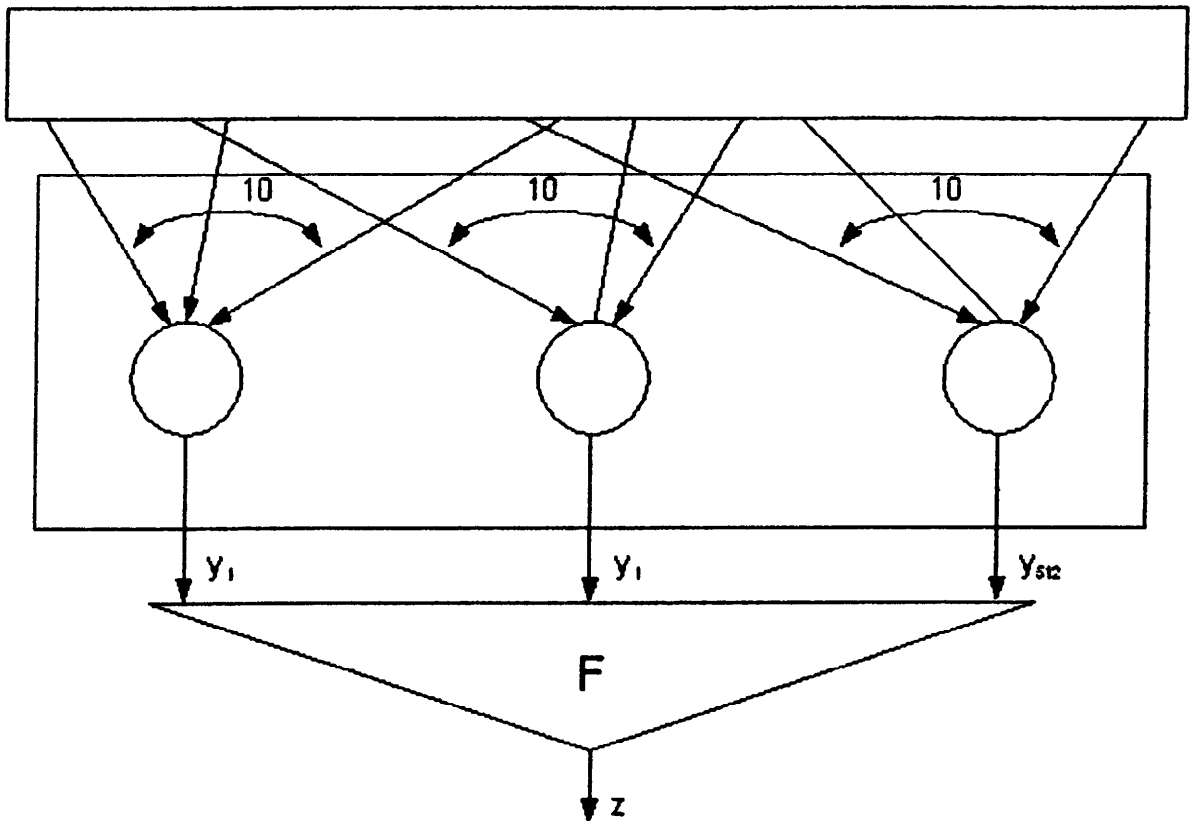

Рис. 1.8: Персептрон Розенблатта

где x_{ji} — входы j-го элемента рецепторного слоя;

- решающий элемент, принимающий значение

$$z = F(y_1, \ldots, y_{512}) = \begin{cases} 1, & \text{если } \bar{a} \cdot \bar{y} \geqslant c \\ 0, & \text{иначе} \end{cases},$$

где $\bar{a}$ — весовой вектор решающего элемента, c — пороговое число.

Считается, что схема соединения нейронов фиксирована и не может изменяться в процессе обучения, а дендриты нейронов могут менять чувствительность. То есть в процессе обучения персептрона можно менять весовой вектор $\bar{a}$ и порог c.

В данной схеме образ подается на рецепторный слой. Поскольку каждый элемент передающего слоя жестко связан с элементами рецепторного слоя, то на входе решающего элемента образ кодируется вектором длины, равной количеству элементов в передающем слое, в данном случае 512. Этот вектор $\bar{y} = (y_1, \ldots, y_{512})$ называется вектором признаков.

Считается, что образы, подаваемые на персептрон, принадлежат одному из двух классов. Хотелось бы так настроить весовые коэффициенты $\bar{a} = (a_1, \ldots, a_{512})$ решающего элемента, чтобы на образах из первого класса решающий элемент выдавал 0, а на образах из второго класса — 1. Настройка весовых коэффициентов осуществляется с помощью обучающего алгоритма, на вход которого поступает обучающая выборка, т.е. последовательность образов, для которых известно, к какому классу они принадлежат. В зависимости от правильности отнесения очередного образа из обучающей выборки к своему классу обучающий алгоритм может менять весовые коэффициенты.

Возникает вопрос: какие классы образов могут распознаваться персептроном? Ответ на этот вопрос тривиален. Из самого вида решающего правила видно, что распознаваться могут только классы, которые в признаковом пространстве могут быть отделены друг от друга гиперплоскостью. Тогда встает вопрос: если классы образов отделимы гиперплоскостью в признаковом пространстве, то существует ли алгоритм обучения персептрона? Ответ на этот вопрос дает теорема американского ученого Новикова.

1.2.3 Теорема Новикова

Пусть $\mathbb{R}^k$ — признаковое пространство, $M_i \subseteq \mathbb{R}^k$, $M_i \neq \varnothing$, $i = 1, 2$, $M_1 \bigcap M_2 = \varnothing$. Множества M_1 и M_2 *линейно отделимы* (*строго линейно отделимы*) точно тогда, когда существуют $\bar{a} \in \mathbb{R}^k$ и $c \in \mathbb{R}$, что для любых $\bar{x}_1 \in M_1$ и $\bar{x}_2 \in M_2$ выполнено $\bar{a} \cdot \bar{x}_1 \geqslant c, \bar{a} \cdot \bar{x}_2 < c$ ($\bar{a} \cdot \bar{x}_1 > c, \bar{a} \cdot \bar{x}_2 < c$).

Множества $M_1, M_2 \subseteq \mathbb{R}^k$ строго 0-отделимы точно тогда, когда существуют $\bar{a} \in \mathbb{R}^n$, что для любых $\bar{x}_1 \in M_1$ и $\bar{x}_2 \in M_2$ выполнено $\bar{a} \cdot \bar{x}_1 > 0, \bar{a} \cdot \bar{x}_2 < 0$.

Множество $M \subseteq \mathbb{R}^k$ строго линейно отделимо от нуля точно тогда, когда существует $\bar{a} \in \mathbb{R}^k$, что для любых $\bar{x} \in M$ выполнено $\bar{a} \cdot \bar{x} > 0$.

Утверждение 1. *Если* $M_1, M_2 \in \mathbb{R}^k$ *и* $\widetilde{M}_i = \{(x_1, \ldots, x_k, 1) : (x_1, \ldots, x_k) \in M_i\}, i = 1, 2$, *то* M_1 *и* M_2 *строго линейно отделимы тогда и только тогда, когда* $\widetilde{M}_1$ *и* $\widetilde{M}_2$ *строго* 0*-отделимы.*

Доказательство. 1. Пусть M_1 и M_2 строго линейно отделимы гиперплоскостью $l : \bar{a} \cdot \bar{x} = c$. Определим $\bar{b} \in \mathbb{R}^{k+1}$, $\bar{b} = (a_1, \ldots, a_k, -c)$. Тогда для всех $\bar{x}' \in \widetilde{M_1}, \bar{x}' = (\bar{x}, 1)$ выполнено $\bar{b} \cdot \bar{x}' = \bar{a} \cdot \bar{x} - 1 \cdot c > 0$. И для всех $\bar{x}' \in \widetilde{M_2}, \bar{x}' = (\bar{x}, 1)$ выполнено $\bar{b} \cdot \bar{x}' = \bar{a} \cdot \bar{x} - 1 \cdot c < 0$. То есть множества $\widetilde{M_1}$ и $\widetilde{M_2}$ строго 0-отделимы гиперплоскостью $l' : \bar{b} \cdot \bar{x}' = 0$.

2. Обратно, пусть множества $\widetilde{M_1}$ и $\widetilde{M_2}$ строго 0-отделимы гиперплоскостью $l' : \bar{b} \cdot \bar{x}' = 0$. Положим $c = -b_{k+1}$, $\bar{a} = (b_1, \ldots, b_k)$. Тогда M_1 и M_2 строго линейно отделимы гиперплоскостью $l : \bar{a} \cdot \bar{x} = c$. □

Утверждение 2. *Если* $M_1, M_2 \in \mathbb{R}^k$ *и* $M' = M_1 \bigcup \{-\bar{x} : \bar{x} \in M_2\}$, *то* M_1 *и* M_2 *строго* 0*-отделимы тогда и только тогда, когда* M' *строго линейно отделимо от* 0.

Доказательство. 1. Пусть M' строго линейно отделимо от 0, то есть $\exists \bar{a} \forall \bar{x} \in M' \bar{a} \cdot \bar{x} > 0$. Пусть $\bar{x} \in M_1$, тогда $\hat{x} \in M'$ и $\bar{x} \cdot \bar{a} > 0$, а если $\bar{x} \in M_2$, то $-\bar{x} \in M'$ и $\bar{x} \cdot \bar{a} < 0$. То есть M_1 и M_2 строго 0-отделимы гиперплоскостью $l : \bar{a} \cdot \bar{x} = 0$.

2. Обратно, пусть M_1 и M_2 строго 0-отделимы гиперплоскостью $l : \bar{a} \cdot \bar{x} = 0$. Тогда $\bar{x} \cdot \bar{a} > 0$ для всех $x \in M'$. То есть M' строго линейно отделимо от 0. □

Пусть даны два строго линейно отделимых множества M_1, $M_2 \subset \mathbb{R}^{n-1}$. Пусть $M' = M_1 \bigcup \{-\bar{x} : \bar{x} \in M_2\}$ и

$$M = \{(x_1, \ldots, x_{n-1}, 1) : (x_1, \ldots, x_{n-1}) \in M'\} \subset \mathbb{R}^n.$$

Тогда согласно утверждениям 1 и 2 множество M строго линейно отделимо от нуля.

Рассмотрим следующий алгоритм A, который для строго линейно отделимого от нуля множества M позволяет находить нормаль отделяющей гиперплоскости. На вход алгоритма поступает бесконечная последовательность $\bar{y}_1, \bar{y}_2, \ldots$ такая, что для любого $i = 1, 2, \ldots$ выполнено $\bar{y}_i \in M$. Эта последовательность называется обучающей последовательностью. Алгоритм A состоит в итеративном уточнении нормали $\bar{a}$ отделяющей гиперплоскости, называемой весовым вектором, по следующей схеме:

Шаг 0. $\bar{a}_0 := (0, \ldots, 0)$.

Шаг i ($i > 0$). Если $\bar{y}_i \cdot \bar{a}_{i-1} \leq 0$, то $\bar{a}_i = \bar{a}_{i-1} + \bar{y}_i$, иначе $\bar{a}_i = \bar{a}_{i-1}$.

Для конечного $M = \{\bar{y}_1, \ldots, \bar{y}_s\}$ из $\mathbb{R}^n$ введем обозначения:

$$D(M) = \max_{\bar{y} \in M} ||\bar{y}||,$$

$$V(M) = \left\{ \sum_{i=1}^{s} \alpha_i \bar{y}_i : \sum_{i=1}^{s} \alpha_i = 1, \alpha_i \geq 0, i = 1, \ldots, s \right\} -$$

выпуклая оболочка множества M,

$$\rho(M) = \min_{\bar{y} \in V(M)} ||\bar{y}||.$$

Теорема 4 (Новикова). *Пусть $M \subset \mathbb{R}^n$ строго линейно отделимо от 0, $|M| < \infty$ и $\rho(M) > 0$. Пусть $\bar{y}_1, \bar{y}_2, \ldots$ — обучающая последовательность такая, что для любого $i = 1, 2, \ldots$ выполнено $\bar{y}_i \in M$, и каждый элемент $\bar{y}$ из M встречается в обучающей последовательности бесконечное число раз. Пусть $\bar{a}_1, \bar{a}_2, \ldots$ — последовательность весовых векторов, полученных в результате применения алгоритма A к обучающей последовательности $\bar{y}_1, \bar{y}_2, \ldots$. Тогда существует натуральное N такое, что для любого $i \geq N$ выполняется $\bar{a}_i = \bar{a}_N$, при этом для любого $\bar{y} \in M$ справедливо $\bar{a}_N \cdot \bar{y} > 0$, и для числа изменений s в последовательности $\bar{a}_1, \bar{a}_2, \ldots$ выполнено $s \leq \left[\frac{D^2(M)}{\rho^2(M)}\right]$.*

Доказательство. Пусть $i_1, i_2, \ldots$ — последовательность всех индексов, для которых $\bar{y}_{i_j} \cdot \bar{a}_{i_j - 1} \leq 0$, $j = 1, 2, \ldots$, т.е. это последовательность индексов, в которых последовательность $\bar{a}_1, \bar{a}_2, \ldots$ изменяется. Обозначим $\tilde{y}_j = \bar{y}_{i_j}$, $\tilde{a}_0 = \bar{a}_0$, $\tilde{a}_j = \bar{a}_{i_j}$, $j = 1, 2, \ldots$.

Пусть $\bar{x}$ — ближайшая к 0 точка из $V(M)$, т.е. $||\bar{x}|| = \rho(M)$. Обозначим $\bar{e} = \frac{\bar{x}}{||\bar{x}||}$. Рассмотрим гиперплоскость H, задаваемую уравнением $\bar{e} \cdot \bar{y} = ||\bar{x}||$.

Легко показать, что все точки из $V(M)$ лежат не ниже, чем гиперплоскость H, т.е. для любой точки $\bar{y} \in V(M)$ выполнено $\bar{e} \cdot \bar{y} \geq ||\bar{x}||$. В самом деле, предположим, что существует точка $\bar{y} \in V(M)$ такая, что $\bar{e} \cdot \bar{y} < ||\bar{x}||$. Проведем через 3 точки $\bar{x}$, $\bar{y}$ и точку 0 плоскость L. На рисунке 1.9 изображены эти точки на плоскости L. Здесь прямая $(l, \bar{x})$ — есть прямая пересечения

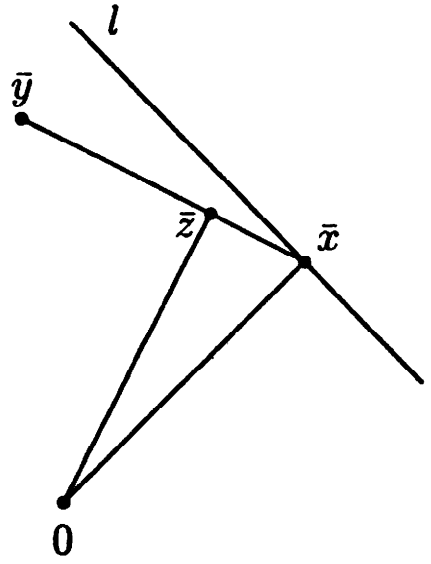

Рис. 1.9:

плоскости L и гиперплоскости H, т.е. прямая $(l, \bar{x})$ перпендикулярна отрезку $[0, \bar{x}]$. Так как $\bar{e} \cdot \bar{y} < ||\bar{x}||$, то угол $\angle 0\bar{x}\bar{y}$ — острый. Из точки 0 опустим на отрезок $[\bar{x}, \bar{y}]$ перпендикуляр $[0, \bar{z}]$. Так как $\bar{y} \in V(M)$ и $\bar{x} \in V(M)$, то и $\bar{z} \in V(M)$. Из остроты угла $\angle 0\bar{x}\bar{y}$ следует, что $||\bar{z}|| < ||\bar{x}||$, что противоречит минимальности $||\bar{x}||$.

Так как для любого допустимого индекса k справедливо $\tilde{a}_k = \tilde{a}_{k-1} + \tilde{y}_k$, то

$$||\tilde{a}_k|| \geq \tilde{a}_k \cdot \bar{e} = \tilde{a}_{k-1} \cdot \bar{e} + \tilde{y}_k \cdot \bar{e} \geq \tilde{a}_{k-1} \cdot \bar{e} + \rho(M) \geq$$
$$\geq \tilde{a}_0 \cdot \bar{e} + k\rho(M) = k\rho(M).$$

С другой стороны

$$\tilde{a}_k \cdot \tilde{a}_k = \tilde{a}_{k-1} \cdot \tilde{a}_{k-1} + \tilde{y}_k \cdot \tilde{y}_k + 2\tilde{a}_{k-1} \cdot \tilde{y}_k.$$

Но $\tilde{y}_k \cdot \tilde{y}_k \leq D^2(M)$ и $\tilde{a}_{k-1} \cdot \tilde{y}_k \leq 0$ и, значит,

$$||a_k||^2 \leq ||a_{k-1}||^2 + D^2(M) \leq ||a_0||^2 + kD^2(M) \leq kD^2(M).$$

Следовательно, $k^2\rho^2(M) \leq kD^2(M)$ и $k \leq [D^2(M)/\rho^2(M)]$ для любого допустимого индекса k. Откуда сразу следует, что для числа s изменений весовых векторов верно

$$s \leq [D^2(M)/\rho^2(M)].$$

Возьмем $N = i_s$. Тогда для любого индекса $j \geq N$ справедливо $\tilde{a}_j = \tilde{a}_N$, а так как в последовательности $\bar{y}_N, \bar{y}_{N+1}, \ldots$ встреча-

ются все элементы множества M, то для любого $\bar{y} \in M$ справедливо $\bar{a}_N \cdot \bar{y} > 0$.

Теорема доказана. □

Упражнения

1.13. Построить линейную функцию, разделяющую множества $M_1, M_2 \subset \mathbb{R}^2$, если

а) $M_1 = \{(-2,3),(-1,-2),(0,0),(1,1),(3,-3)\}$ и
$M_2 = \{(1,4),(2,1),(2,2),(4,-1)\}$;

б) $M_1 = \{(1,-1),(2,2),(-1,2),(-3,-2)\}$ и
$M_2 = \{(1,-3),(0,-6),(3,-2),(4,-4),(5,4)\}$.

в) $M_1 = \{(2,2),(3,3),(4,4),(4,5),(3,1),(5,1)\}$ и
$M_2 = \{(5,4),(7,5),(7,-1),(9,2)\}$.

1.14. Существует ли прямая, разделяющая множества $M_1, M_2 \subset \mathbb{R}^2$, если $M_1 = \{(0,2),(2,0),(2,2),(2,4),(4,2)\}$ и $M_2 = \{(3,0),(4,4),(5,0)\}$?

1.15. Доказать, что множества $M_1, M_2 \subset \mathbb{R}^2$ нельзя отделить линейной функцией и построить нелинейную функцию разделяющую их, если

а) $M_1 = \{(0,2),(0,-2),(1,-2),(1,4),(2,3),(-1,3),(-1,-2)\}$,
$M_2\{(1,0),(2,0),(2,-1),(3,1),(4,-2),(5,2)\}$;

б) $M_1 = \{1,0),(2,0),(2,1),(3,-1),(4,2),(5,-2)\}$,
$M_2 = \{(0,-2),(0,2),(1,2),(1,-4),(2,-3),(-1,-3),(-1,2)\}$.

1.16. Дано: $M_1 = \{(1,3),(2,2),(2,4),(3,1),(4,1),(4,4)\}$;
$M_2 = \{(1,4),(1,5),(2,5),(3,4),(4,2),(5,3)\}$. Определить, к какому классу относится точка $\xi = (3,3)$

а) методом ближайшего соседа;

б) методом k-ближних (рассмотреть случай $k = 5$).

1.17. Дано: $M_1 = \{(1,4),(2,3),(3,4),(4,3),(5,4)\}$;
$M_2 = \{(1,1),(2,0),(3,1),(4,0),(5,1)\}$. Определить, к какому классу относится точка $\xi = (3,2)$

а) методом ближайшего соседа;

б) методом k-ближних (рассмотреть случай $k = 3$).

1.18. $M_1 = \{2, 5, 8\}, M_2 = \{3, 4, 7\}$. Определить, к какому классу относится точка $\xi = 6$ методом потенциальных функций. Потенциальная функция имеет вид

$$P(x) = \begin{cases} 0, & \text{если } x \leq -2 \\ x/2 + 1, & \text{если } -2 < x \leq 0 \\ 1 - x/2, & \text{если } 0 < x \leq 2 \\ 0, & \text{если } x > 2. \end{cases}$$

1.19. Даны множества $M_1, M_2 \subset \mathbb{R}^2$. Отнести точку $\xi = (0, 1)$ к одному из классов по методу

а) ближайшего соседа,

б) потенциальных функций с потенциалом $P(x, y) = \max(0, 2 - |x| - |y|)$,

в) потенциальных функций с потенциалом $P(x, y) = \max(0, 5 - |x| - |y|)$,

при условии, что

а) $M_1 = \{(0, 0), (1, -3), (-1, -3), (2, -2), (-2, -2)\}$,
$M_2 = \{(1, 2), (-1, 2), (0, 3), (2, 3), (-2, 3)\}$;

б) $M_1 = \{(0, 2), (1, 5), (-1, 5), (2, 4), (-2, 4)\}$,
$M_2 = \{(1, 0), (-1, 0), (0, -1), (2, -1), (-2, -1)\}$.

1.20. Описать критерий отделимости двух множеств персептроном.

1.21. Какие функции алгебры логики от двух переменных представимы персептроном, а какие – нет? Ответ обосновать.

1.22. Дано множество $M \subset \mathbb{R}^n$, строго линейно отделимое от 0, $|M| = 7$, $D(M) = 10$, $\rho(M) = 1$. Во власти экспериментатора составить обучающую выборку для подачи на вход персептрона. Каждую секунду на вход персептрона может подаваться 1 элемент обучающей выборки. Чему равно минимальное время, через которое экспериментатор гарантированно будет знать, что персептрон получил нормальный вектор отделяющей гиперплоскости для множества M? Как должна выглядеть обучающая выборка, чтобы обеспечить этот результат?

1.3 Статистический подход к распознаванию

Рассматривается следующая задача распознавания образов. Дано множество X объектов, относительно которых производится распознавание. Без существенного ограничения общности будем считать, что оно представляется в виде объединения двух классов K_1 и K_2. Классы K_1 и K_2 неизвестны, но дана обучающая выборка, которая представляет собой конечное подмножество множества X, и про каждый элемент этого подмножества сообщается, к какому классу он принадлежит. По обучающей выборке надо выработать решающее правило, которое по предъявлению ему объекта из X решает, к какому из классов его отнести.

1.3.1 Качество и надежность решающего правила

При статистическом подходе к распознаванию считается, что обучающая выборка формируется следующим образом. На множестве X задано распределение вероятностей $P(x)$, и объекты x появляются случайно и независимо в соответствии с этим распределением. Существует "учитель", который относит объекты к одному из двух классов согласно условной вероятности $P(i|x)$, которая говорит о вероятности того, что объект x будет отнесен к классу K_i, $i \in \{1, 2\}$. Ни распределение $P(x)$, ни правило классификации $P(i|x)$ нам не известны, но известно, что обе функции существуют, и, значит, существует совместное распределение вероятностей $P(x, i) = P(x) \cdot P(i|x)$.

Классы K_1 и K_2 при статистическом подходе так же задаются функциями распределения вероятностей. А именно, считается, что существуют условные плотности распределения $P(x|i = 1)$ и $P(x|i = 2)$, задающие плотность распределения вероятностей объектов первого и второго классов соответственно. Также существуют величины P_1 и P_2, которые определяют вероятность появления объектов соответственно первого и второго классов. Существование этих функций и величин не предполагает, что мы их знаем.

Также считается, что определено множество $\mathcal{F}$ решающих правил $F(x,\alpha)$. В этом множестве каждое правило определяется заданием параметра α (иногда удобно понимать параметр α как вектор). Все правила $F(x,\alpha)$ при фиксации параметра α могут принимать одно из двух значений: единица или двойка, говорящие к какому из двух классов отнесен объект x.

Для каждой функции $F(x,\alpha)$ может быть определено качество $P(\alpha)$ как вероятность неправильных классификаций, т.е. вероятность того, что учитель и решающее правило классифицируют объект по разному. Формально качество $P(\alpha)$ функции $F(x,\alpha)$ определяется так:

а) в случае, когда пространство X дискретно и состоит их точек $x_1,\ldots,x_n$,

$$P(\alpha)=\sum_{i=1}^{2}\sum_{j=1}^{n}(i-F(x_j,\alpha))^2P(x_j)P(i|x_j),$$

где $P(x_j)$ — вероятность возникновения объекта x_j;

б) в случае, когда в пространстве X существует плотность распределения $P(x)$,

$$P(\alpha)=\sum_{i=1}^{2}\int(i-F(x,\alpha))^2P(x)P(i|x)dx;$$

в) в общем случае можно считать, что в пространстве $X\times\{1,2\}$ задана вероятностная мера $P(x,i)$, и тогда

$$P(\alpha)=\int_{x,i}(i-F(x,\alpha))^2dP(x,i).$$

Среди всех функций $F(x,\alpha)$ есть такая $F(x,\alpha_0)$, которая минимизирует вероятность ошибок. Эту-то наилучшую в классе функцию (или близкую к ней, т.е. функцию с качеством, отличным от $P(\alpha_0)$ не более чем на малую величину ε) и следует найти. Однако, поскольку совместное распределение вероятностей $P(x,i)$ неизвестно, поиск ведется с использованием обучающей последовательности

$$x_1i_1,x_2i_2,\ldots,x_li_l,$$

т.е. случайной и независимой выборки примеров фиксированной длины l, где x_j — объект, i_j — номер класса, которому принадлежит объект x_j, $j = 1, 2, \ldots, l$. Понятно, что в общем случае нельзя найти алгоритм, который по конечной выборке безусловно гарантировал бы успех поиска. Успех можно гарантировать с некоторой вероятностью $1 - \eta$.

Таким образом, задача заключается в том, чтобы для любой функции $P(x, i)$ среди решающих правил $F(x, \alpha)$ найти по обучающей последовательности фиксированной длины l такую функцию $F(x, \alpha^*)$, о которой с надежностью, не меньшей $1 - \eta$, можно было бы утверждать, что ее качество отличается от качества лучшей функции $F(x, \alpha_0)$ на величину, не превышающую ε.

Эта задача есть частный случай известной в математике задачи, называемой задачей минимизации среднего риска.

В качестве примера можно привести решающее правило персептрона. Если классы K_1 и K_2 0-отделимы, и

$$\theta(z) = \begin{cases} 1, & \text{если } z \geq 0, \\ 2, & \text{если } z < 0, \end{cases} \tag{1.1}$$

то решающее правило персептрона выглядит как

$$F(x, \alpha) = \theta(x \cdot \alpha),$$

где α — весовой вектор и в данном случае параметр решающего правила. Качество же решающего правила определяется так:

$$P(\alpha) = \int_{x,i} (i - \theta(x \cdot \alpha))^2 dP(x, i).$$

Подведем итог. Способность к обучению характеризуется двумя понятиями:

а) качеством полученного решающего правила (вероятностью неправильных ответов; чем меньше эта вероятность, тем выше качество);

б) надежностью получения решающего правила с заданным качеством (вероятностью получения заданного качества; чем выше эта вероятность, тем выше надежность успешного обучения).

Задача сводится к созданию такого обучающего устройства, которое по обучающей последовательности строило бы решающее правило, качество которого с заданной надежностью было бы не ниже требуемого.

1.3.2 Байесовское решающее правило

Один из путей решения задачи минимизации среднего риска связан с идеей восстановления функций распределения вероятностей.

Как мы уже упоминали, считаем, что существуют условные плотности распределения $P(x|i = 1)$ и $P(x|i = 2)$, задающие плотность распределения вероятностей объектов первого и второго классов образов соответственно, а также существуют величины P_1 и P_2, которые определяют вероятность появления объектов соответственно первого и второго классов. Если мы известными в статистике методами по обучающей последовательности сумеем восстановить эти функции и величины, то дальше мы можем действовать следующим образом.

Сначала с помощью формулы Байеса определяем вероятность принадлежности объекта x к первому или второму классу:

$$P(i = 1|x) = cP(x|i = 1)P_1,$$

$$P(i = 2|x) = cP(x|i = 2)P_2,$$

где

$$c = \frac{1}{P(x|i = 1)P_1 + P(x|i = 2)P_2} -$$

нормирующий множитель.

Нетрудно понять, что минимальные потери получены при такой классификации объектов, при которой объект x будет отнесен к первому классу в случае выполнения неравенства

$$P(i = 1|x) \geq P(i = 2|x),$$

или, иными словами, если выполняется байесовское решающее правило

$$\frac{P(x|i = 1)}{P(x|i = 2)} \geq \frac{P_2}{P_1}.$$

Следовательно, оптимальную классификацию обеспечивает следующее решающее правило

$$F(x) = \theta(\ln P(x|i=1) - \ln P(x|i=2) + \ln \frac{P_1}{P_2}), \qquad (1.2)$$

где $\theta(z)$ определяется формулой (1.1). Такие решающие правила иногда называют дискриминантными.

Таким образом, знание плотностей условных распределений $P(x|i=1)$, $P(x|i=2)$ и вероятностей P_1, P_2 гарантирует отыскание оптимального правила классификации. Однако следует заметить, на этом пути мы решение сравнительно простой задачи — построение дискриминантной функции — подменяем решением значительно более сложной задачи — задачи о восстановлении функции распределения. Ведь восстанавливаемые функции распределения вероятностей составляют исчерпывающие сведения о классах распознаваемых объектов, в то время как нужная нам дискриминантная функция отражает только одну из характеристик взаимного расположения объектов различных классов. Поэтому решать задачу обучения распознаванию образов, восстанавливая неизвестные функции распределения вероятностей, в общем случае нерационально. Исключения составляют специальные случаи, когда задачи о восстановлении многомерных функций распределений сильно вырождаются. Например, когда функция распределения такова, что координаты вектора $x = (x_1, \ldots, x_n)$ распределены независимо, или если $P(x|i=1)$, $P(x|i=2)$ — нормальные распределения с одинаковыми ковариационными матрицами, отличающиеся только векторами средних. В последнем случае решающее правило (1.2) задает разделяющую гиперплоскость. Если же ковариационные матрицы $P(x|i=1)$, $P(x|i=2)$ не только одинаковы, но и диагональны, то байесовское решающее правило эквивалентно методу эталонов и относит объект к тому классу, евклидово расстояние до эталона которого минимально.

Таким образом, решающие правила, ранее рассмотренные нами как эвристические, имеют еще и статистическую трактовку и даже, в ряде конкретных случаев, являются статистически оптимальными.

1.3.3 Метод минимизации эмпирического риска

Пусть $z = (x, i)$ — некоторая ситуация, состоящая в том, что объект x относится к классу K_i. Если $F(x, \alpha)$ — некоторое решающее правило, то функция $Q(z, \alpha) = (i - F(x, \alpha))^2$ определяет величину потерь при появлении ситуации z, т.е она равна 1, если решающее правило неправильно классифицировало объект x, и 0 — в противном случае.

Как мы уже отмечали, задача распознавания образов в статистическом подходе сводится к минимизации функционала

$$P(\alpha) = \int Q(z, \alpha) dP(z),$$

определяющего среднюю величину потерь.

Один из путей решения этой задачи связан с идеей замены неизвестного функционала

$$P(\alpha) = \int Q(z, \alpha) dP(z)$$

функцией

$$P_{\text{эмп}}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} Q(z_i, \alpha),$$

построенной по случайной и независимой обучающей выборке $z_1, z_2, \ldots, z_l$.

Функция $P_{\text{эмп}}$ получила название функции, исчисляющей величину эмпирического риска. Для каждого фиксированного значения параметра α она определяет среднюю величину потерь на выборке $z_1, z_2, \ldots, z_l$.

Идея метода состоит в том, чтобы найти значение параметра $\alpha = \alpha^*_{\text{э}}$, обеспечивающего минимум функции эмпирического риска, а затем в качестве решения задачи о минимизации среднего риска предложить функцию с этим значением параметра.

Возникают вопросы, для каких функций $Q(z, \alpha)$ такая подмена возможна, и какая при этом совершается ошибка?

Мы ответим на эти вопросы для случая, когда множество решающих правил конечно.

Теорема 5. *Если из множества, состоящего из N решающих правил, среди которых есть правило, которое не ошибается ни*

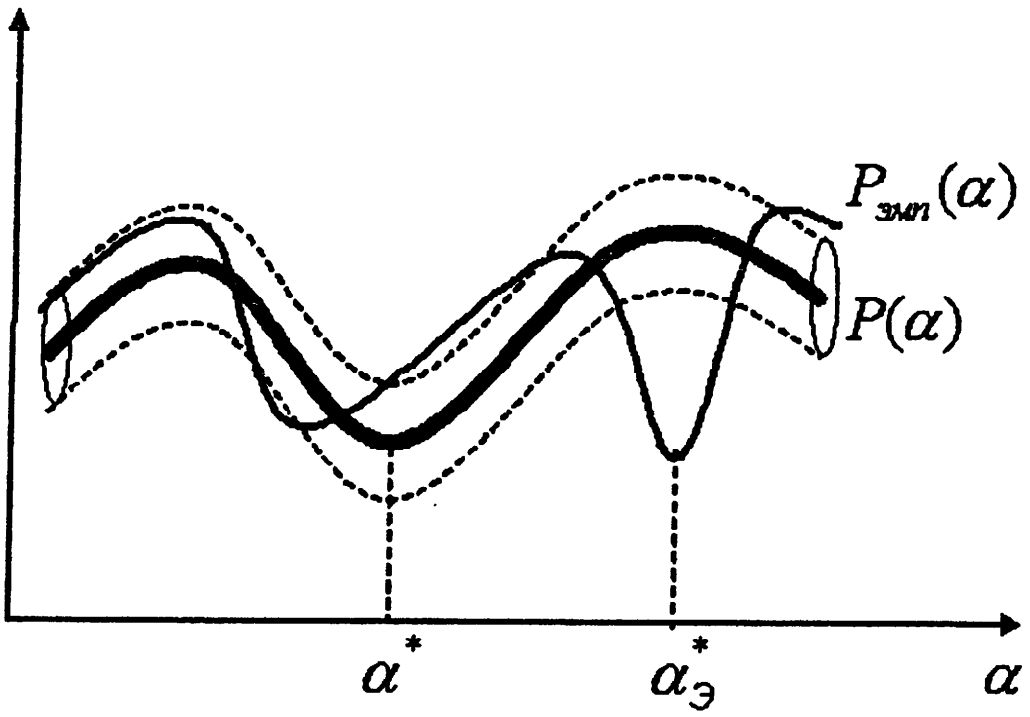

Рис. 1.10:

для какого объекта x, выбирается такое правило, которое на обучающей последовательности не совершает ни одной ошибки, то с вероятностью $1-\eta$ можно утверждать, что вероятность ошибочной классификации с помощью выбранного правила составит величину, меньшую ε, если длина обучающей последовательности не меньше

$$l = \left]\frac{\ln N - \ln \eta}{-\ln(1-\varepsilon)}\right[.$$

Доказательство. Согласно классическим теоремам теории вероятностей частота появления любого события сходится к вероятности этого события при неограниченном увеличении числа испытаний. Однако из этих теорем никак не следует, что решающее правило $F(x, \alpha^*)$, которое имеет минимальную частоту ошибок $P_{\text{эмп}}(\alpha)$, будет иметь минимальную или близкую к минимальной вероятность ошибки.

Предположим для наглядности, что решающие правила $F(x, \alpha)$ задаются скаляром α, который может принимать значения от 0 до 1. Каждому значению α ставится в соответствие решающее правило, для которого существует вероятность ошибки

$P(\alpha)$. На рисунке 1.10 функция $P(\alpha)$ изображена жирной линией. Наряду с этой функцией может быть построена и функция $P_{\text{эмп}}(\alpha)$, которая для каждого α определяет частоту ошибочной классификации с помощью правила $F(x, \alpha)$, вычисленную на обучающей последовательности.

Метод минимизации эмпирического риска предлагает по минимуму функции $P_{\text{эмп}}(\alpha)$ судить о минимуме функции $P(\alpha)$. Для того чтобы по точке минимума и минимальному значению функции $P_{\text{эмп}}(\alpha)$ можно было судить о точке минимума функции $P(\alpha)$ и ее минимальном значении, достаточно, чтобы кривая $P_{\text{эмп}}(\alpha)$ находилась внутри ε-трубки кривой $P(\alpha)$. Напротив, выброс хотя бы в одной точке (как на рисунке 1.10) может привести к тому, что в качестве точки минимума $P(\alpha)$ будет выбрана точка выброса. В этом случае минимум $P_{\text{эмп}}(\alpha)$ никак не характеризует минимум функции $P(\alpha)$.

Это означает, что нас интересуют не классические условия, когда для любых α и ε имеет место

$$P\{|P_{\text{эмп}}(\alpha) - P(\alpha)| > \varepsilon\} \to 0$$

при $l \to \infty$, а более сильные условия равномерного приближения, когда для любого ε справедливо

$$P\{\sup_{\alpha} |P_{\text{эмп}}(\alpha) - P(\alpha)| > \varepsilon\} \to 0$$

при $l \to \infty$.

Итак, пусть класс решающих правил состоит из конечного числа N элементов $\{F(x, \alpha_j) : j = 1, 2, \ldots, N\}$.

Обозначим $\nu_j = P_{\text{эмп}}(\alpha_j)$, $\mu_j = P(\alpha_j)$,

$$\theta'(a) = \begin{cases} 1, & \text{если } a = 0, \\ 0, & \text{если } a > 0. \end{cases}$$

По условию теоремы среди функций

$$\{F(x, \alpha_j) : j = 1, 2, \ldots, N\}$$

есть та, которая идеально решает задачу. Следовательно, на любой выборке $x_1, \ldots, x_l$ значение минимума эмпирического риска будет равно нулю.

Однако этот минимум может достигаться на многих функциях. Поэтому возникает необходимость оценивать вероятность

того, что при выборе любой функции, доставляющей ноль величине эмпирического риска, можно гарантировать, что выбрана функция, качество которой не хуже заданного ε.

Оценка скорости равномерной сходимости частот к вероятностям по множеству правил, для которых частота ошибок равна нулю, связана с оценкой вероятности следующего события:

$$\{\sup_j |\nu_j - \mu_j| \cdot \theta'(\nu_j) > \varepsilon\}.$$

Так как число функций, на которых достигается нуль величины эмпирического риска, не превосходит N — числа всех элементов в классе, то справедливо неравенство

$$P\{\sup_j |\nu_j - \mu_j| \cdot \theta'(\nu_j) > \varepsilon\} < NP_\varepsilon^l, \tag{1.3}$$

где P_ε^l — вероятность того, что решающее правило, для которого вероятность совершить ошибку есть величина большая ε, правильно классифицирует все объекты обучающей последовательности. Эту вероятность легко оценить:

$$P_\varepsilon^l < (1-\varepsilon)^l.$$

Подставляя оценку P_ε^l в (1.3), получим

$$P\{\sup_j |\nu_j - \mu_j| \cdot \theta'(\nu_j) > \varepsilon\} < N(1-\varepsilon)^l.$$

Для того чтобы вероятность $P\{\sup_j |\nu_j - \mu_j| \cdot \theta'(\nu_j) > \varepsilon\}$ не превосходила η, достаточно выполнения условия

$$N(1-\varepsilon)^l \le \eta.$$

Разрешая относительно l это неравенство, получим

$$l \ge \frac{\ln N - \ln \eta}{-\ln(1-\varepsilon)}.$$

Теорема доказана. □

Упражнения

1.23. Образы S_1 и S_2 заданы нормальными распределениями $N(0,1)$ и $N(1,1)$. Вычислить вероятность ошибки распознавания по правилу Байеса при условии, что априорное распределение вероятностей появления образов S_1 и S_2 таково:

а) $\{1/2, 1/2\}$;

б) $\{1/3, 2/3\}$.

1.24. Образы S_1, S_2 и S_3 заданы равномерными распределениями на отрезках $[0; 1/2]$, $[1/3; 2/3]$ и $[0; 2]$ соответственно. Вычислить вероятность ошибки распознавания по правилу Байеса, если априорное распределение вероятностей появления образов S_1, S_2 и S_3 таково: $\{1/2; 1/4; 1/4\}$.

1.25. Доказать, что если задано два нормальных распределения с одинаковыми ковариационными матрицами, отличающимися только векторами средних, то байесовской границей будет гиперплоскость.

1.26. Доказать, что если два нормальных распределения имеют одинаковые ковариационные матрицы и эти матрицы диагональны, то алгоритм максимального правдоподобия (метод Байеса) эквивалентен методу эталонов.

1.27. Величина из класса S_1 имеет нормальное распределение N_1, а из S_2 — N_2. Вероятность события S_1 равна P_1, а S_2 — P_2. Цена ошибки первого рода (т.е. событие S_1 распознать как S_2) равна C_1, а цена ошибки второго рода — C_2. Определить решающее правило и выразить математическое ожидание цены ошибки через $ERF(x) = \frac{1}{\sqrt{2\pi}} \int_0^x \exp(-t^2)dt$ при условии что

а) $N_1 = N(0, 1)$, $N_2 = N(-2, 4)$, $P_1 = 0.1$, $P_2 = 0.9$, $C_1 = 90$, $C_2 = 20$;

б) $N_1 = N(-1, 1)$, $N_2 = N(1, 4)$, $P_1 = 0.1$, $P_2 = 0.9$, $C_1 = 18$, $C_2 = 4$.

1.4 Тестовый подход к распознаванию

В статистическом подходе, рассматриваемом в предыдущем разделе, говорится, что при наличии достаточно большой обучающей выборки мы можем с большой вероятностью найти хорошее решающее правило. Но в жизни часто бывают ситуации, когда невозможно получить обучающую выборку нужной длины. Например, в задачах медицинской диагностики длина обучающей выборки не может быть больше, чем число случае с точно установленными диагнозами, а их может оказаться недостаточно. Еще более плохая ситуация с геологическими задачами оценки запасов руды. Число разведанных месторождений с их описаниями в самом лучшем случае исчисляется десятками, что никоим образом не может обеспечить надежность статистических

методов. Геометрические методы в этом случае тоже не могут быть использованы, потому что описания объектов представляют собой векторы большой размерности, которые, как правило, не могут быть разделены простыми поверхностями.

В этих случаях, когда число элементов в обучающей выборке мало, а размерность пространства признаков велика, когда ни геометрический, ни статистический подходы не могут дать результата, может быть использован тестовый подход.

1.4.1 Понятие теста

Тестовый подход связан с понятие теста, предложенного С.В.Яблонским и И.А.Чегис [62, 63].

Пусть имеется множество объектов $M = \{x_1, \ldots, x_m\}$, и каждый из объектов описывается в заданной системе признаков наборами значений этих признаков, или, иными словами, задается вектором признакового пространства. Тогда множество M может быть описано матрицей

$$T_{m,n} = \begin{pmatrix} x_1(1) & x_1(2) & \ldots & x_1(n) \\ x_2(1) & x_2(2) & \ldots & x_2(n) \\ . & . & \ldots & . \\ x_m(1) & x_m(2) & \ldots & x_m(n) \end{pmatrix},$$

где через $x_i(j)$ обозначено значение j-го признака объекта x_i, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$. Пусть M разбито на два класса K_1 и K_2, а тем самым и матрица $T_{m,n}$ состоит из двух подматриц T_{K_1} и T_{K_2}. Классы K_1 и K_2 нам неизвестны, но нам дана некоторая обучающая выборка. Пусть T_1 и T_2 — подматрицы матриц T_{K_1} и T_{K_2}, соответствующие элементам обучающей выборки. Требуется для любого набора признаков, который есть описание объекта из M определить, к какому классу он относится.

Будем считать, что значения признаков $1, 2, \ldots, n$ принадлежат множеству $\{0, 1\}$, и эти признаки выбраны так, что в матрице $T_{m,n}$ нет двух одинаковых строк, отвечающих объектам из разных классов.

Если T — некоторая матрица, а τ — некоторый набор признаков (набор столбцов), то через $T(\tau)$ обозначим подматрицу

матрицы T, полученную удалением всех столбцов кроме столбцов из τ.

Набор столбцов τ назовем *тестом*, если не существует строки, содержащейся в $T_1(\tau)$ и $T_2(\tau)$ одновременно. Тем самым, тест — это множество признаков, которое позволяет отличить на обучающей выборке классы K_1 и K_2.

Будем говорить, что τ — *тупиковый тест*, если τ — тест, а любой его поднабор не является тестом. Иными словами, тупиковый тест — минимальное по вложению множество признаков, которое все еще может отличать классы K_1 и K_2 на обучающей выборке.

Идея тестового подхода основывается эвристике голосования теста. Если дан объект x, который надо распознать, и некоторый тест τ, то отбрасываются все признаки, которые не входят в тест. Если x на множестве оставшихся признаков больше похож на элементы обучающей выборки из первого класса, то говорят, что тест τ голосует за первый класс, иначе — за второй. Далее выбирается некоторое опорное множество тестов, в качестве которого можно взять множество всех тестов, множество тупиковых тестов, или множество тестов фиксированной длины, и вычисляется, за что интегрально голосует опорное множество.

Тем самым, поскольку не хватает длины обучающего множества, чтобы говорить о статистически надежных результатах, то "статистика" набирается на множестве признаков, пробегая по опорному множеству тестов.

1.4.2 Линейные тестовые алгоритмы распознавания

Алгоритм A_1 [14], как и многие другие, основывается на понятии информационного веса признака, предложенного Ю.И.Журавлевым.

Для пары матриц T_1 и T_2 *информационным весом* p_i *признака* i, $i \in \{1, 2, \ldots, n\}$, будем называть отношение числа тупиковых тестов, в которые вошел признак i, к общему числу тупиковых тестов.

Чем больше вес p_i, тем важнее признак i с точки зрения

различения классов K_1 и K_2.

Вектор $p = (p_1, \ldots, p_n)$ назовем *вектором информационных весов признаков* или *весовым вектором*.

Сначала в алгоритме A_1 для каждого признака i вычисляется его информационный вес p_i, $i \in \{1, 2, \ldots, n\}$.

Если

$$T_1 = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ . & . & \ldots & . \\ a_{m_1 1} & a_{m_1 2} & \ldots & a_{m_1 n} \end{pmatrix},$$

$$T_2 = \begin{pmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ . & . & \ldots & . \\ b_{m_2 1} & b_{m_2 2} & \ldots & b_{m_2 n} \end{pmatrix}$$

и $x = (x_1, x_2, \ldots, x_n)$ — объект, подлежащий классификации, то вычисляются две величины

$$r_1 = \frac{1}{m_1} \sum_{i=1}^{m_1} \sum_{j=1}^{n} p_j (a_{ij} \oplus x_j) \tag{1.4}$$

и

$$r_2 = \frac{1}{m_2} \sum_{i=1}^{m_2} \sum_{j=1}^{n} p_j (b_{ij} \oplus x_j), \tag{1.5}$$

где операция $\oplus$ — есть сумма по модулю 2, а $\sum$ — обычная сумма.

Теперь, если $r_1 > r_2$, то x следует отнести к классу K_2, и если $r_1 < r_2$, то x следует отнести к классу K_1. Описанное решающее правило разбивает множество объектов M на две части, разделяемые гиперплоскостью $r_1 = r_2$.

В алгоритме A_2 [29] решающее правило выглядит следующим образом. Для каждой строки $s = (s_1, \ldots, , s_n)$ матриц T_1 и T_2 находим скалярное произведение

$$s \cdot p = \sum_{i=1}^{n} s_i \cdot p_i,$$

называемое весом строки.

Пусть q — вес объекта $x = (x_1, \ldots, x_n)$, подлежащего классификации. Тогда, если $q \geq h$, то отнесем x к классу K_1, а иначе — к классу K_2. Значение h подбирается с учетом требования, чтобы наибольшее число строк матриц T_1 и T_2 классифицировались правильно. Разделяющей поверхностью в алгоритме A_2 является гиперплоскость, заданная уравнением

$$\sum_{i=1}^{n} x_i \cdot p_i = h.$$

Алгоритм A_3 [47, 48] является модификацией алгоритма A_2 [29]. Пусть $a_1, a_2, \ldots, a_{m_1}$ — веса строк матрицы T_1, $b_1, b_2, \ldots, b_{m_2}$ — веса строк матрицы T_2, $z = \min_i a_i - \max_j b_j$. Для некоторого признака i заменим его на его отрицание. При этом в таблицах T_1 и T_2 i-ый столбец инвертируется. Такая замена не влияет на информационные веса признаков $p = (p_1, \ldots, p_n)$ и сохраняет все тупиковые тесты. Но при этом изменяются величины $a_1, a_2, \ldots, a_{m_1}$ и $b_1, b_2, \ldots, b_{m_2}$, и величина z может увеличиться. Если это произошло мы можем принять это инвертирование i-го признака, и перейти к рассмотрению следующего признака. При $z > 0$ тестовый алгоритм A_3 правильно классифицирует всю обучающую выборку, и в качестве пороговой величины h можно взять

$$h = (\min_i a_i + \max_j b_j)/2.$$

1.4.3 Алгоритм Кудрявцева голосования по тестам

Пусть $\{1, 2, \ldots, n\}$ — множество признаков. Договоримся подмножество $\tau \subset \{1, 2, \ldots, n\}$ обозначать булевским вектором $t = (t_1, \ldots, t_n)$, подразумевая, что признак $i \in \tau$ точно тогда, когда $t_i = 1$.

Пусть как и ранее $T_1 = \{a_j = (a_{j1}, \ldots, a_{jn}) : j = 1, 2, \ldots, m_1\}$, $T_2 = \{b_j = (b_{j1}, \ldots, b_{jn}) : j = 1, 2, \ldots, m_2\}$ — множества элементов обучающей выборки, принадлежащие классам K_1 и K_2 соответственно.

Обозначим $\mathcal{T} = \mathcal{T}(T_1, T_2)$ — опорное множество тестов для обучающей выборки T_1, T_2. В качестве опорного множества $\mathcal{T}$

могут выступать, например, множество всех тестов, множество тупиковых тестов, или множество тестов фиксированной длины.

Если $a = (a_1, \ldots, a_n)$ — элемент обучающей выборки, $t = (t_1, \ldots, t_n) \in \mathcal{T}$ — некоторый тест, а $x = (x_1, \ldots, x_n)$ — объект, подлежащий классификации, то скажем, что *тест t голосует за элемент a на объекте x*, если

$$\gamma_t^a(x) = \prod_{i=1}^{n}(1 - t_i|x_i - a_i|) = 1,$$

т.е. на признаках из теста t объекты a и x совпадают.

Тогда сумму

$$\sum_{a \in T_j} \gamma_t^a(x)$$

можно интерпретировать как интеграл голосования множества T_j, $j = 1, 2$, а многочлен

$$\Gamma_j(x) = \frac{1}{m_j} \sum_{t \in \mathcal{T}} \sum_{a \in T_j} \gamma_t^a(x)$$

есть интегральный голос всего опорного множества тестов $\mathcal{T}$ за класс K_j, $j = 1, 2$.

Многочлен $R(x) = \Gamma_2(x) - \Gamma_1(x)$ называется *многочленом голосования.*

Алгоритм голосования по тестам, предложенный В.Б.Кудрявцевым, состоит в том, что на объекте x, подлежащем классификации, вычисляется многочлен голосования $R(x)$, и если $R(x) \leq 0$, то x относится к классу K_1, и к классу K_2 — в противном случае.

Рассмотрим многочлен $\Gamma_1(x)$. Раскрывая скобки в произведениях, получим

$$\Gamma_1(x) = \frac{1}{m_1} \sum_{\tau \subseteq \{1,\ldots,n\}} (-1)^{|\tau|} \Big(\sum_{t \in \mathcal{T}} \prod_{i \in \tau} t_i \Big) \Big(\sum_{j=0}^{m_1} \prod_{i \in \tau} |x_i - a_{ji}| \Big).$$

Возьмем линейную часть этого многочлена по переменным $z_{ji} = |x_i - a_{ji}|$:

$$l_1(x) = \frac{1}{m_1} \sum_{i=1}^{n} (-1) \Big(\sum_{t \in \mathcal{T}} t_i \Big) \Big(\sum_{j=1}^{m_1} |x_i - a_{ji}| \Big).$$

Тогда линейная часть многочлена голосования будет

$$l(x) = l_2(x) - l_1(x) = \frac{1}{m_1}\sum_{i=1}^{n}\Big(\sum_{t\in T} t_i\Big)\Big(\sum_{j=1}^{m_1}|x_i - a_{ji}|\Big) -$$

$$-\frac{1}{m_2}\sum_{i=1}^{n}\Big(\sum_{t\in T} t_i\Big)\Big(\sum_{j=1}^{m_2}|x_i - b_{ji}|\Big) =$$

$$= |T|\sum_{i=1}^{n}\Big(\frac{1}{|T|}\sum_{t\in T} t_i\Big)\cdot$$

$$\cdot\Big(\frac{1}{m_1}\sum_{j=1}^{m_1}|x_i - a_{ji}| - \frac{1}{m_2}\sum_{j=1}^{m_2}|x_i - b_{ji}|\Big) =$$

$$= |T|\sum_{i=1}^{n} p_i \cdot \Big(\frac{1}{m_1}\sum_{j=1}^{m_1} x_i \oplus a_{ji} - \frac{1}{m_2}\sum_{j=1}^{m_2} x_i \oplus b_{ji}\Big).$$

Сравнивая полученное с (1.4) и (1.5), можем заметить, что известный алгоритм A_1 является линейным приближением многочлена голосования.

Алгоритм голосования по тестам предполагает умение перечислять все тупиковые тесты. В работах [3, 5, 15, 16] приводятся асимптотически оптимальные алгоритмы перечисления всех тестов.

В работе [20] показано, что если в качестве опорного множества брать множество "коротких" тестов (тестов, мощность которых порядка $\log n$), то алгоритм голосования становится не только значительно менее трудоемким, но и более надежным.

С помощью тестовых алгоритмов были успешно решены многие практические задачи распознавания, в частности геологические задачи по оценке запасов руды и подобные им [26, 28, 29, 34, 64].

Так, исследование ртутных месторождений позволило дать практические рекомендации о необходимости постановки геолого-разведочных работ по некоторым малоизученным месторождениям Северо-востока СССР; по материалам Приморского геологического управления о месторождениях олова Приморья были получены высокие оценки масштабов оруднения некоторых малоизученных месторождений, что в дальнейшем подтвердилось при проверке разведывательными работами. На двух ме-

сторождениях (Южном и Смирновском) было установлено, что рудные тела, сравнительно бедные оловом на поверхности, на глубине переходят в богатые руды с высоким содержанием оловянного камня.

1.4.4 Теорема Анселя о числе монотонных функций

Обозначим $E = \{0, 1\}$.

Для $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in E^n$ будем писать $\alpha \leq \beta$, если $\alpha_i \leq \beta_i$ для всех $i = 1, \ldots, n$. Пишем $(\alpha_1, \ldots, \alpha_n) < (\beta_1, \ldots, \beta_n)$, если $(\alpha_1, \ldots, \alpha_n) \leq (\beta_1, \ldots, \beta_n)$ и существует такой номер $i \in \{1, \ldots, n\}$, что $\alpha_i < \beta_i$.

Как и ранее будем описывать тест с помощью булевого вектора $t = (t_1, t_2, \ldots, t_n) \in E^n$.

Пусть $\mathcal{T} = \mathcal{T}(T_1, T_2)$ — множество всех тестов для обучающей выборки T_1, T_2.

Мы хотим понять как устроено множество $\mathcal{T}$ и сколько различных множеств $\mathcal{T}$ может быть получено, если варировать обучающую выборку, т.е. хотим оценить мощность следующего множества

$$\mathfrak{T}(n) = \{\mathcal{T}(T_1, T_2) : T_1, T_2 \subseteq E^n, T_1 \cap T_2 = \emptyset\}.$$

Сопоставим множеству $\mathcal{T}$ его характеристическую функцию $f_{\mathcal{T}}(t) : E^n \to E$ такую, что

$$f_{\mathcal{T}}(t) = 1 \Leftrightarrow t \in \mathcal{T}.$$

Из очевидного свойства, что если некоторое множество признаков есть тест, то любое множество, содержащее данное множество, также является тестом, следует, что функция $f_{\mathcal{T}}(t)$ является монотонной функцией.

Напомним, что булева функция $f(x_1, \ldots, x_n)$ называется *монотонной*, если для любых наборов $(\alpha_1, \ldots, \alpha_n)$ и $(\beta_1, \ldots, \beta_n)$ таких, что $\alpha_i \leq \beta_i$, $i = 1, \ldots, n$, имеет место соотношение

$$f(\alpha_1, \ldots, \alpha_n) \leq f(\beta_1, \ldots, \beta_n).$$

Отсюда следует, что число $|\mathfrak{T}(n)|$ различных множеств $\mathcal{T}$ всех тестов для некоторых обучающих выборок над множеством признаков $\{1, 2, \ldots, n\}$ оценивается сверху с числом монотонных функций от n переменных.

Набор $(\alpha_1, \ldots, \alpha_n)$ называется *единицей (нулем)* монотонной функции f, если

$$f(\alpha_1, \ldots, \alpha_n) = 1$$

(соответственно $f(\alpha_1, \ldots, \alpha_n) = 0$).

Единица (нуль) монотонной функции f $(\alpha_1, \ldots, \alpha_n)$ называется *нижней единицей (верхним нулем)* f, если для любого $(\beta_1, \ldots, \beta_n)$, такого, что $(\beta_1, \ldots, \beta_n) < (\alpha_1, \ldots, \alpha_n)$, выполняется $f(\beta_1, \ldots, \beta_n) = 0$ (соответственно, для любого $(\beta_1, \ldots, \beta_n)$, такого, что $(\alpha_1, \ldots, \alpha_n) < (\beta_1, \ldots, \beta_n)$, выполняется $f(\beta_1, \ldots, \beta_n) = 1$).

С другой стороны справедливо утверждение.

Лемма 3. *Для любой монотонной булевой функции $f \not\equiv 0$ существует обучающая выборка T_1, T_2, для которой*

$$f_{\mathcal{T}(T_1,T_2)} = f.$$

Доказательство. Пусть $\{b_1, \ldots, b_k\} \subseteq E^n$ — множество верхних нулей функции f. Положим $T_1 = \{a\}$, где $a = (a_1, \ldots, a_n) = (1, \ldots, 1)$ — единичный набор, $T_2 = \{b_1, \ldots, b_k\}$.

Покажем, что $f_{\mathcal{T}(T_1,T_2)} = f$.

Рассмотрим произвольный набор $t = (t_1, \ldots, t_n)$, на котором $f(t_1, \ldots, t_n) = 1$, и произвольный верхний ноль

$$b_i = (b_{i1}, \ldots, b_{in}) \in T_2.$$

Так как $f(b_{i1}, \ldots, b_{in}) = 0$, то наборы t и b_i либо несравнимы, либо $b < t$. Следовательно, существует номер $j \in \{1, \ldots, n\}$ такой, что $t_j > b_{ij}$, т.е. $t_j = a_j = 1$, $b_{ij} = 0$. Откуда в силу произвольности b_i следует, что t — тест для T_1, T_2 и $f_{\mathcal{T}(T_1,T_2)}(t) = 1$.

Рассмотрим произвольный набор $t = (t_1, \ldots, t_n)$, на котором $f(t_1, \ldots, t_n) = 0$. Тогда существует верхний ноль

$$b_i = (b_{i1}, \ldots, b_{in}) \in T_2,$$

такой, что $t \leq b_i$. Это означает, что для любого такого номера $j \in \{1, 2, \ldots, n\}$, что $t_j = 1$, выполнено $b_{ij} = 1 = a_j$. Следовательно, на множестве t наборы b_i и a совпадают, и, значит, t — не тест для T_1, T_2 и $f_{\mathcal{T}(T_1,T_2)}(t) = 0$.

Лемма доказана. □

Из леммы сразу следует, что $|\Im(n)|$ равно числу монотонных функций от n переменных.

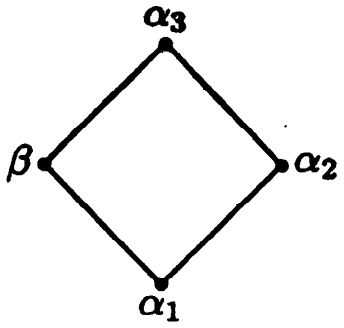

Рис. 1.11:

Задача определения числа $\psi(n)$ монотонных булевых функций от n переменных была поставлена Дедекиндом в 1897 г. [76] и была решена им для $n = 4$.

Здесь мы покажем, что

$$2^{C_n^{[n/2]}} \leq \psi(n) \leq 3^{C_n^{[n/2]}},$$

где $[n/2]$ — наибольшее целое, меньшее или равное $n/2$, C_n^m — число сочетаний из n элементов по m, причем считается, что $C_n^{-1} = 0$. Нижняя оценка в этих неравенствах была получена Э.Н.Гильбертом [13], а верхняя — Ж.Анселем [6]. Отметим, что Коршуновым А.Д. [31] получена асимптотика числа $\psi(n)$ при $n \to \infty$.

Последовательность $\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(m)}$ элементов из E^n называется *цепью*, если $\beta^{(i+1)}$ получается из $\beta^{(i)}$ заменой одного нуля (в наборе координат) на единицу, $i = 1, 2, \ldots, m-1$. Тем самым, $\beta^{(1)} < \beta^{(2)} < \ldots < \beta^{(m)}$.

Если три элемента $\alpha_1, \alpha_2, \alpha_3$ из E^n образуют цепь, то четвертый элемент β, образующий вместе с ними квадрат (см. рис. 1.11), называют дополнением цепи $\alpha_1, \alpha_2, \alpha_3$ до квадрата.

Лемма 4 (Анселя). *Единичный n-мерный куб E^n может быть покрыт множеством из $C_n^{[n/2]}$ попарно непересекающихся цепей, обладающих следующими свойствами:*

а) число цепей длины $n - 2p + 1$ равно $C_n^p - C_n^{p-1}$ $(0 \leq p \leq [n/2])$, и минимальный элемент каждой такой цепи есть набор с p единицами и $n - p$ нулями, а максимальный — с p нулями и $n - p$ единицами;

б) если заданы три элемента $\alpha_1, \alpha_2, \alpha_3$, образующие цепь и принадлежащие одной и той же цепи длины $n - 2p + 1$,

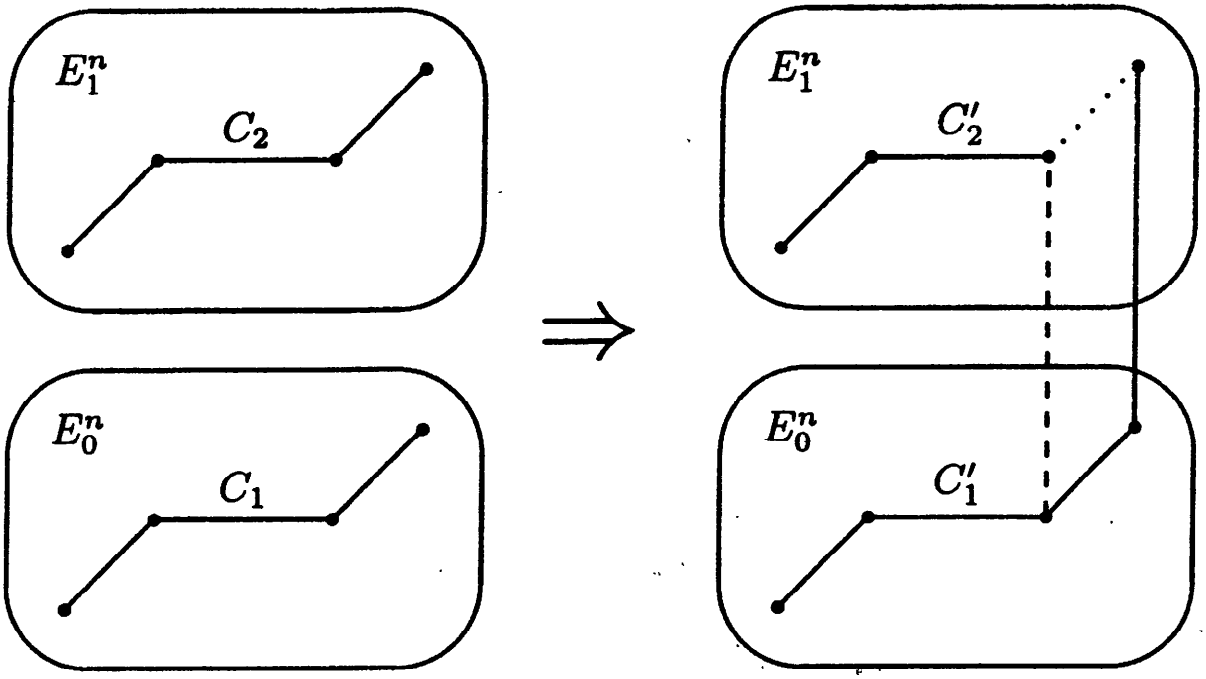

Рис. 1.12:

то дополнение цепи $\alpha_1, \alpha_2, \alpha_3$ до квадрата принадлежит цепи длины $n - 2p - 1$.

Доказательство. Доказывать лемму будем индукцией по n.

Базис индукции. Для $n = 1, 2$ лемма верна.

Индуктивный переход.

1) Куб E^n делится на два подмножества E_0^n и E_1^n (минимальное и максимальное), изоморфные кубу E^{n-1} и полученные соответственно присоединением 0 и 1 слева к координатам куба E^{n-1}. В предположении, что лемма верна для куба E^{n-1}, мы покроем E^n множеством цепей определяемых следующим образом:

1° Пусть E_0^n и E_1^n покрыты каждое множеством $C_{n-1}^{[(n-1)/2]}$ цепей, обладающих свойствами указанными в формулировке леммы (вторая часть свойства а) — без учета добавления разряда).

2° Пусть C_1 — одна из цепей E_0^n. Пусть C_2 — цепь, изоморфная C_1 в E_1^n, и пусть y_2 — ее наибольший элемент. Мы продолжим C_1, добавив к ней y_2 (см. рис. 1.12).

3° Отнимем у цепи C_2 ее наибольший элемент y_2. После осуществления преобразований 2° и 3° для новых цепей будет выполнена вторая часть свойства а).

4° Произведем те же самые операции со всеми цепями из

E_0^n.

Тогда цепи, покрывающие E^n, не будут пересекаться и число цепей длины $n-2p+1$ будет равно

$$(C_{n-1}^p - C_{n-1}^{p-1}) + (C_{n-1}^{p-1} - C_{n-1}^{p-2}) = (C_n^p - C_n^{p-1}).$$

2) Свойство б) леммы является непосредственным следствием вышеприведенной конструкции.

В самом деле, цепи в E^n — двух сортов: "удлиненные", возникшие из цепей E_0^{n-1}, и "укороченные", возникшие из цепей E_1^{n-1}. Рассмотрим возможные случаи для трех элементов α_1, α_2, α_3, образующих цепь и лежащих на построенной цепи C в E^n (длины $n-2p+1$).

I. C — удлиненная цепь, и α_3 не есть ее максимальный элемент. В этом случае $\alpha_1, \alpha_2, \alpha_3$ лежат на некоторой "старой" цепи в E_0^{n-1} (длины $n-2p$). Дополнение цепи $\alpha_1, \alpha_2, \alpha_3$ до квадрата находится на некоторой цепи C' (длины $n-2p-2$) в E_0^{n-1}, которая будет продолжена при переходе к E^n до цепи длины $n-2p-1$.

II. C — удлиненная цепь, и α_3 — ее максимальный элемент. Пусть $C = C_1'$ (рис 1.12) возникла в результате удлинения цепи C_1 из E_0^{n-1}; C_2 — цепь в E_1^{n-1}, изоморфная C_1; C_2' — укороченная цепь, возникшая из C_2 (C_2' имеет длину $n-2p-1$). Тогда дополнение цепи $\alpha_1, \alpha_2, \alpha_3$ до квадрата есть максимальный элемент цепи C_2'.

III. C — укороченная цепь. Пусть $C = C_2'$ возникла из цепи C_2 (длины $n-2p+2$) в E_1^{n-1}. Тогда, во-первых, α_3 — не максимальный элемент цепи C_2 и, во-вторых, в силу второй части свойства а) максимальный α элемент цепи C_2 имеет $p-1$ нулей. В силу свойства б) дополнение β цепи $\alpha_1, \alpha_2, \alpha_3$ до квадрата принадлежит некоторой цепи $\tilde{C}$ длины $n-2p$ в E_1^{n-1}; максимальный элемент этой цепи имеет p нулей. Так как $\beta < \alpha_3 < \alpha$, то β имеет не менее $p+1$ нулей и поэтому не является максимальным элементом цепи $\tilde{C}$. Следовательно, β принадлежит укороченной цепи длины $n-2p-1$, получающейся из $\tilde{C}$.

Лемма доказана. □

Теорема 6. *Число $\psi(n)$ монотонных булевых функций n переменных удовлетворяет неравенствам*

$$2^{C_n^{[n/2]}} \le \psi(n) \le 3^{C_n^{[n/2]}}.$$

Доказательство. *Нижняя оценка.* Понятно, что монотонная функция однозначно задается своими нижними единицами. Любая пара нижних единиц несравнима. Поэтому любое множество несравнимых наборов может быть объявлено множеством нижних единиц монотонной функции и будет ее задавать. Поэтому число монотонных функций не меньше, чем число подмножеств попарно несравнимых наборов булевого куба. Рассмотрим множество S, являющееся $[n/2]$-м слоем булевого куба E^n, т.е. S — множество всех наборов, содержащих ровно $[n/2]$ единиц. Понятно, что любая пара наборов из S попарно несравнима. Следовательно, любое подмножество S может быть использовано в качестве множества нижних единиц некоторой монотонной функции. Откуда сразу следует, что

$$\psi(n) \geq 2^{|S|} = 2^{C_n^{[n/2]}}.$$

Верхняя оценка. Рассмотрим произвольную монотонную функцию $f(x_1, \ldots, x_n)$. Свойство б) леммы 4 и монотонный характер функции $f(x_1, \ldots, x_n)$ позволяют сделать следующий вывод: если значение $f(x_1, \ldots, x_n)$ известно во всех вершинах цепей длины $n - 2p - 1$, то в каждой цепи длины $n - 2p + 1$ существует самое большее две (соседние) вершины, где значения $f(x_1, \ldots, x_n)$ остаются неопределенными. Тогда, если мы будем определять значения функции, начиная с самых коротких цепей, то в каждой цепи будет не более двух (соседних) вершин α_1, α_2, $\alpha_1 < \alpha_2$, где функция неопределена, причем в этих вершинах функция может принимать лишь следующие наборы значений: (0,0) (0,1), (1,1). В результате получаем

$$\psi(n) \leq 3^{C_n^{[n/2]}}.$$

Теорема доказана. □

1.4.5 Расшифровка монотонных функций

Известно, что в общем случае для однозначного определения функции алгебры логики необходимо знать значения функции во всех точках булевого куба. Если же функция принадлежит к некоторому классу, более узкому, чем множество всех функций алгебры логики, то для однозначного определения значений функций во всех точках E^n не обязательно знать значе-

ния функций во всех точках E^n, а иногда достаточно знать на некотором подмножестве E^n. Так для однозначного определения симметрической функции достаточно знать ее значения на n наборах, по одному с каждого слоя куба.

Задача расшифровки монотонной функции состоит в следующем. Задан оператор A_f, вычисляющий для произвольной точки $\alpha \in E^n$ значение монотонной функции $f(x)$ в этой точке α, т.е. $f(\alpha)$. Вопрос: как минимальным числом обращений к оператору A_f полностью восстановить таблицу значений монотонной функции $f(x)$?

Очевидно, что работа любого алгоритма F, решающего указанную задачу, будет заключаться в следующем. Алгоритм выбирает некоторую точку $\alpha \in E^n$ и с помощью оператора A_f вычисляет значение функции $f(x)$ в точке α. Полученное значение функции в точке α заносится в таблицу значений исследуемой функции. По монотонности определяются значения $f(x)$ в других точках E^n. Затем по некоторому правилу выбирается другая точка E^n, и процесс повторяется до тех пор, пока таблица значений не окажется заполненной полностью.

Паре — алгоритм F и монотонная функция f — сопоставляем число $\varphi(F, f)$, равное числу обращений к оператору A_f в процессе восстановления таблицы значений функции f с помощью алгоритма F.

Обозначим

$$\varphi(F, n) = \max_{f \in M_n} \varphi(F, f),$$

где M_n — множество монотонных функций n переменных;

$$\varphi(n) = \min_{F} \varphi(F, n),$$

где минимум берется по всем алгоритмам, решающим задачу расшифровки.

Пусть N — некоторый класс булевых функций n переменных и $f \in N$. Множество $G(f, N)$ точек их E^n называется *разрешающим для пары* (f, N), если из того, что функция $g \in N$, и на множестве $G(f, N)$ значения функций f и g совпадают, следует, что $f = g$.

Разрешающее множество называется *тупиковым разрешающим множеством* для пары (f, N), если никакое его собственное подмножество не является разрешающим для пары (f, N).

Легко видеть, что у каждой монотонной функции существует единственное тупиковое разрешающее множество, состоящее из всех ее верхних нулей и нижних единиц.

Теорема 7. *Имеет место*

$$\varphi(n) = C_n^{[n/2]} + C_n^{[n/2]+1}.$$

Доказательство. *Нижняя оценка* [27]. Из единственности тупикового разрешающего множества монотонной функции $f(x_1, \ldots, x_n)$ следует, что необходимо восстановить ее значение во всех точках $G(f, M_n)$. Рассмотрим функцию

$$h(x_1, \ldots, x_n) = \begin{cases} 0, & \text{если } 0 \le \sum_{i=1}^n x_i \le [n/2], \\ 1 & \text{в противном случае.} \end{cases}$$

Множество ее верхний нулей есть $[n/2]$-й слой куба E^n, а нижних единиц — $([n/2]+1)$-й слой. Откуда сразу следует, что

$$\varphi(n) \ge C_n^{[n/2]} + C_n^{[n/2]+1}.$$

Верхняя оценка [6]. Как мы уже отмечали, для произвольной монотонной функции $f(x_1, \ldots, x_n)$ из свойства б) леммы 4 следует, что если значение функции $f(x_1, \ldots, x_n)$ известно во всех вершинах цепей длины $n-2p-1$, то в каждой цепи длины $n-2p+1$ существует самое большее две (соседние) вершины, где значения $f(x_1, \ldots, x_n)$ остаются неопределенными. Рассмотрим два случая.

1) $n = 2k$. Тогда по лемме 4 имеется $C_n^k - C_n^{k-1}$ цепей длины 1, и C_n^{k-1} цепей длины более 1. Для первой группы цепей нам надо одно обращение к оператору A_f, а для второй — два. Следовательно

$$\varphi(n) \le C_n^k - C_n^{k-1} + 2C_n^{k-1} = C_n^{[n/2]} + C_n^{[n/2]+1}.$$

2) $n = 2k+1$. Тогда все C_n^k цепей имеют длину более 1, и для каждой из них требуется два обращения к оператору A_f. Следовательно

$$\varphi(n) \le 2C_n^k = C_n^{[n/2]} + C_n^{[n/2]+1}.$$

Теорема доказана. □

Упражнения

1.28. Для данной обучающей выборки, заданной парой таблиц (T_1, T_2) найти все тупиковые тесты и определить, к какому классу будет отнесен вектор (1010), если воспользоваться описанными выше тестовыми алгоритмами, а именно

а) алгоритмом A_1 [14],

б) алгоритмом A_2 [29],

в) алгоритмом A_3 [47, 48],

г) алгоритмом голосования по тестам, где в качестве опорного взято множество тупиковых тестов,

при условии, что пара таблиц (T_1, T_2) имеет вид

а)
$$\left(\begin{array}{ccccc} T_1 & 0 & 0 & 1 & 0 \\ & 1 & 1 & 1 & 1 \\ \hline T_2 & 1 & 0 & 0 & 1 \\ & 1 & 0 & 1 & 1 \end{array}\right),$$

б)
$$\left(\begin{array}{ccccc} T_1 & 1 & 0 & 1 & 1 \\ & 1 & 0 & 0 & 0 \\ & 1 & 1 & 0 & 1 \\ & 1 & 0 & 0 & 1 \\ \hline T_2 & 0 & 0 & 0 & 0 \\ & 0 & 1 & 1 & 1 \\ & 0 & 1 & 1 & 0 \\ & 0 & 0 & 0 & 1 \end{array}\right),$$

в)
$$\left(\begin{array}{ccccc} T_1 & 1 & 1 & 1 & 1 \\ & 1 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 1 \\ & 1 & 1 & 1 & 0 \\ \hline T_2 & 0 & 1 & 0 & 0 \\ & 0 & 1 & 0 & 1 \\ & 0 & 0 & 1 & 0 \\ & 0 & 1 & 1 & 0 \end{array}\right).$$

1.29. Перечислить все монотонные функции, зависящие (не обязательно существенно) от переменных x_1, x_2.

1.30. Найти число $\psi(3)$ монотонных булевых функций от 3-х переменных.

1.31. Перечислить, описанное в лемме Анселя покрытие булевого куба E^n попарно непересекающимися цепями при условии, что

а) $n = 2$,

б) $n = 3$,

в) $n = 4$.

Глава 2

Базы данных

База данных (БД) — формализованное представление информации, удобное для хранения и поиска данных в нем. Понятие БД возникло в 60-е годы 20 века и связано с развитием вычислительной техники и информатики. Тематика теории БД связана с поиском удобного представления, компактного хранения, быстрого поиска, защищенности и других свойств данных. Развитию этого направления способствовали как производственные и творческие коллективы: IBM (иерархическая модель данных), Рабочая группа по БД Ассоциации по языкам систем обработки данных — CODASYL (сетевая модель данных), исследовательская группа по системам управления БД Американского национального института стандартов — ANSI/SPARC Study Group on DataBase Management Systems (принципы проектирования БД), так и отдельные исследователи: Кодд (E.F.Codd) (реляционная модель данных), Галлейр (H.Gallaire), Минкер (J.Minker) (дедуктивные БД), Тальхайм (B.Thalheim) (моделирование семантики в БД, теория ограничений целостности), Аткинсон (M.Atkinson), Бири (C.Beeri) и др. (объектно-ориентированные БД), В.Н.Решетников (алгебраическая модель информационного поиска), Думи (A.Dumey), А.П.Ершов (методы хеширования), Бентли (J.L.Bentley), Кнут (D.E.Knuth), Ли (D.T.Lee), Маурер (W.D.Maurer), Ульман (J.D.Ullman), Шеймос (M.I.Shamos) и др. (сложность алгоритмов обработки дан-

ных), Э.Э.Гасанов (информационно-графовая модель данных, сложность алгоритмов поиска) и многие другие.

Тематика теории БД связана со следующими основными направлениями.

Первое в основном опирается на аппарат формально-логических теорий и связано с изучением качественных свойств моделей данных и их семантическим обоснованием. К этому же направлению относятся вопросы выразительности и сложности языков запросов, соответствующих моделям данных, изучение БД как теорий (например, в интуиционистской логике высшего порядка), а также *теория ограничений целостности* [82]. В ней изучаются ограничения, накладываемые предметной областью, определяющие связи между компонентами БД и описывающие поведение БД во времени. Ограничения целостности могут быть использованы в качестве языка описания семантики БД. Изучаются особого вида ограничения целостности, называемые зависимостями. Они делятся на статические (отражающие семантику всех возможных состояний БД) и динамические (описывающие поведение БД во времени, то есть корректность последовательностей ее состояний). Изучается проблема выводимости некоторой данной зависимости из заданного списка зависимостей. Выделены классы зависимостей, в которых эта проблема неразрешима и P- или NP-разрешима. При разрешимости проблемы выводимости актуальной становится задача об аксиоматизации соответствующего класса зависимостей. Известны случаи аксиоматизируемых и неаксиоматизируемых классов.

Специальное направление в теории БД составляет защищенность данных от случайного или преднамеренного доступа к ним несанкционированных пользователей [54]. Интенсивное развитие всемирной компьютерной сети Internet делает это направление особенно актуальным.

Третье основное направление тематики теории БД связано с вопросами сложности алгоритмов обработки данных.

Это направление связано с проблематикой физической организации баз данных. При физической организации баз данных мы имеем дело не с представлением данных в прикладных программах, а с их размещением на запоминающих устройствах.

Критерии, определяющие выбор физической организации,

отличаются от тех, которые определяют выбор логической организации данных. При выборе физической организации решающим фактором является эффективность, причем согласно Дж. Мартину [40] на первом месте стоит обеспечение эффективности поиска, далее идут эффективность операций занесения и удаления и затем обеспечение компактности данных.

Методы физической организации данных хорошо изложены в классических книгах Д. Кнута [23], Дж. Мартина [40], А. Ахо, Дж. Хопкрофта, Дж. Ульмана [7].

Основную рекомендуемую литературу к данной главе составляют [11, 41].

2.1 Модели логической организации данных

Основные виды представления (модели) данных сформировались под влиянием практики с использованием средств математики.

В основу *реляционной модели данных* [41, 73, 74] положено понятие отношения. Подмножество $R \subseteq D_1 \times D_2 \times \cdots \times D_n$ есть отношение арности n с доменами $D_1, D_2, \ldots, D_n$. Рассматриваемые отношения, как правило, конечные, поэтому удобно представлять их в виде плоских таблиц. Строки таблицы называются кортежами, а столбцы — атрибутами. Подмножество атрибутов отношения называется ключом, если проекция таблицы на это множество состоит из разных строк, но удаление любого атрибута из ключа нарушает это свойство. Понятие ключа соответствует тупиковому тесту, поэтому результаты, полученные в теории тестов (А.Е.Андреев [3, 4, 5], А.Д.Коршунов [30], В.Н.Носков [44, 45, 46], Г.Р.Погосян [49] и др.) могут быть использованы для оценки мощности ключа и минимального числа атрибутов в ключе. Оценивались мощности ключей в случайных БД (таблицах) (О.Б.Селезнев, Б.Тальхайм [55] и др.). Реляционная БД — это множество таблиц, обогащенное операциями объединения, пересечения, разности, декартова произведения, проекции, соединения, селекции, частного и др. Изучение этих алгебр отношений составляет содержание теории реляци-

онных БД. Ассоциируемый с *табличным представлением* способ хранения данных не компактен, а способ поиска — полный перебор. В связи с этим в современных реляционных БД реляционная модель используется только как внешнее представление, то есть представление пользователя, тогда как внутреннее представление данных, то есть представление на машинных носителях, — принципиально другое. Простота представления при сохранении всех функциональных возможностей БД делают реляционную модель удобной для изучения качественных свойств и характеристик БД. Наглядность и простота понимания — причина популярности этой модели среди большинства пользователей.

Если в табличном представлении произвести лексикографическое упорядочение и сделать склейку по совпадающим префиксам, то получается *древовидное представление данных*, в основе которого лежит понятие дерева. Такое представление ведет к большей компактности данных и к ускорению поиска нужных данных. Такие древовидные структуры, как бинарные деревья, 2-3-деревья, B-деревья, сортирующие деревья [7] используются для внутреннего представления данных. Древовидное представление удобно использовать в лингвистических и подобных им БД, например, когда надо найти то или иное слово. В случае поиска множества однокоренных слов, то есть слов с заданной средней частью, древовидное представление не очень удобно. Древовидное представление все еще достаточно просто для понимания, хотя и не так наглядно как табличное. При использовании *древовидной (или иерархической) модели данных* как внешнего представления данных предполагаются иерархические отношения между данными, то есть отношения типа родитель-потомки, когда у каждого объекта только один родитель, но может быть несколько потомков. Такие отношения принято изображать в виде дерева, где ребро между объектами отображает наличие некоторого отношения, причем название отношения пишется на ребре. Например, между объектами "клиент" и "заказ" может быть отношение, которое называется "делает", а между "заказ" и "товары" — отношение "состоит из". Одной из систем, использующих иерархическую модель данных, является система IMS фирмы IBM [75, 78].

Обобщение понятия дерева до графа аналогично переходу от древовидного к *сетевому представлению данных*. При векторном виде данных в древовидном представлении склейка данных может быть сделана только по начальному отрезку; в сетевом представлении она допустима по любым отрезкам. В *сетевой модели данных* доступ к данным может быть осуществлен по многим путям. Она позволяет реализовать более широкий класс отношений между объектами, чем древовидная. Эта модель данных развивается ассоциацией CODASYL [72]. Поскольку сетевая модель является обобщением древовидной, то она предоставляет больше возможностей как для описания предметной области, представляемой БД, так и для нахождения оптимальных решений хранения и поиска данных. Но использование сетевой модели требует высокой квалификации от разработчика и поэтому она не была воспринята массовым пользователем.

В *объектно-ориентированной модели данных* [65], опирающейся на принципы объектно-ориентированного программирования, каждый объект представляется как черный ящик, доступ к данным которого осуществляется только через специальные функции, прилагаемые к нему. Из таких ящиков строятся более сложные объекты, которые в свою очередь могут служить новыми ящиками для построения объектов следующего уровня сложности и т.д. Главное преимущество объектно-ориентированной модели — ее технологичность, в связи с чем это одна из наиболее развивающихся моделей сегодня.

Под влиянием математической логики строится *дедуктивная модель данных* [81]. Данные в дедуктивных базах данных рассматриваются как аксиомы, а новые данные получаются из аксиом путем логического вывода. Преимущество этой модели — компактность начального множества данных (все зашито в аксиомы и правила вывода) и потенциальная бесконечность множества выводимых фактов. Недостаток — большие ресурсы по времени и памяти, необходимые для процесса вывода. Дедуктивные базы данных удобно использовать в системах принятия решений, когда заранее не очерчена область возможных ситуаций.

Работа с БД предполагает создание удобных языков — *язы-*

ков манипулирования данными, — примеры которых доставляют формальные языки логики и алгебры. В алгебраических языках манипулирования данными запрос к БД определяет последовательность операций, которые приведут к ответу. Такими языками являются ISDL и АСТРИД [41, 57]. В языках манипулирования данными, основанных на исчислении предикатов, запрос к БД соответствует формуле некоторой формально-логической теории, а ответом является множество объектов из области интерпретации, на котором истинна формула, соответствующая запросу. Такими языками являются QUEL, SQL, QBE [41, 57].

2.2 Реляционная модель данных

Если $D_1, D_2, \ldots, D_n$ — некоторые множества, то подмножество $R \subseteq D_1 \times D_2 \times \cdots \times D_n$ называется *отношением арности* n. В теории реляционных баз данных элементы $r = (r_1, r_2, \ldots, r_n)$ отношения R принято называть *кортежами*, компоненты отношения именовать символами $A_1, A_2, \ldots, A_n$ и называть *атрибутами*, а область D_i значений атрибута A_i $(i = 1, 2, \ldots, n)$ называть *доменом*. *Схемой отношения* $\mathcal{R}$ называется множество атрибутов $\{A_1, A_2, \ldots, A_n\}$, и если D_i — домен атрибута A_i, $i = 1, 2, \ldots, n$, (обозначается $D_i = dom(A_i)$), то любое отношение $R \subseteq dom(\mathcal{R}) = D_1 \times D_2 \times \cdots \times D_n$ называется *отношением со схемой* $\mathcal{R}$; в этом случае также используют обозначение $R(A_1, A_2, \ldots, A_n)$.

В теории реляционных баз данных рассматривают только конечные отношения (т.е. $|R| < \infty$) и отображают их в виде плоских таблиц со строками — кортежами и столбцами — атрибутами.

Реляционная база данных — это множество отношений.

2.2.1 Реляционная алгебра

Определим некоторые операции над отношениями.

Булевы операции

Если R и S отношения с одной и той же схемой $\mathcal{R}$, т.е. подмножества одного универсума $dom(\mathcal{R})$, то к ним могут быть применены обычные теоретико-множественные операции: объединение $\cup$, пересечение $\cap$, разность $\setminus$, дополнение $\bar{}$.

Если какой-то из атрибутов схемы $\mathcal{R}$ имеет бесконечный домен, то дополнение R не будет конечным, и тем самым не будет отношением в реляционном понимании. Поэтому вводится понятие активного дополнения, которое всегда дает конечное отношение.

Если R — отношение со схемой $\mathcal{R} = \{A_1, \ldots, A_n\}$, и $r = (r_1, \ldots, r_n) \in R$, то обозначим $r(A_i) = r_i$; если

$$X = \{A_{i_1}, \ldots, A_{i_k}\} \subseteq \mathcal{R},$$

то обозначим $r(X) = (r_{i_1}, \ldots, r_{i_k})$. Обозначим

$$adom(A_i, R) = \{d \in dom(A_i) : \text{ существует } r \in R, \text{ что } r(A_i) = d\},$$

$$adom(\mathcal{R}, R) = adom(A_1, R) \times \cdots \times adom(A_n, R).$$

Активным дополнением отношения R называется

$$\tilde{R} = adom(\mathcal{R}, R) \setminus R.$$

Оператор выбора

Выбор — это унарная операция над отношением. Результатом ее применения к отношению R является другое отношение, которое представляет собой подмножество кортежей отношения R с определенным значением в выделенном атрибуте.

Пусть R — отношение со схемой $\mathcal{R}$, A — атрибут из $\mathcal{R}$ и a — элемент из $dom(A)$. Тогда $\sigma_{A=a}(R)$ — обозначение операции выбора в R кортежей, в которых значение A равно a, т.е.

$$\sigma_{A=a}(R) = \{r \in R : r(A) = a\}.$$

Если на $dom(A)$ задано бинарное отношение

$$\rho \subseteq dom(A) \times dom(A)$$

(пишем $a \rho b$, если $(a, b) \in \rho$), то выбор можно осуществлять с помощью этого отношения

$$\sigma_{A \rho a}(R) = \{r \in R : r(A) \rho a\}.$$

Оператор проекции

Оператор *проекции* также является унарным оператором на отношениях. Но если оператор выбора выбирает подмножество строк в отношении, то оператор проекции выбирает подмножество столбцов.

Пусть R — отношение со схемой $\mathcal{R}$ и $X \subseteq \mathcal{R}$. Проекция R на X, записанная как $\pi_X(R)$, есть отношение, полученное вычеркиванием столбцов, соответствующих атрибутам из $\mathcal{R} \setminus X$, и исключением из оставшихся столбцов повторяющихся строк, т.е.

$$\pi_X(R) = \{r(X) : r \in \mathcal{R}\}.$$

Оператор соединения

Соединение — это бинарный оператор для комбинирования двух отношений по всем их общим атрибутам. Рассмотрим два отношения $R_1(\mathcal{R}_1)$ и $R_2(\mathcal{R}_2)$. Положим $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$. *Соединением* отношений R_1 и R_2, записываемым как $R_1 \bowtie R_2$, является отношение $R(\mathcal{R})$, содержащее все кортежи r над $\mathcal{R}$, такие, что существуют кортежи $r_1 \in R_1$ и $r_2 \in R_2$, что $r(\mathcal{R}_1) = r_1$ и $r(\mathcal{R}_2) = r_2$.

Таким образом, каждый кортеж в R является комбинацией корежа из R_1 и кортежа из R_2 с равными $(\mathcal{R}_1 \cap \mathcal{R}_2)$-значениями.

Если $\mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset$, то $R_1 \bowtie R_2$ — это декартово произведение R_1 и R_2.

Пример. Пусть R_1 и R_2 таковы:

$R_1(A$	$B)$
a_1	b_1
a_2	b_1

$R_2(B$	$C)$
b_1	c_1
b_2	c_1
b_2	c_2

Тогда $R_1 \bowtie R_2$ будет:

$R_1 \bowtie R_2(A$	B	$C)$
a_1	b_1	c_1
a_2	b_1	c_1

Переименование атрибутов

Если $\mathcal{R}$ — схема отношения, $A \in \mathcal{R}$, $B \notin \mathcal{R} \setminus \{A\}$ — атрибуты, и $dom(A) = dom(B)$, то переименование атрибута A на B состоит

в замене в множестве $\mathcal{R}$ элемента A на B. Если R отношение со схемой $\mathcal{R}$, тогда R с A переименованным в B, обозначаемое как $\delta_{A \leftarrow B}(R)$, есть отношение

$$\delta_{A \leftarrow B}(R) = \{r' : \text{ существует } r \in R, \text{ что } r'(\mathcal{R} \setminus \{A\}) = r(\mathcal{R} \setminus \{A\}) \text{ и } r'(B) = r(A)\}.$$

Реляционная алгебра

Пусть $\mathbf{U}$ — множество атрибутов, называемое универсумом; $\mathcal{D}$ — множество доменов; dom — полная функция из $\mathbf{U}$ в $\mathcal{D}$; $\mathbf{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_p\}$ — множество различных схем отношений, где $\mathcal{R}_i \subseteq \mathbf{U}, i \in \{1, 2, \ldots, p\}$; $X = \{x_1, \ldots, x_p\}$ — множество переменных, причем переменная $x_i \in X$ может принимать в качестве значения любое отношение со схемой $\mathcal{R}_i$, $i \in \{1, 2, \ldots, p\}$; Θ — множество бинарных отношений над доменами из $\mathcal{D}$, содержащее по крайней мере отношения равенства и неравенства для каждого домена; O — множество операторов над отношениями, описанных выше, а именно: объединения, пересечения, разности, активного дополнения, проекции, соединения, переименования, использующего атрибуты из $\mathbf{U}$, выбора, использующего отношения из Θ. Тогда семерка $\mathcal{B} = (\mathbf{U}, \mathcal{D}, dom, \mathbf{R}, X, \Theta, O)$ называется *реляционной алгеброй*.

Согласуясь с ограничениями, наложенными на операторы, из переменных, принадлежащих X, постоянных отношений со схемами из $\mathbf{R}$ (под постоянным отношением со схемой $\mathcal{R}$ понимается любое фиксированное отношение со схемой $\mathcal{R}$) используя открывающие и закрывающие скобки и операторы из O, стандартным образом можно построить *алгебраические выражения* над $\mathcal{B}$. Каждому алгебраическому выражению над $\mathcal{B}$ естественным образом сопоставляется функция, реализуемая данным отношением, которая отображает некоторое множество отношений в одно отношение.

2.2.2 Функциональные зависимости

Пусть R — отношение со схемой $\mathcal{R}$, и X, Y — подмножества $\mathcal{R}$. Отношение R *удовлетворяет функциональной зависимости* (*F-зависимости*) $X \to Y$, если $\pi_Y(\sigma_{X=x}(R))$ имеет не более чем

один кортеж для каждого X-значения x. Иными словами, фиксация значений столбцов из X однозначно определяет значение столбцов из Y. В F-зависимости $X \to Y$ подмножество X называется *левой частью*, а Y — *правой частью*.

Если отношение R со схемой $\mathcal{R}$ удовлетворяет функциональной зависимости $X \to \mathcal{R}$, то множество X называется *суперключом*. То есть суперключ однозначно определяет кортеж. Понятие суперключа в точности соответствует понятию теста таблицы. Напомним, что *тестом таблицы* называется такое множество столбцов, что проекция таблицы на эти столбцы не имеет одинаковых строк. Суперключ отношения называется *ключом*, если любое его подмножество не является суперключом. Понятие ключа полностью соответствует понятию тупикового теста таблицы.

Если $\mathcal{R}$ — некоторая схема отношений, то через $\mathcal{M}(\mathcal{R})$ обозначим множество всех отношений над схемой $\mathcal{R}$. Если $M \subseteq \mathcal{M}(\mathcal{R})$, то скажем что *множество M удовлетворяет функциональной зависимости* $X \to Y$, если каждое отношение из M удовлетворяет $X \to Y$.

Часто поступают наоборот и описывают множество M допустимых отношений описывается с помощью множества F-зависимостей, которым должно удовлетворять каждое отношение. Это множество F-зависимостей фактически описывает наши семантические знания об отношении. Поэтому в дальнейшем мы будем считать, что вместе со схемой $\mathcal{R}$ задано множество $\mathcal{F}$ F-зависимостей, которым должны удовлетворять все допустимые отношения.

Если известны некоторые F-зависимости из $\mathcal{F}$, то часто можно вывести остальные. *Множество F-зависимостей $\mathcal{F}$ влечет F-зависимость* $X \to Y$ (обозначение: $\mathcal{F} \models X \to Y$), если каждое отношение, удовлетворяющее F-зависимостям из $\mathcal{F}$, удовлетворяет также F-зависимости $X \to Y$. В этом случае также говорят, что $X \to Y$ *является логическим следствием* $\mathcal{F}$, или что $X \to Y$ *следует из* $\mathcal{F}$.

Аксиома вывода — это правило, устанавливающее, что если отношение удовлетворяет определенным F-зависимостям, то оно должно удовлетворять и некоторым другим F-зависимостям.

Теперь введем шесть аксиом вывода для F-зависимостей. В их формулировках используется обозначение R для отношения со схемой $\mathcal{R}$ и W, X, Y и Z — для подмножеств $\mathcal{R}$.

F1. Рефлексивность. Отношение $\pi_X(\sigma_{X=x}(R))$ всегда имеет не более одного кортежа, следовательно, $X \to X$ всегда имеет место в R.

F2. Пополнение. Это аксиома имеет дело с расширением левой части F-зависимости. Если R удовлетворяет $X \to Y$, то $\pi_Y(\sigma_{X=x}(R))$ имеет не более одного кортежа для любого X-значения x. Если Z является любым подмножеством $\mathcal{R}$, то $\sigma_{X \cup Z=xz}(R) \subseteq \sigma_{X=x}(R)$ и, следовательно, $\pi_Y(\sigma_{X \cup Z=xz}(R)) \subseteq \pi_Y(\sigma X = x(R))$. Таким образом $\pi_Y(\sigma_{X \cup Z=xz}(R))$ имеет самое большее один кортеж, и R должно удовлетворять F-зависимости $X \cup Z \to Y$.

F3. Аддитивность. Эта позволяет объединить две F-зависимости с одинаковыми левыми частями. Если отношение R удовлетворяет $X \to Y$ и $X \to Z$, то обе проекции $\pi_Y(\sigma_{X=x}(R))$ и $\pi_Z(\sigma_{X=x}(R))$ имеют не более одного кортежа для любого X-значения x. Если бы отношение $\pi_{Y \cup Z}(\sigma_{X=x}(R))$ имело более одного кортежа, то по крайней мере одно из отношений $\pi_Y(\sigma_{X=x}(R))$ или $\pi_Z(\sigma_{X=x}(R))$ имело бы более одного кортежа. Таким образом, R удовлетворяет F-зависимости $X \to Y \cup Z$.

F4. Проективность. Эта аксиома в некоторой степени обратна аксиоме аддитивности. Если R удовлетворяет $X \to Y \cup Z$, то $\pi_{Y \cup Z}(\sigma_{X=x}(R))$ имеет не более одного кортежа для любого X-значения x. Так как $\pi_Y(\pi_{Y \cup Z}(\sigma_{X=x}(R))) = \pi_Y(\sigma_{X=x}(R))$, то $\pi_Y(\sigma_{X=x}(R))$ также может иметь не более одного кортежа. Следовательно, R удовлетворяет $X \to Y$.

F5. Транзитивность. Эта и следующая аксиома являются наиболее мощными из аксиом вывода. Пусть отношение R удовлетворяет F-зависимостям $X \to Y$ и $Y \to Z$. Рассмотрим кортежи r_1 и r_2 в R. Если $r_1(X) = r_2(X)$, то $r_1(Y) = r_2(Y)$, а из $r_1(Y) = r_2(Y)$, следует, что $r_1(Z) = r_2(Z)$. Таким образом, если $r_1(X) = r_2(X)$, то $r_1(Z) = r_2(Z)$, т.е. R удовлетворяет $X \to Z$.

F6. Псевдотранзитивность. Пусть отношение R удовлетворяет F-зависимостям $X \to Y$ и $Y \cup Z \to W$, а r_1 и r_2 являются кортежами в R. По условию если $r_1(X) = r_2(X)$, то $r_1(Y) = r_2(Y)$, и, аналогично, если $r_1(Y \cup Z) = r_2(Y \cup Z)$,

то $r_1(W) = r_2(W)$. Из $r_1(X \cup Z) = r_2(X \cup Z)$ получаем, что $r_1(X) = r_2(X)$ и $r_1(Z) = r_2(Z)$, значит, по предыдущему $r_1(Y) = r_2(Y)$ и, кроме того, $r_1(Y \cup Z) = r_2(Y \cup Z)$, откуда следует, что $r_1(W) = r_2(W)$. Таким образом, R удовлетворяет $X \cup Z \to W$.

Итак, если X, Y, Z, W являются подмножествами $\mathcal{R}$, то для любого отношения R справедливы следующие аксиомы вывода:

F1. Рефлексивность. $X \to X$.

F2. Пополнение. $X \to Y \models X \cup Z \to Y$.

F3. Аддитивность. $X \to Y,\ X \to Z \models X \to Y \cup Z$.

F4. Проективность. $X \to Y \cup Z \models X \to Y$.

F5. Транзитивность. $X \to Y,\ Y \to Z \models X \to Z$.

F6. Псевдотранзитивность. $X \to Y,\ Y \cup Z \to W \models X \cup Z \to W$.

Используя аксиомы F1–F6, можно получить другие правила вывода для F-зависимостей. Более того аксиомы F1–F6 не являются независимыми, т.е. одни из них могут быть получены из других. Так если даны аксиомы F1, F2 и F6, то из них можно вывести остальные. Если даны $X \to Y$ и $X \to Z$, то используем F1, чтобы получить $Y \cup Z \to Y \cup Z$, и дважды применяем F6, получая сначала $X \cup Z \to Y \cap Z$, а затем $X \to Y \cup Z$. Поэтому F3 следует из F1, F2 и F6. Чтобы доказать F4, предположим, что $X \to Y \cup Z$. Из F1 имеем $Y \to Y$, а из F2 получаем $Y \cup Z \to Y$. Применение F6 дает $X \to Y$. F5 является частым случаем F6 при $Z = \emptyset$. В качестве самостоятельного упражнения можно показать, что аксиомы F1, F2 и F6 являются независимыми, т.е. ни одна из них не может быть выведена из двух других.

2.2.3 Полнота системы аксиом вывода

Пусть A — некоторое множество аксиом вывода, $\mathcal{F}$ — множество функциональных зависимостей. Понятие *замыкания* $[\mathcal{F}]_A$ *множества* $\mathcal{F}$ *относительно системы аксиом* A вводится индуктивно:

1) если $f \in \mathcal{F}$, то $f \in [\mathcal{F}]_A$;

2) если f выводится из $[\mathcal{F}]_A$ однократным применением аксиом из A, то $f \in [\mathcal{F}]_A$;

3) других F-зависимостей в $[\mathcal{F}]_A$ нет.

Если $A = \{\text{F1,F2,F3,F4,F5,F6}\}$, то индекс A будем опускать и писать $[\mathcal{F}]$.

Если $f \in [\mathcal{F}]$, то будем говорить, что f *выводится из* $\mathcal{F}$.

В предыдущем разделе мы фактически показали, что если $f \in [\mathcal{F}]$, то $\mathcal{F} \models f$, т.е. если f выводится из $\mathcal{F}$, то f является логическим следствием $\mathcal{F}$. В этом разделе мы покажем обратное.

Система аксиом вывода A *полна*, если из $\mathcal{F} \models f$ следует, что $f \in [\mathcal{F}]_A$.

Теорема 8. *Если домен каждого из атрибутов имеет мощность более 1, то система аксиом* $A = \{\text{F1,F2,F3,F4,F5,F6}\}$ *полна.*

Доказательство. Пусть $\mathcal{R} = \{A_1, A_2, \ldots, A_n\}$ — схема отношения, $\mathcal{F}$ — некоторое множество F-зависимостей над $\mathcal{R}$, и $X \subseteq \mathcal{R}$, $Y \subseteq \mathcal{R}$. Покажем, что если $X \to Y \notin [\mathcal{F}]$, то $\mathcal{F} \not\models X \to Y$.

Чтобы показать, что $\mathcal{F} \not\models X \to Y$, нам надо найти такое отношение R, которое удовлетворяет всем зависимостям из $\mathcal{F}$, но не удовлетворяет $X \to Y$.

Пусть X^+ такое подмножество $\mathcal{R}$, что для любого $Z \subseteq \mathcal{R}$ из того, что $X \to Z \in [\mathcal{F}]$ следует, что $Z \subseteq X^+$, т.е. X^+ — это максимальное подмножество, зависящее от X. Понятно, что $X \subseteq X^+$.

Пусть $a_i, b_i \in dom(A_i)$, $i = 1, 2, \ldots, n$. Рассмотрим отношение R, состоящее из двух кортежей $r = (a_1, a_2, \ldots, a_n)$ и r', где

$$r'(A_i) = \begin{cases} a_i, & \text{если } A_i \in X^+, \\ b_i, & \text{если } A_i \notin X^+, \end{cases} \quad (i = 1, 2, \ldots, n).$$

Покажем, что если $X \to Y \notin [\mathcal{F}]$, то отношение R не удовлетворяет $X \to Y$.

По определению $r(X) = r'(X)$. Предположим, что $r(Y) = r'(Y)$. Тогда $Y \subseteq X^+$ и, значит, $X^+ \to Y$, но $X \to X^+ \in [\mathcal{F}]$, откуда следует, что $X \to Y \in [\mathcal{F}]$. Противоречие.

Теперь покажем, что R удовлетворяет всем F-зависимостям из $[\mathcal{F}]$.

Имеет смысл рассматривать только F-зависимости вида $W \to Z$, где $W \subseteq X^+$, так как если $W \not\subseteq X^+$, то $r(W) \neq r'(W)$. Если $W \subseteq X^+$, то $X^+ \to W \in [\mathcal{F}]$. Учитывая $X \to X^+ \in [\mathcal{F}]$ и $W \to Z \in [\mathcal{F}]$, по транзитивности имеем $X \to Z \in [\mathcal{F}]$. Откуда следует, что $Z \subseteq X^+$ и, значит, $r(Z) = r'(Z)$. Следовательно, R удовлетворяет $W \to Z$.

Значит, R удовлетворяет всем F-зависимостям из $[\mathcal{F}]$, но не удовлетворяет $X \to Y$. Значит $\mathcal{F} \not\models X \to Y$.

Теорема доказана. □

Упражнения

2.1. Пусть $R(A, B, C)$ и $S(D, C, D)$ — отношения, $a \in dom(A)$ и $b \in dom(B)$. Какие из следующих выражений правильно составлены:

а) $R \cap S$;

б) $\pi_B(R) \setminus \pi_B(S)$;

в) $\sigma_{B=b)}(R)$;

г) $\sigma_{A=a,B=b)}(S)$;

д) $R \bowtie S$;

е) $\pi_A(R) \bowtie \pi_D(S)$?

2.2. Отношения R и S таковы:

$R(A$	B	$C)$
a	b	c
a	b'	c'
a	b''	c'
a'	b'	c

$S(B$	C	$D)$
b'	c'	d
b''	c'	c'
b''	c	d

Вычислить значения следующих выражений:

а) $\widetilde{R}$;

б) $\widetilde{S}$;

в) $\sigma_{A=a)}(R)$;

г) все правильно составленные выражения в упр. 2.1.

2.3. Пусть R и S — отношения со схемой $\mathcal{R}$ и ключом K. Какие из следующих отношений обязательно должны иметь ключ K:

а) $R \cup S$;

б) $R \cap S$;

в) $R \setminus S$;

г) $\overline{R}$, где $\overline{R}$ — отношение;

д) $\pi_K(R)$;

е) $R \bowtie S$?

2.4. Пусть $X \subseteq \mathcal{R}$, и R, S — отношения со схемой $\mathcal{R}$. Докажите или опровергните утверждение

а) $\pi_X(R \cap S) = \pi_X(R) \cap \pi_X(S)$;

б) $\pi_X(R \cup S) = \pi_X(R) \cup \pi_X(S)$;

в) $\pi_X(R \setminus S) = \pi_X(R) \setminus \pi_X(S)$;

г) $\pi_X(\overline{R}) = \overline{\pi_X(R)}$, где $\overline{R}$ — отношение.

2.5. Пусть R и R' — отношения со схемой $\mathcal{R}$ и S — отношение со схемой $\mathcal{S}$. Докажите или опровергните утверждение

а) $(R \cap R') \bowtie S = (R \bowtie S) \cap (R' \bowtie S)$;

б) $(R \setminus R') \bowtie S = (R \bowtie S) \setminus (R' \bowtie S)$;

в) $\widetilde{R} \bowtie \widetilde{S} = \widetilde{R \bowtie S}$.

2.6. Для данных отношений $R(\mathcal{R})$ и $S(\mathcal{S})$ и $Q = R \bowtie S$ определено $R' = \pi_{\mathcal{R}}(Q)$ и $S' = \pi_{\mathcal{S}}(Q)$. Докажите, что $Q = R' \bowtie S'$.

2.7. Докажите, что отношение R удовлетворяет $X \to Y$ тогда и только тогда, когда X является ключом для $\pi_{X \cup Y}(R)$.

2.8. Пусть R — отношение со схемой $\mathcal{R}$, $X \subset \mathcal{R}$. Покажите, что если $\pi_X(R)$ имеет то же число кортежей, что и R, то R удовлетворяет $X \to Y$ для любого Y из $\mathcal{R}$.

2.9. Докажите или опровергните следующие правила вывода для отношения R со схемой $\mathcal{R}$, в которых W, X, Y и Z — подмножества $\mathcal{R}$:

а) из $X \to Y$ и $Z \to W$ следует $X \cup Y \to Y \cup W$;

б) из $X \cup Y \to Z$ и $Z \to X$ следует $Z \to X$;

в) из $X \to Y$ и $Y \to Z$ следует $X \to Y \cup Z$;

г) из $X \to Y$, $W \to Z$ и $W \subseteq Y$ следует $X \to Z$.

2.10. Докажите, что аксиомы вывода F1, F2 и F6 являются независимыми. Это значит, что ни одна их них не может быть получена из двух других.

2.3 Информационно-графовая модель данных

Если рассматриваемая в предыдущем разделе реляционная модель данных является моделью логической организации данных, описывающая базу данных с точки зрения пользователя, то предлагаемая в это разделе модель данных, предназначена для исследования физической организации данных, т.е. представления данных на физических носителях. Эта модель данных называется информационно-графовой и была разработана вторым из авторов.

Критерии, определяющие выбор физической организации, отличаются от тех, которые определяют выбор логической организации данных. При выборе физической организации решающим фактором является эффективность, причем согласно Дж. Мартину [40] на первом месте стоит обеспечение эффективности поиска, далее идут эффективность операций занесения и удаления и затем обеспечение компактности данных.

На первом шаге нам необходимо ввести понятие задачи информационного поиска.

В понятие задачи поиска исследователи вкладывают по крайней мере 3 различных смысла.

Специалисты в теории исследования операций понимают задачи поиска как задачи управления сближением одной системы (поисковой) с другой (искомым объектом) по неполной априорной информации. Понимается, что цель поиска — это обнаружение искомого объекта, определяемое как выполнение определенных терминальных условий. Интенсивно проблемой поиска подвижных объектов начали заниматься в период Второй мировой войны. Интерес к этой проблеме в тот период был вызван необходимостью разработки тактики борьбы против подводных лодок. Среди работ, посвященных этой тематике, можно выделить, например, работы О. Хеллмана [61] и Д. П. Кима [21].

Другое понимание задач поиска можно найти в книге Р.Альсведе, И.Вегенера "Задачи поиска" [2]. Приведем поясняющий пример из этой книги.

Во время Второй мировой войны все призываемые в армию США подвергались проверке на реакцию Вассермана. При этом

проверялось, есть ли в крови обследуемого определенные антитела, имеющиеся только у больных. Во время этого массового обследования было замечено, что разумнее анализировать пробу крови целой группы людей. Если такая объединенная проба крови не содержит антител, то, значит, ни один из обследуемых не болен. В противном случае среди них есть хотя бы один больной. Хороший алгоритм поиска для этой задачи — это тот, который для типичной выборки людей позволяет "наискорейшим образом" выявлять множество всех больных.

Другой пример мы находим в статье Д.Ли, Ф.Препараты "Вычислительная геометрия" [36]. Дано множество горизонтальных и вертикальных отрезков. Надо найти все точки пересечения отрезков.

Эти два примера объединяет то, что в обоих случаях поиск производится однократно. Это порождает свои особенности таких задач поиска. Как правило, данные в этих задачах не имеют сложной организации. Фиксация множества, в котором производится поиск, однозначно определяет множество найденных объектов. "Хорошесть" алгоритма поиска определяется при варьировании множества, в котором производится поиск.

В третьем понимании задачи поиска предполагается многократное обращение к одним и тем же данным, но возможно каждый раз с разными требованиями к искомым объектам, то есть с разными запросами на поиск. Такие задачи поиска обычно возникают в системах, использующих базы данных. Многократное использование порождает особую проблему — проблему специальной организации данных, направленной на последующее ускорение поиска. Процесс такой специальной организации данных, проводимый до того, как осуществляется поиск, называется предобработкой и часто может занимать очень большое время, которое затем окупается сторицей в результате многократности поиска. Простейшим примером предобработки является сортировка. Построение "хорошего" алгоритма поиска в этом случае сводится к нахождению хороших структур данных, то есть к осуществлению такой хорошей предобработки данных, которая обеспечила бы хорошую скорость поиска. "Хорошесть" алгоритма поиска в этом случае определяется варьированием запроса на поиск, например, как среднее время

поиска на запросе.

В зависимости от того, является ли база данных фиксированной или изменяется на протяжении времени работы с ней, мы имеем два типа организации баз данных: *статическую* и *динамическую* соответственно.

Задачи поиска, возникающие в статических базах данных и предполагающие многократное обращение к одним и тем же данным, и являются объектом исследования данного раздела.

Приведем примеры таких задач поиска.

Простейшим и самым распространенным примером задачи поиска, встречающейся в любой базе данных, является задача поиска по ключу. Суть ее состоит в том, что любой объект в базе данных имеет свой уникальный ключ. Это может быть порядковый номер, уникальное имя, или уникальное значение некоторого поля, например, номер паспорта. Задача состоит в том, чтобы по заданному в запросе ключу найти в базе данных объект с этим ключом (если такой объект в базе имеется). Более формально эту задачу можно поставить следующим образом. Имеется некоторое конечное множество ключей. Имеется некоторое более широкое множество запросов. Требуется по произвольному запросу из множества запросов найти в множестве ключей ключ идентичный (равный) ключу-запросу. В такой постановке эта задача называется задачей поиска идентичных объектов.

Другой пример взят из систем машинной графики, обработки изображений и систем автоматизированного проектирования Дано конечное множество точек из отрезка $[0, 1]$ вещественной прямой. Множество запросов есть множество всех отрезков, содержащихся в $[0, 1]$. Надо для произвольного запроса $[u, v]$ из множества запросов перечислить все точки из нашего множества точек, которые попадают в отрезок $[u, v]$. Эта задача носит название одномерной задачи интервального поиска.

Если мы хотим изучать сложность алгоритмов поиска, то без введения математической модели объекта изучения нам не обойтись.

Поэтому начнем с формализации понятия задачи информационного поиска (ЗИП). В работах [7, 53, 56, 69] вводились различные формализации ЗИП. Так в [69] вводилась формализа-

ция наиболее близкая к той, которую мы будем использовать в данной работе. Формализация ЗИП в [69] вводилась следующим образом: вопрос к базе данных представляет некоторый запрос, имеющий тип $T1$, сама база данных состоит из элементов типа $T2$, а ответ на вопрос — значение типа $T3$, например, $T3$ может иметь логический тип, если предполагается, что ответ на запрос должен быть "да" или "нет", $T3$ может совпадать с $T2$, если ответом на запрос является элемент базы данных, и наконец $T3$ может быть множеством элементов типа $T2$, если в ответ на запрос надо перечислять некоторые элементы из базы данных. Вопрос Q рассматривается как отображение из $T1$ и множества подмножеств $T2$ в $T3$, то есть $Q: T1 \times 2^{T2} \to T3$.

Мы будем рассматривать только такие задачи поиска, в которых в ответ на запрос надо перечислить элементы базы данных, удовлетворяющие запросу. ЗИП такого типа называются задачами на перечисление. С учетом этого факта мы и дадим собственную формализацию ЗИП, более удобную для использования в дальнейшем.

Из приведенных примеров видно, что в задачах поиска имеется некий универсум, из которого берутся объекты поиска (элементы базы данных). В первом примере таким универсумом является множество всевозможных ключей, а во втором — отрезок $[0,1]$ вещественной прямой. Этот универсум обозначим через Y и будем называть множеством объектов поиска или *множеством записей*, а элементы множества Y, будем называть, *записями*.

Далее в задачах поиска всегда имеется *множество запросов*. В первом примере множество запросов совпадает с множеством объектов поиска и является множеством всевозможных ключей, а во втором примере множество запросов есть множество всех отрезков, содержащихся в $[0,1]$, то есть множество $\{(u,v): 0 \le u \le v \le 1\}$. Множество запросов будем обозначать через X.

На декартовом произведении $X \times Y$ имеется бинарное отношение, которое позволяет устанавливать, когда запись из Y удовлетворяет запросу из X. Это отношение будем называть *отношением поиска*. В первом примере отношение поиска есть отношение идентичности (равенства), то есть запись удовлетво-

ряет запросу, если они идентичны. Во втором примере отношение поиска, которое обозначим через ρ_{int}, определяется соотношением

$$(u, v)\rho_{int} y \Longleftrightarrow u \leq y \leq v, \tag{2.1}$$

где $(u, v) \in X$, $y \in Y$.

Тройку $S = \langle X, Y, \rho \rangle$, где X — множество запросов, Y — множество записей, ρ — отношение поиска, заданное на $X \times Y$, будем называть *типом*, или иногда более развернуто *типом задач информационного поиска*.

Тройку $I = \langle X, V, \rho \rangle$, где X — множество запросов; V — некоторое конечное подмножество множества Y, в дальнейшем называемое *библиотекой*; ρ — отношение поиска, заданное на $X \times Y$, будем называть *задачей информационного поиска* (ЗИП) типа $S = \langle X, Y, \rho \rangle$. Содержательно будем считать, что задача $I = \langle X, V, \rho \rangle$ состоит в перечислении для произвольно взятого запроса $x \in X$ всех тех и только тех записей из V , которые находятся в отношении ρ с запросом x, то есть удовлетворяют запросу x.

Тем самым мы формализовали понятие ЗИП.

Для полной определенности отметим, что поскольку библиотека V есть множество, то в ней все элементы различные, то есть в ней нет повторяющихся элементов.

Следующий шаг, который мы обязаны сделать — это выбрать математическую модель для алгоритмов поиска.

Отметим, что здесь и всюду далее, используя термин "алгоритм", мы часто будем подменять им понятие "условного алгоритма" (или "относительного алгоритма", см. [39, с. 44–45]), то есть будем рассматривать алгоритмы, которые выполняются при условии, что мы умеем выполнять некоторые операции из описанного заранее множества. В качестве таких операций могут выступать, например, некоторые операции над вещественными числами, но так или иначе мы всегда будем явно оговаривать операции, относительно которых рассматривается каждый описываемый условный алгоритм.

Прежде чем предложить модель алгоритмов поиска взглянем на алгоритм, решающий задачу поиска с функциональной точки зрения.

Рассмотрим произвольную ЗИП $I = \langle X, V, \rho \rangle$. Алгоритм поиска решает ЗИП, если на любой запрос из множества запросов X он выдает все те и только те записи из V, которые удовлетворяют запросу. Возьмем произвольную запись $y \in Y$. Для нее можно ввести *характеристическую функцию*

$$\chi_{y,\rho}(x) = \begin{cases} 1, & \text{если } x \rho y \\ 0 & \text{в противном случае} \end{cases},$$

то есть она равна 1 на тех запросах, которым удовлетворяет запись y. Тогда можно сказать, что алгоритм, решающий ЗИП $I = \langle X, V, \rho \rangle$, где $V = \{y_1, \ldots, y_k\}$, — ни что иное как алгоритм, реализующий систему функций $\{\chi_{y_1,\rho}, \ldots, \chi_{y_k,\rho}\}$. Следовательно, управляющая система, моделирующая алгоритм, решающий ЗИП $I = \langle X, V, \rho \rangle$, должна представлять собой многополюсник, реализующий множество характеристических функций $\{\chi_{y_1,\rho}, \ldots, \chi_{y_k,\rho}\}$.

Рассмотрим случай когда множество запросов $X = B^n$ — n-мерный единичный куб. Тогда каждая из функций $\chi_{y_i,\rho}$ будет функцией алгебры логики и, значит, в классе контактных схем [37] можно построить многополюсник, реализующий множество функций $\{\chi_{y_1,\rho}, \ldots, \chi_{y_k,\rho}\}$ как функций проводимости.

Если множество запросов не является n-мерным единичным кубом, то можно вместо множества $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$ использовать для нагрузки ребер некоторое множество F предикатов, определенных на множестве запросов X. Тогда при удачном выборе множества F и правильной нагрузке ребер многополюсника предикатами из F мы можем в качестве функций проводимости между полюсами получить характеристические функции $\chi_{y_i,\rho}$ $(i = \overline{1,k})$. Но если вводить сложность полученной многополюсной сети так же как в контактных схемах (то есть как количество ребер сети), то эта сложность скорее будет характеризовать объем памяти, соответствующего алгоритма, но не время поиска. Чтобы сложность сети характеризовала время поиска, ее иначе. А именно, будем считать, что сеть у нас ориентированная, что в сети есть такой полюс, который называется *корнем*, и каждая из характеристических функций записей $\chi_{y_i,\rho}$ $(i = \overline{1,k})$ реализуется как функция проводимости между корнем и своим полюсом, который отметим приписыванием ему записи y_i. Теперь если взять произвольный запрос

$x \in X$, то алгоритм функционирования сети на запросе x можно описать аналогично алгоритму разметки графа [18, 32]: считаем, что в начальный момент все вершины сети, кроме корня, неотмеченные, а некоторое упорядоченное множество вершин сети, которое назовем *множеством активных вершин*, содержит только корень сети. На каждом очередном шаге делаем следующее. Если множество активных вершин не пусто, то выбираем первую активную вершину и удаляем ее из множества активных вершин. Если выбранная вершина является полюсом, то соответствующую ей запись включаем в ответ на запрос x. Просматриваем в некотором порядке все ребра, исходящие из выбранной вершины, и если предикат, приписанный просматриваемому ребру, на запросе x принимает значение 1 и конец просматриваемого ребра неотмеченная вершина, то отмечаем конец просматриваемого ребра и включаем его в множество активных вершин (если мы включим его в начало множества активных вершин, то получим алгоритм обхода "сначала вглубь", а если в конец — то "сначала вширь"). Алгоритм завершает работу в тот момент, когда множество активных вершин окажется пустым.

Легко видеть, что если между корнем сети и некоторой вершиной на запросе x есть проводимость, то есть из корня в эту вершину ведет некоторая цепочка ребер, каждому из которых приписан предикат, принимающий значение 1 на запросе x, то обязательно в результате описанного алгоритма обхода сети данная вершина окажется отмеченной, то есть алгоритм обхода посетит ее.

Описанный алгоритм обхода сети можно считать соответствующим сети алгоритмом поиска на запросе x, а сложность сети на запросе x можно считать равной, числу просмотренных ребер, или, что то же самое числу вычисленных алгоритмом обхода предикатов. Такая сложность будет характеризовать время работы соответствующего алгоритма поиска на запросе x.

Описанная сеть с описанным функционированием является частным случаем *информационного графа (ИГ)*.

Прежде чем дать строгое формальное определение ИГ, введем некоторые вспомогательные понятия и обозначения.

Пусть M — некоторое конечное множество. Через $|M|$ обо-

значим число элементов во множестве M, называемое *мощностью множества* M.

Через $\overline{\{1,m\}}$ договоримся обозначать множество $\{1,2,\ldots,m\}$.

Некоторые оценки мы будем приводить с точностью до главного члена, поэтому введем обозначения, обычно принятые при описании асимптотических оценок.

Будем писать $\alpha(n) = \bar{o}(1)$, если $\lim\limits_{n\to\infty} \alpha(n) = 0$; $A(n) = \bar{o}(B(n))$, если $A(n) = B(n)\cdot\bar{o}(1)$. Скажем, что $A(n)$ *асимптотически не превосходит* $B(n)$ при $n\to\infty$ и обозначим $A \lesssim B$, если существует $\alpha(n) = \bar{o}(1)$ такое, что начиная с некоторого номера n_0, $A(n) \le (1+\alpha(n))\cdot B(n)$. Если $A \lesssim B$ и $B \lesssim A$, то будем говорить, что A и B *асимптотически равны* при $n\to\infty$ и обозначать $A \sim B$. Будем писать $A \precsim B$, если существует такая положительная константа c, что, начиная с некоторого номера n_0, $A(n) \le c\cdot B(n)$. Если $A \precsim B$ и $B \precsim A$, то будем говорить, что A и B *равны по порядку* при $n\to\infty$ и обозначать $A \asymp B$ или $A = \underline{O}(B)$.

Через $\binom{n}{k}$ будем обозначать *число сочетаний из* n *элементов по* k. Если r — действительное число, то через $[r]$ будем обозначать максимальное целое, не превышающее r, а через $]r[$ — минимальное целое, не меньшее, чем r. Значок $\stackrel{\text{def}}{=}$ будем понимать как "по определению равно". Математическое ожидание будем обозначать значком $\mathbf{M}$, а значок $\mathbf{M}_x$ будем понимать как среднее значение при вариации переменной x.

Договоримся также о теоретико-графовой терминологии.

Пусть нам дан ориентированный граф. В ориентированном ребре (α,β) вершину α будем называть *началом ребра*, а β — *концом*. Скажем, что ориентированное ребро графа *исходит из вершины* β (*входит в вершину* β), если β — начало (конец) данного ребра. Скажем, что ребро *инцидентно* вершине, если эта вершина является одним из концов данного ребра. *Полустепенью исхода* (*захода*) вершины графа назовем число ребер, исходящих из данной вершины (входящих в данную вершину). *Степенью инцидентности вершины* назовем число инцидентных ей ребер. Вершину графа назовем *концевой*, если полустепень ее исхода равна 0. Остальные вершины графа назовем *внутрен-*

ними.

Последовательность ориентированных ребер графа

$$(\alpha_1, \alpha_2), (\alpha_2, \alpha_3), \ldots, (\alpha_{m-1}, \alpha_m)$$

назовем *ориентированной цепью* от вершины α_1 к вершине α_m.

Если f — одноместный предикат, определенный на X , то есть $f : X \to \{0, 1\}$, то множество $N_f = \{x \in X : f(x) = 1\}$ назовем *характеристическим множеством предиката* f.

Множество $O(y, \rho) = \{x \in X : x \rho y\}$ назовем *тенью записи* $y \in Y$.

Функцию $\chi_{y,\rho} : X \to \{0, 1\}$ такую, что $N_{\chi_{y,\rho}} = O(y, \rho)$, назовем *характеристической функцией записи* y.

В формальном определении понятия ИГ используются 4 множества:

- множество запросов X;
- множество записей Y;
- *множество* F *одноместных предикатов*, заданных на множестве X;
- *множество* G *одноместных переключателей*, заданных на множестве X (*переключатели* — это функции, область значений которых является начальным отрезком натурального ряда).

Пару $\mathcal{F} = \langle F, G \rangle$ будем называть *базовым множеством.*

Определение понятия ИГ разбивается на два шага. На первом шаге раскрывается структурная (схемная) часть этого понятия, на втором — функциональная.

Определение ИГ с точки зрения его структуры.

Пусть нам дана ориентированная многополюсная сеть.

Выделим в ней один полюс и назовем его *корнем*, а остальные полюса назовем *листьями.*

Выделим в сети некоторые вершины и назовем их *точками переключения* (полюса могут быть точками переключения).

Если β — вершина сети, то через ψ_β обозначим *полустепень исхода* вершины β.

Каждой точке переключения β сопоставим некий символ из G. Это соответствие назовем *нагрузкой точек переключения.*

Для каждой точки переключения β ребрам, из нее исходящим, поставим во взаимно однозначное соответствие числа из множества $\{\overline{1,\psi_\beta}\}$. Эти ребра назовем *переключательными*, а это соответствие — *нагрузкой переключательных ребер.*

Ребра, не являющиеся переключательными, назовем *предикатными.*

Каждому предикатному ребру сети сопоставим некоторый символ из множества F. Это соответствие назовем *нагрузкой предикатных ребер.*

Сопоставим каждому листу сети некоторую запись из множества Y. Это соответствие назовем *нагрузкой листьев.*

Полученную нагруженную сеть назовем *информационным графом* над базовым множеством $\mathcal{F} = \langle F, G \rangle$.

Определение функционирования ИГ.

Скажем, что предикатное ребро проводит запрос $x \in X$, если предикат, приписанный этому ребру, принимает значение 1 на запросе x; переключательное ребро, которому приписан номер n, проводит запрос $x \in X$, если переключатель, приписанный началу этого ребра, принимает значение n на запросе x; ориентированная цепочка ребер проводит запрос $x \in X$, если каждое ребро цепочки проводит запрос x; запрос $x \in X$ проходит в вершину β ИГ, если существует ориентированная цепочка, ведущая из корня в вершину β, которая проводит запрос x; запись y, приписанная листу α, попадает в ответ ИГ на запрос $x \in X$, если запрос x проходит в лист α. *Ответом ИГ U на запрос x* назовем множество записей, попавших в ответ ИГ на запрос x, и обозначим его $\mathcal{J}_U(x)$. Эту функцию $\mathcal{J}_U(x) : X \to 2^Y$ будем считать результатом функционирования ИГ U и называть *функцией ответа ИГ U.*

Понятие ИГ полностью определено.

Проиллюстрируем приведенное определение на примере одномерной задачи интервального поиска. В этом случае 4 множества, определяющие ИГ, имеют вид:

- множество запросов $X_{int} = \{(u, v) : 0 \le u \le v \le 1\}$;

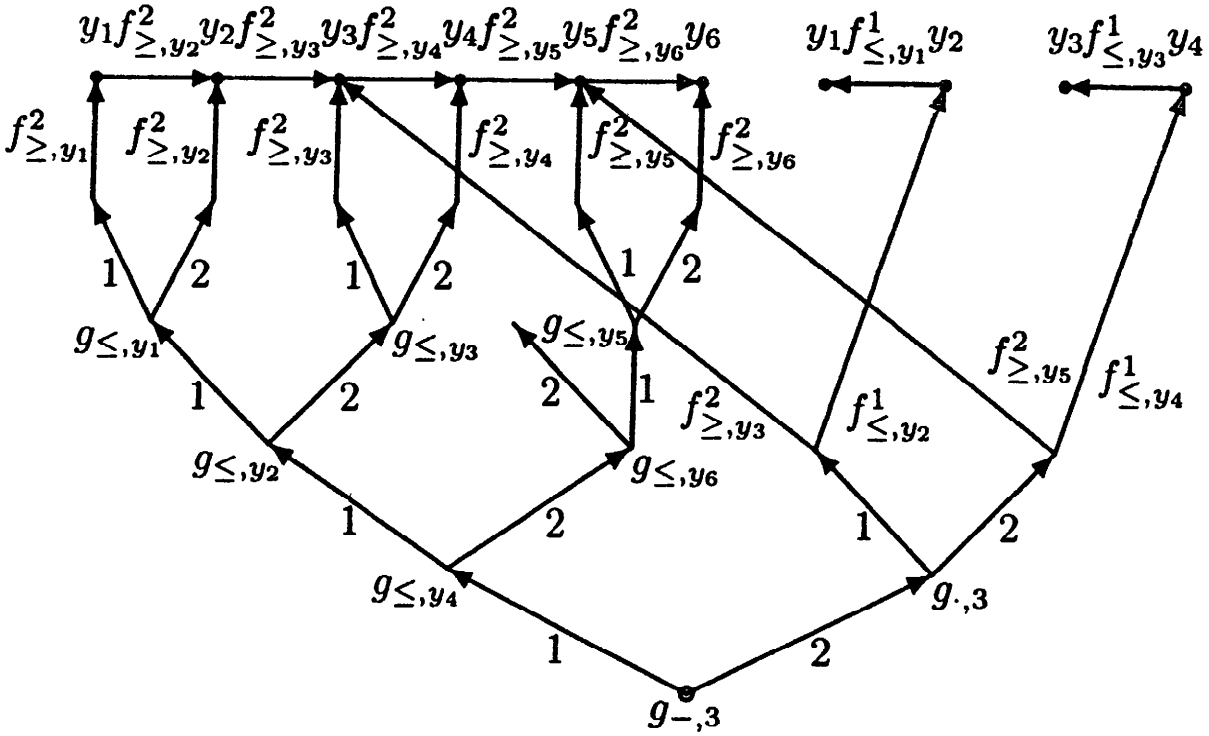

Рис. 2.1: Решение одномерной задачи интервального поиска

- множество записей $Y_{int} = [0, 1]$;

- множество предикатов
  $F = F_1 \bigcup F_2$, где $F_1 = \{f^1_{\leq,a} : a \in [0,1]\}$, $F_2 = \{f^2_{\geq,a} : a \in [0,1]\}$,

$$f^1_{\leq,a}(u,v) = \begin{cases} 1, & \text{если } u \leq a \\ 0, & \text{если } u > a \end{cases}, \tag{2.2}$$

$$f^2_{\geq,a}(u,v) = \begin{cases} 1, & \text{если } v \geq a \\ 0, & \text{если } v < a \end{cases}, \tag{2.3}$$

- множество переключателей
  $G = G_1 \bigcup G_2 \bigcup G_3$, где $G_1 = \{g_{\cdot,m} : m \in \mathbf{N}\}$, $G_2 = \{g_{-,m} : m \in \mathbf{N}\}$, $G_3 = \{g_{\leq,a} : a \in (0,1]\}$,

$$g_{\cdot,m}(u,v) = \max(1,]u \cdot m[), \tag{2.4}$$

$$g_{-,m}(u,v) = \begin{cases} 1, & \text{если } v - u < 1/m \\ 2, & \text{если } v - u \geq 1/m \end{cases}, \tag{2.5}$$

$$g_{\leq,a}(u,v) = \begin{cases} 1, & \text{если } u \leq a \\ 2, & \text{если } u > a \end{cases}. \tag{2.6}$$

В информационном графе, приведенном на рисунке 2.1, корень изображен полым кружком. Листья изображены жирными точками, а записи, приписанные листьям, — это символы y с индексами. На рисунке имеется 8 переключательных вершин (им приписаны символы g с индексами) и 17 предикатных ребра (им приписаны символы f с индексами).

Если n — натуральное число, а $g(x)$ — некий переключатель, то через $\xi_g^n(x)$ обозначим предикат, определенный на X , такой, что

$$N_{\xi_g^n} = \{x \in X : g(x) = n\}.$$

Обозначим

$$\widehat{G} = \{\xi_g^n : g \in G, n \in \mathbf{N}\},$$

где $\mathbf{N}$ — множество натуральных чисел.

Если c — ребро ИГ, то через $[c]$ обозначим его нагрузку.

В соответствии с приведенными выше определениями введем функции проводимости.

Проводимостью ребра (α, β) назовем предикат, равный $[(\alpha, \beta)]$, если ребро предикатное, и $\xi_g^{[(\alpha,\beta)]}$, если ребро переключательное, где g — переключатель, соответствующий вершине α.

Проводимостью ориентированной цепи назовем конъюнкцию проводимостей ребер цепи.

Если зафиксировать запрос x, то цепь, проводимость которой на запросе x равна 1, назовем *проводящей цепью на запросе* x.

В ИГ по аналогии с контактными схемами введем для каждой пары вершин α и β *функцию проводимости* $f_{\alpha\beta}$ от вершины α к вершине β следующим образом:

- если $\alpha = \beta$, то $f_{\alpha\beta}(x) \equiv 1$ $(x \in X)$;
- если $\alpha \neq \beta$ и в ИГ не существует ориентированных цепей от α к β, то $f_{\alpha\beta}(x) \equiv 0$;
- если $\alpha \neq \beta$ и множество ориентированных цепей от α к β не пусто, то $f_{\alpha\beta}(x)$ равно дизъюнкции проводимостей всех ориентированных цепей от α к β.

Функцию проводимости от корня ИГ к некоторой вершине β ИГ назовем *функцией фильтра вершины* β и обозначим $\varphi_\beta(x)$.

Через $\mathcal{R}(U), \mathcal{P}(U), \mathcal{L}(U)$ (или просто $\mathcal{R}, \mathcal{P}, \mathcal{L}$) обозначим множества вершин, точек переключения и листьев ИГ U соответственно.

Пусть $\mathcal{N}$ — некоторая подсеть (то есть произвольное подмножество вершин и ребер) ИГ U. Через $\langle\mathcal{N}\rangle$ обозначим множество записей, соответствующих листьям этой подсети (если α — некоторый лист ИГ U, то под $\langle\alpha\rangle$ будем понимать запись, соответствующую листу α).

Легко видеть, что функция ответа ИГ U определяется соотношением

$$\mathcal{J}_U(x) = \langle\{\alpha \in \mathcal{L}(U) : \varphi_\alpha(x) = 1\}\rangle.$$

Из определения функционирования ИГ видно, что ИГ как управляющая система может рассматриваться в качестве модели алгоритма поиска, работающего над данными, организованными в структуру, определяемую структурой ИГ.

В случае, когда базовое множество переключателей G пусто, то есть в графах нет переключателей, то ИГ называются *предикатными информационными графами* (ПИГ).

ПИГ, различным листьям которого соответствуют различные записи, называется *однозначным информационным графом* (ОИГ).

ОИГ, имеющий вид дерева, листья которого совпадают с концевыми вершинами дерева, назовем *информационным деревом* (ИД).

ИД удобны и интересны тем, что структуры данных, им соответствующие, практичны и их гораздо проще реализовать на ЭВМ.

Приведем пример еще одного ИГ. Пусть $I = \langle X, V, = \rangle$ — задача поиска идентичных объектов, где на множестве $V = \{y_1, \ldots, y_7\}$ задан линейный порядок и записи упорядочены в порядке возрастания, то есть $y_1 \le y_1 \le \cdots \le y_7$. Пусть множество предикатов имеет вид

$$F = \{f_{=,a}(x) = \begin{cases} 0, & \text{если } x \neq a \\ 1, & \text{если } x = a \end{cases} : a \in X\}, \qquad (2.7)$$

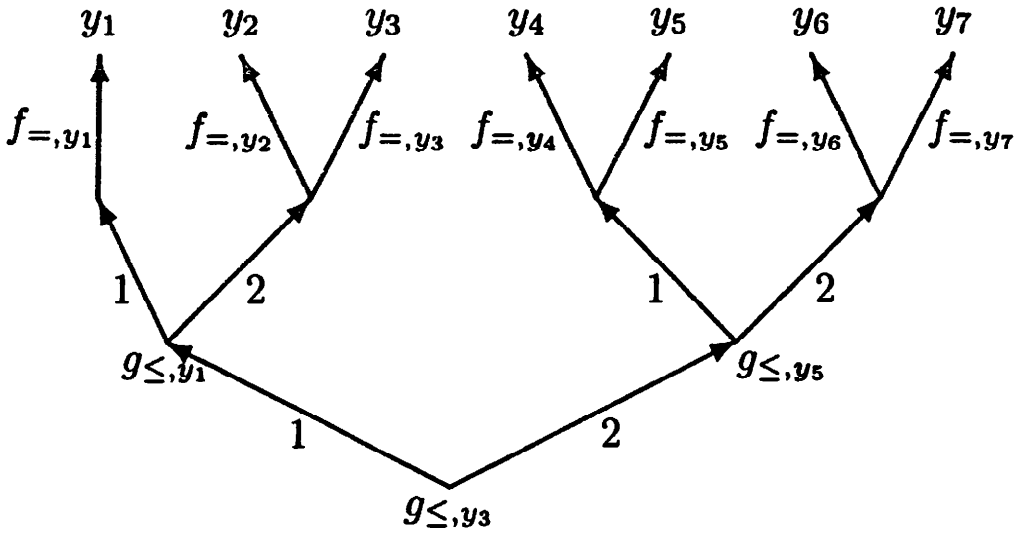

Рис. 2.2: Информационный граф бинарного поиска

а множество переключателей — вид

$$G = \{g_{\leq,a}(x) = \begin{cases} 1, & \text{если } x \leq a \\ 2, & \text{если } x > a \end{cases} : a \in X\}. \qquad (2.8)$$

Бинарным поиском или *методом деления пополам* (см., например, [7, 23, 40]) называется алгоритм поиска в упорядоченном массиве, при котором массив делится пополам, запрос сравнивается со средней точкой и в зависимости от результата сравнения поиск рекурсивно повторяется в одной из половин.

На рисунке 2.2 приведен ИГ над базовым множеством $\mathcal{F} = \langle F, G \rangle$, решающий ЗИП I, соответствующий бинарному поиску в версии Боттенбрука [71, с. 214], в которой вопрос о равенстве записи и запроса откладывается до самого последнего момента.

Введение управляющей системы, то есть схемы с функционированием, для моделирования алгоритмов поиска позволяет использовать аппарат теории сложности управляющих систем для исследования сложности алгоритмов поиска.

Упражнения

2.11. По аналогии с одномерной задачей интервального поиска приведите тип, описывающий n-мерные задачи интервального поиска.

2.12. Опишите тип, задающий задачи интервального поиска на n-мерном булевом кубе, которые состоят в поиске в конечном подмножестве n-мерного булевого куба всех тех точек, которые попадают в подкуб, задаваемый запросом. Какова мощность множества запросов у данного типа?

2.13. Задача о метрической близости состоит в том, чтобы по произвольно взятой точке-запросу единичного n-мерного куба n-мерного евклидового пространства найти в конечном подмножестве этого куба (библиотеке) все точки, находящиеся на расстоянии не более, чем R от точки запроса. Опишите тип, задающий задачи о метрической близости.

2.14. Опишите тип, задающий задачи включающего поиска. Напомним, что в задаче включающего поиска имеется некоторое конечное множество свойств, и каждый элемент библиотеки (множества данных) обладает или не обладает каждым из этих свойств. Запрос задает некоторое подмножество множества свойств, и необходимо найти все элементы библиотеки, которые обладают всеми свойствами из запроса.

2.15. Рассмотрим следующую задачу поиска, которая может возникнуть, например, при разгадывании кроссвордов. Элементы библиотеки (записи) есть слова фиксированной длины n в алфавите $\{0,1\}$. Запрос задает некоторый набор позиций и значения букв в этих позициях. Необходимо найти в библиотеке все записи, у которых в позициях, задаваемых запросом, стоят буквы, совпадающие с соответствующими значениями позиций запроса. Опишите тип, задающий эти задачи поиска. Сравните полученный тип с типом задач интервального поиска на булевом кубе (см. упражнение 2.12).

2.3.1 Критерий допустимости ИГ

Пусть нам дана ЗИП $I = \langle X, V, \rho \rangle$.

Скажем, что ИГ U *решает* ЗИП $I = \langle X, V, \rho \rangle$, если для любого запроса $x \in X$ ответ на этот запрос содержит все те и только те записи из V, которые удовлетворяют запросу x, то есть

$$\mathcal{J}_U(x) = \{y \in V : x \rho y\}.$$

ИГ U, решающий ЗИП I, будем также называть *допусти-*

мым для задачи I.

Пусть U — некоторый ИГ, y — запись из Y. Через $L_U(y)$ обозначим множество листьев ИГ U, которым соответствует запись y.

Справедлива следующая теорема.

Теорема 9. *ИГ U решает ЗИП $I = \langle X, V, \rho \rangle$ тогда и только тогда, когда для любой записи $y \in V$, такой, что $O(y, \rho) \neq \emptyset$, справедливо $L_U(y) \neq \emptyset$ и $\bigvee\limits_{\alpha \in L_U(y)} \varphi_\alpha = \chi_{y,\rho}$, а для любой записи $y \in V$, такой, что $O(y, \rho) = \emptyset$, справедливо либо $L_U(y) = \emptyset$, либо $\bigvee\limits_{\alpha \in L_U(y)} \varphi_\alpha \equiv 0$.*

Доказательство. *Достаточность.*

Возьмем произвольный запрос $x \in X$.

Возьмем произвольную запись $y \in V$.

Если $O(y, \rho) = \emptyset$ и, следовательно, $x \notin O(y, \rho)$, то по предположению либо $L_U(y) = \emptyset$, либо $\bigvee\limits_{\alpha \in L_U(y)} \varphi_\alpha \equiv 0$. Откуда следует, что $y \notin \mathcal{J}_U(x)$.

Теперь рассмотрим случай, когда $O(y, \rho) \neq \emptyset$.

Если $x \rho y$, то $\chi_{y,\rho}(x) = 1$, и согласно предположению

$$\bigvee_{\alpha \in L_U(y)} \varphi_\alpha(x) = 1.$$

Откуда следует, что существует лист $\alpha \in \mathcal{L}(U)$ такой, что $\varphi_\alpha(x) = 1$. Следовательно, $y \in \mathcal{J}_U(x)$.

Если $x \notin O(y, \rho)$, то $\chi_{y,\rho}(x) = 0$, и согласно предположению $\bigvee\limits_{\alpha \in L_U(y)} \varphi_\alpha(x) = 0$. Откуда следует, что $y \notin \mathcal{J}_U(x)$.

Таким образом, мы показали, что

$$\mathcal{J}_U(x) = \{y \in V : x \rho y\}$$

и тем самым доказали достаточность.

Необходимость.

Возьмем произвольную запись $y \in V$.

Если $O(y,\rho) = \emptyset$ и, следовательно, для любого запроса $x \in X$ $x \notin O(y,\rho)$, значит, для любого $x \in X$ $y \notin \mathcal{J}_U(x)$. Откуда следует, что либо $L_U(y) = \emptyset$, либо $\bigvee\limits_{\alpha \in L_U(y)} \varphi_\alpha \equiv 0$.

Теперь рассмотрим случай, когда $O(y,\rho) \neq \emptyset$.

Предположим, что для данной записи y не выполняются предположения теоремы, то есть существует такой запрос x, что

$$\bigvee_{\alpha \in L_U(y)} \varphi_\alpha(x) \neq \chi_{y,\rho}(x).$$

Но это означает, что y принадлежит в точности одному из множеств $\mathcal{J}_U(x)$ или $\{y \in V : x \rho y\}$.

Следовательно, при этом x

$$\mathcal{J}_U(x) \neq \{y \in V : x \rho y\},$$

и, значит, ИГ U не решает ЗИП I.

Тем самым теорема доказана. □

По сути теорема 9 говорит, что если нам дана ЗИП $I = \langle X, V, \rho \rangle$, и мы хотим построить ИГ, решающий эту ЗИП, мы должны построить многополюсник, который между корнем и остальными полюсами реализует как функции проводимости все функции $\chi_{y,\rho}$, где $y \in V$.

Упражнения

2.16. Пусть $S = \langle X, X, = \rangle$ — тип поиска идентичных объектов, множество предикатов F задается соотношением (2.7), базовое множество имеет вид $\mathcal{F} = \langle F, \emptyset \rangle$, $V = \{y_1, y_2, \ldots, y_k\} \subseteq X$. Приведите пример информационного графа над базовым множеством $\mathcal{F}$, решающего ЗИП $I = \langle X, V, = \rangle$.

2.17. Пусть $S = \langle X, X, = \rangle$ — тип поиска идентичных объектов, множество переключателей имеет вид

$$G = \{g_{=,a}(x) = \begin{cases} 1, & \text{если } x = a \\ 2, & \text{если } x \neq a \end{cases} : a \in X\}, \qquad (2.9)$$

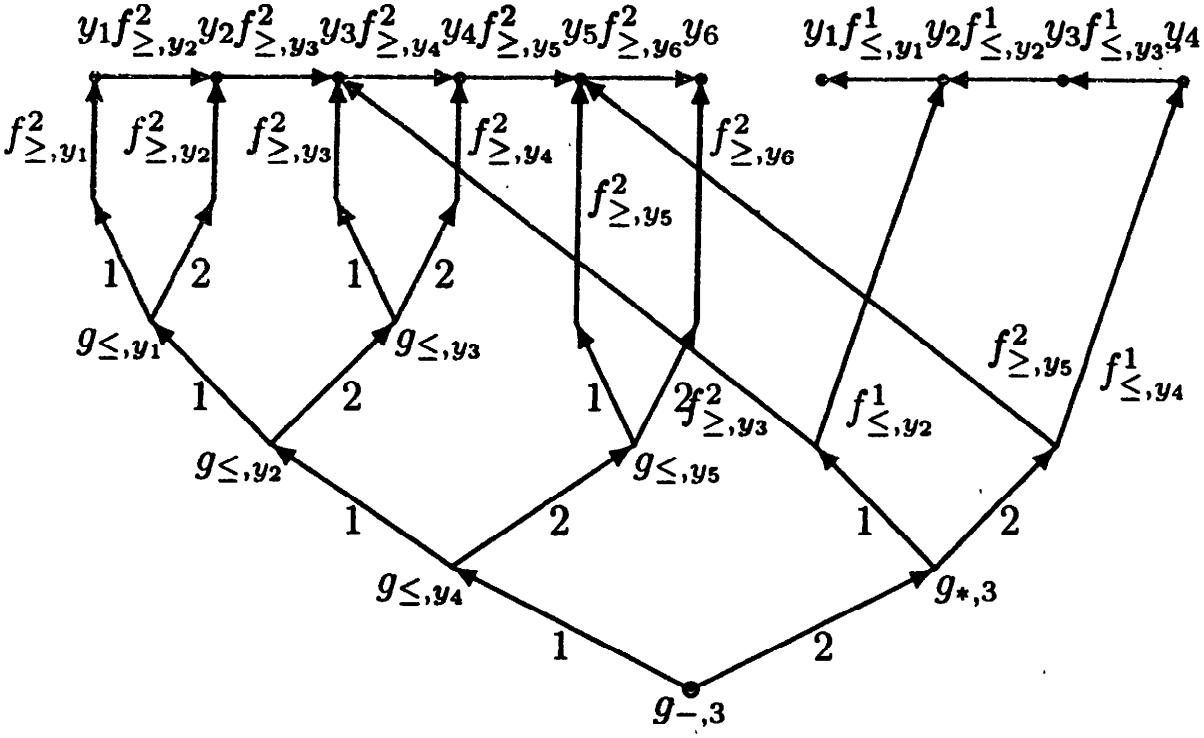

Рис. 2.3:

базовое множество имеет вид $\mathcal{F} = \langle \emptyset, G \rangle$, $V = \{y_1, y_2, \ldots, y_k\} \subseteq X$. Приведите пример информационного графа над базовым множеством $\mathcal{F}$, решающего ЗИП $I = \langle X, V, = \rangle$.

2.18. Пусть $X = \{1, 2, ..., N\}$, $S = \langle X, X, = \rangle$ — тип поиска идентичных объектов, множество переключателей имеет вид

$$G = \{g_a(x) = \begin{cases} 1, & \text{если } x < a \\ 2, & \text{если } x = a \\ 3, & \text{если } x > a \end{cases} : a \in X\}, \qquad (2.10)$$

базовое множество имеет вид $\mathcal{F} = \langle \emptyset, G \rangle$, $V = \{3, 5, 7, 11, 13, 17, 19\}$. Постройте информационный граф над базовым множеством $\mathcal{F}$, решающий ЗИП $I = \langle X, V, = \rangle$.

2.19. Пусть $X = \{1, 2, ..., N\}$, $S = \langle X, X, = \rangle$ — тип поиска идентичных объектов, $V = \{y_1, y_2, \ldots, y_k\} \subseteq X$. Предположим, что $y_1 < y_2 < \cdots < y_k$. Метод блочного поиска с размером блока m, решающий задачу $I = \langle X, V, = \rangle$, состоит в следующем. Если на вход алгоритма поиска подается запрос $x \in X$, то, начиная с $i = 1$ до $i = k/m$, просматриваем записи $y_{i \cdot m}$. Если $x > y_{i \cdot m}$, то увеличиваем i на 1, иначе по очереди просматриваем записи $y_{(i-1)m+1}, y_{(i-1)m+2}, \ldots, y_{i \cdot m}$ и сравниваем их с запросом x. При равенстве мы нашли нужную запись, если же ни для какой записи равенства не наблюдается, то ответ на запрос x пуст. Опишите базовое множество и постройте информационный граф над этим базовым множеством, который бы решал ЗИП $I = \langle X, V, = \rangle$ методом блочного поиска.

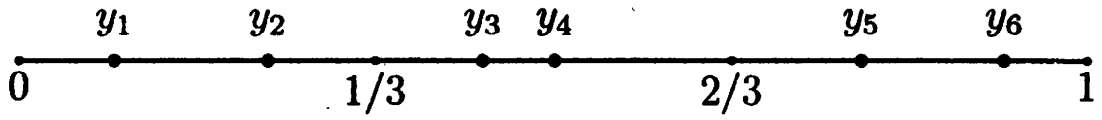

Рис. 2.4:

2.20. Пусть $X = \{1, 2, ..., N\}$, $V \subseteq X$, ρ_c — отношение поиска, задаваемое на $X \times V$ и определяемое соотношением

$$x\rho_c y \Longleftrightarrow (y \in V)\&(x \leq y)\&(\neg(\exists y')((y' \in V)\&(x \leq y')\&(y' < y))), \tag{2.11}$$

т.е. $x\rho_c y$, если $y \in V$, ближайшее справа к x. При выполнении этих условий ЗИП $I = \langle X, V, \rho_c \rangle$ называется задачей о близости. Пусть базовое множество имеет вид $\mathcal{F} = \langle \emptyset, G \rangle$, где множество переключателей G задается соотношением (2.8). Постройте информационный граф над базовым множеством $\mathcal{F}$, решающий ЗИП $I = \langle X, V, \rho_c \rangle$, если $V = \{3, 5, 7, 11, 13, 17, 19\}$.

2.21. Одномерная задача о доминировании задается типом $S_{dom1} = \langle [0, 1], [0, 1], \geq \rangle$. Пусть $V = \{y_1, y_2, \ldots, y_k\} \subseteq [0, 1]$. Опишите некоторое базовое множество и постройте какой-либо информационный граф над этим базовым множеством, который бы решал ЗИП $I = \langle [0, 1], V, \geq \rangle$.

2.22. Пусть $S_{int} = \langle X_{int}, Y_{int}, \rho_{int} \rangle$ — тип одномерного интервального поиска, где отношение ρ_{int} определяется соотношением (2.1), $V = \{y_1, y_2, \ldots, y_6\}$, где $y_1 = 1/6$, $y_2 = 1/4$, $y_3 = 3/8$, $y_4 = 2/5$, $y_5 = 3/4$, $y_6 = 7/8$. Решает ли информационный граф, изображенный на рисунке 2.3, где функции определяются соотношениями (2.2)–(2.6), задачу информационного поиска $I = \langle X_{int}, V, \rho_{int} \rangle$? Обоснуйте ответ.

2.23. Докажите, что информационный граф, изображенный на рисунке 2.1, решает одномерную задачу интервального поиска $I = \langle X_{int}, V, \rho_{int} \rangle$, где $V = \{y_1, y_2, y_3, y_4, y_5, y_6\}$ — библиотека, изображенная на рисунке 2.4.

2.24. Пусть $S_{int} = \langle X_{int}, Y_{int}, \rho_{int} \rangle$ — тип одномерного интервального поиска, где отношение ρ_{int} определяется соотношением (2.1), $V = \{1/8, 1/7, 1/5, 3/7, 3/5, 4/5, 7/8\}$. Опишите некоторое базовое множество и постройте какой-либо информационный граф над этим базовым множеством, который бы решал ЗИП $I = \langle X_{int}, V, \rho_{int} \rangle$.

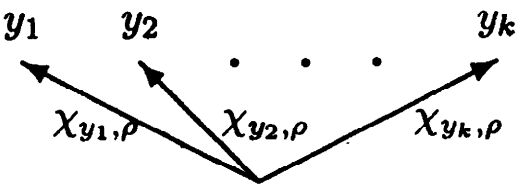

Рис. 2.5: Информационный граф переборного алгоритма

2.3.2 Полнота для информационных графов

Если нам дана ЗИП $I = \langle X, V, \rho \rangle$ и базовое множество $\mathcal{F} = \langle F, G \rangle$, то возникает вопрос, а можно ли построить информационный граф над базовым множеством $\mathcal{F}$, решающий эту задачу I? Если для любой записи $y_i \in V = \{y_1, \ldots, y_k\}$ $\chi_{y_i,\rho} \in F$, то ответ на этот вопрос положительный, и граф, изображенный на рисунке 2.5, решает задачу I.

В данном разделе дается более полный ответ на этот вопрос.

Пусть нам дан тип $S = \langle X, Y, \rho \rangle$, где X — множество запросов, Y — множество записей, ρ — отношение поиска, заданное на $X \times Y$, и базовое множество $\mathcal{F} = \langle F, G \rangle$.

Скажем, что базовое множество $\mathcal{F}$ *полно* для типа $S = \langle X, Y, \rho \rangle$, если для любой ЗИП $I = \langle X, V, \rho \rangle$ типа S существует ИГ U над базовым множеством $\mathcal{F}$, решающий ЗИП I.

Справедлив следующий результат, относящийся к проблеме полноты для ИГ.

Теорема 10. *Пусть заданы множества запросов X, записей Y, отношение поиска ρ на $X \times Y$ и базовое множество $\mathcal{F} = \langle F, G \rangle$, такое, что предикат тождественная 1 принадлежит множеству F. Тогда $\mathcal{F}$ будет полным для типа $S = \langle X, Y, \rho \rangle$ тогда и только тогда, когда для любой записи $y \in Y$ такой, что $O(y, \rho) \neq \emptyset$, функцию $\chi_{y,\rho}(x)$ можно представить формулой вида*

$$\chi_{y,\rho}(x) = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} f_{ij}(x),$$

где $f_{ij} \in F \cup \widehat{G}$.

Доказательство. *Достаточность.*

Пусть нам дана произвольная ЗИП $I = \langle X, V, \rho \rangle$, где $V = \{y_1, \ldots, y_k\} \subseteq Y$. По предположению каждую из функций

$\chi_{y_l,\rho}(x)$ $(l = \overline{1,k})$ можно представить формулой

$$\chi_{y_l,\rho}(x) = \bigvee_{i=1}^{n_l} \bigwedge_{j=1}^{m_{li}} f_{lij}(x),$$

где $f_{lij} \in F \cup \widehat{G}$, $l = \overline{1,k}$.

Без ограничения общности можно считать, что каждая конъюнкция $\bigwedge_{j=1}^{m_{li}} f_{lij}(x)$ содержит предикат из F, причем этот предикат стоит первым в конъюнкции. В противном случае мы всегда можем добавить предикат тождественная 1.

ИГ, решающий ЗИП I, будем строить следующим образом.

Сначала возьмем $k+1$ вершину и объявим одну из них корнем, а остальные объявим листьями и мысленно перенумеруем, начиная с 1 до k. Затем для каждого $l \in \{\overline{1,k}\}$ проделаем следующее.

Припишем l-му листу (обозначим его α_l) запись y_l.

Если $O(y_l,\rho) \neq \emptyset$, то проведем из корня в лист α_l n_l ориентированных цепей, причем i-я цепь $(i = \overline{1,n_l})$ будет состоять из m_{li} ребер. Теперь для каждого i $(i \in \{\overline{1,n_l}\})$ проделаем следующее. Если $f_{lij} \in F$, то j-е ребро i-й цепи объявим предикатным и припишем ему предикат f_{lij}. Если $f_{lij} \in \widehat{G}$ (то есть $f_{lij} = \xi_g^n$, где $g \in G$, а $n \in \mathbf{N}$), то вершину β, из которой исходит j-е ребро i-й цепи, объявим переключательной припишем ей переключатель g, j-му ребру i-й цепи припишем число n, из вершины β выпустим еще $r-1$ ребро, где r — мощность области значений переключателя g, и сопоставим им взаимно однозначно числа из множества $\{\overline{1,r}\} \setminus \{n\}$.

Нетрудно видеть, что в этом случае

$$\varphi_{\alpha_l}(x) = \bigvee_{i=1}^{n_l} \bigwedge_{j=1}^{m_{li}} f_{lij}(x) = \chi_{y_l,\rho}(x).$$

Поскольку это условие выполняется для всех листьев, то согласно теореме 9 построенный ИГ решает ЗИП I.

Необходимость.

Пусть $\mathcal{F}$ полно для отношения ρ. Возьмем произвольную запись $y \in Y$ такую, что $O(y,\rho) \neq \emptyset$. Пусть V — любая библиотека, содержащая запись y.

Так как $\mathcal{F}$ полно, то существует ИГ U, решающий ЗИП $I = \langle X, V, \rho\rangle$. Рассмотрим множество $L_U(y)$ листьев ИГ U, которым

соответствует запись y. Так как U решает задачу I, то согласно теореме 9

$$\bigvee_{\alpha \in L_U(y)} \varphi_\alpha(x) \equiv \chi_{y,\rho}(x).$$

Поскольку каждая из функций $\varphi_\alpha(x)$ есть функция проводимости от корня к листу α, а функция проводимости по определению есть дизъюнкция конъюнкций некоторых предикатов из $F \cup \widehat{G}$, то необходимость доказана, что и доказывает теорему. □

Упражнения

2.25. Пусть $S = \langle X, X, = \rangle$ — тип поиска идентичных объектов, базовое множество имеет вид $\mathcal{F} = \langle \emptyset, G \rangle$, где множество переключателей G задается соотношением (2.8). Будет ли полно базовое множество $\mathcal{F}$ для типа S?

2.26. Задача включающего поиска, описанная в упражнении 2.14, может быть задана типом $S_{bool} = \langle B^n, B^n, \overset{b}{\succeq} \rangle$, где B^n — n-мерный булев куб, $\overset{b}{\succeq}$ — отношение поиска на $B^n \times B^n$, определяемое следующим соотношением

$$(x_1, \ldots, x_n) \overset{b}{\succeq} (y_1, \ldots, y_n) \Longleftrightarrow x_i \geq y_i, \ i = \overline{1, n}. \qquad (2.12)$$

Приведите пример базового множества, полного для типа S_{bool}. Приведите пример минимального по мощности базового множества, полного для типа S_{bool}.

2.3.3 Сложность информационных графов

Из определения функционирования ИГ естественным образом вытекает, что каждому ИГ U можно сопоставить следующую процедуру поиска.

Предполагается, что эта процедура хранит в своей (внешней) памяти структуру ИГ U. Входными данными процедуры является запрос. Выходными данными является множество записей.

Пусть на вход процедуры поступил запрос x. Вводим понятие активного множества вершин и вносим в него в начальный момент корень ИГ U и помечаем его. Далее по очереди просматриваем вершины из активного множества и для каждой из них проделываем следующее:

- если рассматриваемая вершина — лист, то запись, приписанную вершине, включаем в ответ;
- если рассматриваемая вершина переключательная, то вычисляем на запросе x переключатель, соответствующий данной вершине, и если конец ребра, исходящего из рассматриваемой вершины, нагрузка которого равна значению переключателя, непомеченная вершина, то помечаем его и включаем в множество активных вершин;
- если рассматриваемая вершина предикатная, то просматриваем по очереди исходящие из нее ребра и вычисляем значения предикатов, приписанных этим ребрам, на запросе x. Концы ребер, которым соответствуют предикаты со значениями, равными 1, если они непомеченные, помечаем и включаем в множество активных вершин;
- исключаем рассматриваемую вершину из активного множества.

Процедура завершается по исчерпании активного множества.

Заметим, что если ИГ решает задачу I, то множество, полученное на выходе процедуры, будет содержать все те и только те записи библиотеки $\langle U \rangle$, которые удовлетворяют запросу x. То есть полученная процедура решает ЗИП $I = \langle X, V, \rho \rangle$, где $V = \langle U \rangle$, и, значит, является алгоритмом поиска.

Таким образом, ИГ как управляющая система может рассматриваться как модель алгоритма поиска, работающего над данными, организованными в структуру, определяемую структурой ИГ.

В данном разделе мы введем понятия сложности ИГ, которые будут характеризовать такие общепринятые меры сложности алгоритмов поиска как объем памяти, время поиска в худшем случае и время поиска в среднем.

Отметим, что в большинстве работ, посвященных исследованию сложности алгоритмов поиска, под сложностью алгоритма понимается время поиска в худшем случае, и в сравнительно редких случаях исследуется среднее время поиска, хотя для задач поиска, используемых в базах данных, для которых характерны массовость и многократность, исследование среднего времени поиска представляется более актуальным. Некоторое объяснение крена в сторону изучения сложности в худшем случае можно найти в цитате из [52, стр.20]: "К сожалению, анализ поведения в среднем значительно более сложная вещь, чем анализ худшего случая, по двум причинам: во-первых существенные математические трудности возникают, даже если удачно выбрано исходное распределение; во-вторых, часто с трудом достигается согласие в том, что именно выбранное распределение является реальной моделью изучаемой ситуации. Вот почему преобладающее большинство результатов связано с анализом худших случаев."

Определим понятие сложности ИГ на запросе.

Будем считать, что время вычисления любого переключателя из G примерно одинаково и характеризуется числом a, а время вычисления любого предиката из F — числом b.

Пусть нам дан некий ИГ U и произвольно взятый запрос $x \in X$.

Сложностью ИГ U на запросе x назовем число

$$T(U,x) = a \cdot \sum_{\beta \in \mathcal{P}} \varphi_\beta(x) + b \cdot \sum_{\beta \in \mathcal{R} \setminus \mathcal{P}} \psi_\beta \cdot \varphi_\beta(x).$$

Величина $T(U,x)$ характеризует время работы описанной выше процедуры поиска, сопоставленной ИГ U, поскольку $T(U,x)$ равно количеству переключателей, вычисленных данной процедурой при подаче на ее вход запроса x, умноженное на a, плюс количество вычисленных предикатов, умноженное на b.

Сложность ИГ можно вводить двумя способами. Во-первых, как максимальную сложность на запросе

$$\widehat{T}(U) = \max_{x \in X} T(U,x)$$

(здесь мы берем max, а не sup, так как $T(U,x)$ принимает целые значения, и, значит, sup всегда достигается). Эта величина ха-

рактеризует время поиска в худшем случае соответствующим ИГ алгоритмом и ее будем называть *B-сложностью ИГ* (верхней сложностью). Эта величина исследуется в большинстве работ, посвященных проблемам сложности задач поиска.

Во-вторых, можно вводить сложность ИГ как среднее значение сложности ИГ на запросе, взятое по множеству всех запросов. С этой целью введем *вероятностное пространство* над множеством запросов X, под которым будем понимать тройку $\langle X, \sigma, \mathbf{P}\rangle$, где σ — некоторая *алгебра подмножеств* множества X, $\mathbf{P}$ — *вероятностная мера* на σ, то есть аддитивная мера, такая, что $\mathbf{P}(X) = 1$.

В связи с тем, что мы ввели вероятностное пространство над множеством запросов, уточним понятие типа. А именно, под *типом* будем понимать тройку $S = \langle X, Y, \rho\rangle$, считая, что множество запросов X рассматривается вместе со своим вероятностным пространством $\langle X, \sigma, \mathbf{P}\rangle$. В тех же случаях, когда мы хотим явно выделить рассматриваемое вероятностное пространство над X, мы будем представлять тип пятеркой $S = \langle X, Y, \rho, \sigma, \mathbf{P}\rangle$.

Скажем, что базовое множество $\mathcal{F} = \langle F, G\rangle$ *измеримое*, если алгебра σ содержит все множества N_f, где $f \in F \cup \widehat{G}$.

Справедлива следующая лемма.

Лемма 5. *Если базовое множество $\mathcal{F} = \langle F, G\rangle$ измеримое, то для любого ИГ U над базовым множеством $\mathcal{F}$ функция $T(U, x)$, как функция от x, является случайной величиной.*

Доказательство. Нам необходимо доказать, что для любого ИГ U над базовым множеством $\mathcal{F}$ и любого действительного числа r множество $\{x \in X : T(U, x) < r\} \in \sigma$.

Покажем, что $(\beta \in \mathcal{R}(U)) \to (N_{\varphi_\beta} \in \sigma)$.

Пусть $\beta \in \mathcal{R}(U)$. Пусть C_β — множество всех ориентированных цепей ИГ U, ведущих из корня в вершину β. Пусть C — некоторая цепь, а c — некоторое ребро. Через $\theta(c)$ обозначим проводимость ребра c.

Нетрудно видеть, что

$$\varphi_\beta = \bigvee_{C \in C_\beta} \bigwedge_{c \in C} \theta(c).$$

Учитывая, что $N_{f\vee g} = N_f \cup N_g$, $N_{f\wedge g} = N_f \cap N_g$, имеем

$$N_{\varphi_\beta} = \bigcup_{C\in C_\beta} \bigcap_{c\in C} N_{\theta(c)} \in \sigma.$$

Введем следующее обозначение:

$$\mathcal{M}_r = \{\mathcal{B} \subset \mathcal{R}(U) : |\{\mathcal{B} \cap \mathcal{P}\}| + \sum_{\beta\in\mathcal{B}\setminus\mathcal{P}} \psi_\beta < r\}.$$

Тогда, как нетрудно видеть,

$$\{x \in X : T(U,x) < r\} = \bigcup_{\mathcal{B}\in\mathcal{M}_r} ((\bigcap_{\beta\in\mathcal{B}} N_{\varphi_\beta}) \bigcap (\bigcap_{\beta\in\mathcal{R}\setminus\mathcal{B}} (X\setminus N_{\varphi_\beta}))) \in \sigma.$$

Тем самым лемма доказана. □

Далее всюду будем предполагать, что базовое множество измеримо.

Сложностью ИГ U назовем математическое ожидание величины $T(U,x)$, то есть число

$$T(U) = \mathbf{M}_x\, T(U,x).$$

Если (β,α) — ребро ИГ, то *сложностью этого ребра* назовем число

- $b\cdot \mathbf{P}(N_{\varphi_\beta})$ — если (β,α) — предикатное ребро;
- $a\cdot \mathbf{P}(N_{\varphi_\beta})/\psi_\beta$ — если это ребро переключательное.

Если β — вершина ИГ, то число $\mathbf{P}(N_{\varphi_\beta})$ назовем *сложностью вершины β*.

Нетрудно показать, что сложность ИГ равна сумме сложностей ребер ИГ. В самом деле

$$\begin{aligned}
T(U) &= \mathbf{M}_x\, T(U,x) = \int_X T(U,x)\mathbf{P}(dx) = \\
&= \int_X (b\cdot \sum_{\beta\in\mathcal{R}\setminus\mathcal{P}} \psi_\beta\cdot\varphi_\beta(x) + a\cdot \sum_{\beta\in\mathcal{P}} \varphi_\beta(x))\mathbf{P}(dx) = \\
&= b\cdot \sum_{\beta\in\mathcal{R}\setminus\mathcal{P}} \psi_\beta \int_X \varphi_\beta(x)\mathbf{P}(dx) + a\cdot \sum_{\beta\in\mathcal{P}} \int_X \varphi_\beta(x)\mathbf{P}(dx) = \\
&= b\cdot \sum_{\beta\in\mathcal{R}\setminus\mathcal{P}} \psi_\beta \mathbf{P}(N_{\varphi_\beta}) + a\cdot \sum_{\beta\in\mathcal{P}} \mathbf{P}(N_{\varphi_\beta}).
\end{aligned}$$

Далее всюду будем предполагать, что $a = b = 1$.

Пусть нам дан ИГ U.

Объемом $Q(U)$ ИГ U назовем число ребер в ИГ U.

В качестве примера мы можем подсчитать сложность ИГ U, изображенного на рисунке 2.5. Легко видеть, что $Q(U) = k$ и $T(U) = k$, то есть объем графа минимально возможный, а время максимальное. Это и не удивительно, так как ИГ U соответствует переборному алгоритму поиска.

Пусть нам дана некая ЗИП I. *Сложностью задачи* I *при базовом множестве* $\mathcal{F}$ *и заданном объеме* q назовем число

$$T(I, \mathcal{F}, q) = \inf\{T(U) : U \in \mathcal{U}(I, \mathcal{F}) \text{ и } Q(U) \leq q\},$$

где $\mathcal{U}(I, \mathcal{F})$ — множество всех ИГ над базовым множеством $\mathcal{F}$, решающих ЗИП I.

Соответственно *В-сложностью задачи* I *при базовом множестве* $\mathcal{F}$ *и заданном объеме* q назовем число

$$\widehat{T}(I, \mathcal{F}, q) = \min\{\widehat{T}(U) : U \in \mathcal{U}(I, \mathcal{F}) \text{ и } Q(U) \leq q\}$$

(здесь мы берем min, а не inf, так как $\widehat{T}(U)$ принимает целые значения, и, значит, inf всегда достигается).

Число

$$T(I, \mathcal{F}) = \inf\{T(U) : U \in \mathcal{U}(I, \mathcal{F})\}$$

назовем *сложностью задачи* I *при базовом множестве* $\mathcal{F}$.

Соответственно *В-сложностью задачи* I *при базовом множестве* $\mathcal{F}$ назовем число

$$\widehat{T}(I, \mathcal{F}) = \min\{\widehat{T}(U) : U \in \mathcal{U}(I, \mathcal{F})\}.$$

Если k — натуральное число, S — тип задач поиска, то обозначим

$$\mathcal{I}(k, S) = \{I = \langle X, V, \rho\rangle \in S : |V| = k\}.$$

Будем исследовать функции, характеризующие сложность класса ЗИП $\mathcal{I}(k, S)$, такие как функции Шеннона:

$$\widehat{T}(k, S, \mathcal{F}) = \max_{I \in \mathcal{I}(k,S)} \widehat{T}(I, \mathcal{F}),$$

$$T(k, S, \mathcal{F}) = \sup_{I \in \mathcal{I}(k,S)} T(I, \mathcal{F}),$$

(в первом случае мы берем max, а не sup, так как $\widehat{T}(I, \mathcal{F})$ принимает целые значения, и, значит, sup всегда достигается).

Если существует такой ИГ $U \in \mathcal{U}(I, \mathcal{F})$, что $T(U) = T(I, \mathcal{F})$, то ИГ U будем называть *оптимальным* для ЗИП I. Соответственно если $\widehat{T}(U) = \widehat{T}(I, \mathcal{F})$, то ИГ U будем называть *B-оптимальным* для ЗИП I.

Можно привести **пример такой ЗИП и такого базового множества, для которых не существует оптимального ИГ.**

Пусть $X = Y = [0, 1]$ и на X задана равномерная вероятностная мера. Пусть отношение поиска есть отношение "$\geq$" для действительных чисел. Пусть библиотека состоит из одной записи $V = \{3/4\}$. Пусть базовое множество имеет вид $\mathcal{F} = \langle F, \emptyset \rangle$, где

$$F = \{f^1\} \cup \{f_a^2 : a \in (0, 1/2)\},$$

$$f^1(x) = \begin{cases} 0, & \text{если } x \in [0, 1/2] \\ 1, & \text{если } x \in (1/2, 1] \end{cases},$$

$$f_a^2(x) = \begin{cases} 0, & \text{если } x \in (a, 3/4) \\ 1, & \text{если } x \in [0, a] \cup [3/4, 1] \end{cases}.$$

Рассмотрим ЗИП $I = \langle X, V, \geq \rangle$. Поскольку

$$\chi_{3/4,\geq}(x) = \begin{cases} 0, & \text{если } x \in [0, 3/4) \\ 1, & \text{если } x \in [3/4, 1] \end{cases},$$

то $\chi_{3/4,\geq} = f^1 \& f_a^2$, для любого $a \in (0, 1/2)$. Рассмотрим ИГ U_a, состоящий из двух последовательно соединенных ребер, начало первого из которых есть корень ИГ, а конец второго — лист, которому приписана запись $3/4$, первому ребру соответствует предикат f_a^2, а второму — f^1. Нетрудно видеть, что $T(U_a) = 1 + a + 1/4 = 5/4 + a$. Очевидно, что $T(I, \mathcal{F}) = \inf\{T(U_a) : a \in (0, 1/2)\} = 5/4$, но не существует ИГ, чья сложность равна $5/4$.

Скажем, что некая вершина ИГ *достижима из корня на запросе* $x \in X$ или просто *достижимая на запросе* $x \in X$, если функция фильтра этой вершины на запросе x равна 1.

Скажем, что некая вершина ИГ *достижима из корня* или просто *достижимая*, если функция фильтра этой вершины не равна тождественному нулю, в противном случае вершину называем *недостижимой*. Скажем, что ребро ИГ *несущественное*, если выполняется хотя бы одно из следующих условий

- ребро исходит из недостижимой вершины,
- ребро является предикатным и входит в корень или в недостижимую вершину,
- ребро является предикатным и не принадлежит ни одной цепи, ведущей из корня в какой либо лист,
- ребро является переключательным, и начало этого ребра не принадлежит ни одной цепи, ведущей из корня в какой либо лист,
- ребро является переключательным, и число, приписанное этому ребру, больше максимально возможного значения переключателя, соответствующего началу этого ребра,
- начало и конец ребра совпадают.

В противном случае ребро называем *существенным.*

Легко заметить, что удаление несущественных ребер из ИГ не изменяет функционирования ИГ и не увеличивает его сложность. Поэтому всегда в дальнейшем мы будем рассматривать ИГ с точностью до несущественных ребер, а точнее будем считать, что все ребра в ИГ — существенные.

Теорема 11 (о существовании оптимальных графов). *Если множество запросов X конечно, то для любой ЗИП $I = \langle X, Y, \rho \rangle$ и любого базового множества $\mathcal{F} = \langle F, G \rangle$ такого, что $\mathcal{U}(I, \mathcal{F}) \neq \emptyset$, существует оптимальный ИГ над базовым множеством $\mathcal{F}$.*

Доказательство. Заметим, что из за конечности множества X множества F и G конечны. В самом деле, если $|X| = m$, то число предикатов, определенных на X не больше, чем 2^m. Следовательно, $|F| \leq 2^m$. Так как область значений любого переключателя есть начальный отрезок натурального ряда, то любой переключатель над X принимает не более m значений, следовательно $|G| < m^m$.

Для произвольного ИГ U обозначим через $N(U)$ подграф графа U, состоящий из ребер, имеющих ненулевую сложность.

Возьмем произвольный ИГ $U_0 \in \mathcal{U}(I, \mathcal{F})$. Обозначим $\mathcal{U}' = \{U \in \mathcal{U}(I, \mathcal{F}) : T(U) \leq T(U_0)\}$, $\mathcal{N}' = \{N(U) : U \in \mathcal{U}'\}$. Очевидно, что

$$T(I, \mathcal{F}) = \inf_{U \in \mathcal{U}(I,\mathcal{F})} T(U) = \inf_{U \in \mathcal{U}'} T(U) = \inf_{U \in \mathcal{N}'} T(U).$$

Для доказательства существования оптимального ИГ для ЗИП I нам достаточно показать конечность множества $\mathcal{N}'$.

Пусть $|F \cup \widehat{G}| = n$. Пусть M — множество, состоящее из тождественно нулевого предиката и всех предикатов, полученных из предикатов множества $F \cup \widehat{G}$ с помощью операций конъюнкции и дизъюнкции. Понятно, что $|M| \leq \min(2^m, 2^{2^n})$. Обозначим $M' = \{f \in M : \mathbf{P}(N_f) > 0\}$. Пусть $\min\limits_{f \in M'} \mathbf{P}(N_f) = r$. По определению $r > 0$. Поскольку для любого ИГ над $\mathcal{F}$ функции фильтров вершин принадлежат множеству M, то сложность любого предикатного ребра ИГ над $\mathcal{F}$ либо нулевая, либо не меньше чем r, а сложность любого переключательного ребра с ненулевой сложностью не меньше чем r/m. Отсюда в любом ИГ из множества $\mathcal{N}'$ число ребер не больше чем $T(U_0) \cdot m/r$.

Так как из конечности множеств F и G следует конечность числа различных нагрузок ИГ, то, значит, множество $\mathcal{N}'$ конечно.

Что и доказывает теорему. □

Упражнения

2.27. Пусть $X = \{1, 2, ..., N\}$, $S = \langle X, X, =, \mathbf{P}, \sigma \rangle$ — тип поиска идентичных объектов, где $\sigma = 2^X$, $\mathbf{P}$ — равномерная вероятностная мера, то есть для любого $x \in X$ выполняется $\mathbf{P}(x) = 1/N$.

1. Посчитайте сложность, B-сложность и объем информационного графа, полученного при решении упражнения 2.16.
2. Посчитайте сложность, B-сложность и объем информационного графа, полученного при решении упражнения 2.17.
3. Посчитайте сложность, B-сложность и объем информационного графа, полученного при решении упражнения 2.18. Для базового множества и ЗИП, приведенных в упражнении 2.18, постройте информационный граф со сложностью, не большей, чем 1.48.

4. Посчитайте сложность, B-сложность и объем информационного графа, полученного при решении упражнения 2.19, если $N = 100$. Для какого значения параметра m сложность будет минимальна. Для какого значения параметра m B-сложность будет минимальна. Для какого значения параметра m объем будет минимальным.

2.28. Если $X = \{1, 2, ..., N\}$ и на X задана равномерная вероятностная мера, то посчитайте сложность, B-сложность и объем информационного графа, полученного при решении упражнения 2.20.

2.29. Пусть на множестве запросов $X = [0, 1]$ задана равномерная вероятностная мера. Посчитайте сложность, B-сложность и объем информационного графа, полученного при решении упражнения 2.21.

2.30. Пусть на множестве запросов $X_{int} = \{(u, v) : 0 \leq u \leq v \leq 1\}$ задана равномерная вероятностная мера. Посчитайте сложность, B-сложность и объем информационного графа, изображенного на рисунке 2.1.

2.3.4 Мощностная нижняя оценка

В работе [52, стр. 92] бездоказательно, просто как очевидный факт утверждается, что время поиска по крайней мере не меньше чем время необходимое на перечисление ответа. В нашей модели этот факт находит свое доказательство и носит название мощностной нижней оценки. В связи повсеместностью применимости мощностной нижней оценки часто при оценке времени алгоритма поиска оценивают только разность между временем поиска и временем перечисления ответа (см., например, [52]).

Пусть нам даны произвольные множества запросов X, записей Y и отношение поиска ρ на $X \times Y$. Причем на множестве запросов задано вероятностное пространство $\langle X, \sigma, \mathbf{P} \rangle$.

Скажем, что базовое множество $\mathcal{F}$ *допустимо для ЗИП* I, если существует ИГ над базовым множеством $\mathcal{F}$, который решает ЗИП I.

Следующий результат, называемый мощностной нижней оценкой, справедлив для любой ЗИП при минимальных ограничениях. Смысл этого результата заключается в том, что время поиска не может быть меньше чем время, необходимое на перечисление ответа.

Справедлива следующая теорема.

Теорема 12 (мощностная нижняя оценка). *Пусть* $I = \langle X, V, \rho \rangle$ *— произвольная ЗИП, такая, что существует такая запись* $y \in V$, *что* $O(y, \rho) \neq \emptyset$, $\mathcal{F}$ *— измеримое базовое множество, допустимое для* I, *тогда*

$$T(I, \mathcal{F}) \geq \max(1, \sum_{y \in V} \mathbf{P}(O(y, \rho))),$$

$$\widehat{T}(I, \mathcal{F}) \geq \max_{x \in X} |\mathcal{J}_I(x)|.$$

Доказательство. Возьмем произвольный ИГ U, решающий задачу I. Такой граф существует, так как $\mathcal{U}(I, \mathcal{F}) \neq \emptyset$.

Возьмем произвольный запрос $x \in X$. Так как ИГ U решает ЗИП I, то ответ на запрос x

$$\mathcal{J}_U(x) = \mathcal{J}_I(x) = \{y \in V : x \rho y\}.$$

Возьмем произвольную запись $y \in \mathcal{J}_U(x)$. Поскольку запись y попала в ответ, то, значит, в ИГ U существует некий лист α, которому приписана запись y и такой, что $\varphi_\alpha(x) = 1$. А так как $\varphi_\alpha(x) = 1$ и никакой лист не совпадает с корнем, то существует цепь, ведущая из корня в лист α, проводимость которой равна 1, и в этой цепи есть ребро, ведущее в α, с проводимостью 1. Это ребро назовем проводящим ребром записи y. Понятно, что разным записям из $\mathcal{J}_U(x)$ соответствуют разные проводящие ребра, так как эти ребра ведут в разные листья. Если проводящее ребро записи предикатное, предикат, приписанный проводящему ребру, обязательно был вычислен перед тем, как мы попали в лист. Если проводящее ребро записи переключательное, то обязательно был вычислен переключатель, приписанный вершине, из которой исходит проводящее ребро. Причем такие переключатели для разных записей из $\mathcal{J}_U(x)$ будут разными, так как только одно из переключательных ребер, исходящих из одной вершины, может иметь проводимость, равную 1. Таким образом, каждой записи из $\mathcal{J}_U(x)$ можно сопоставить переключатель или предикат, вычисляемый непосредственно перед попаданием в соответствующий записи лист. Причем разным записям будут сопоставлены разные переключатели или предикаты. Отсюда следует, что

$$T(U, x) \geq |\mathcal{J}_I(x)|.$$

Следовательно,

$$\begin{aligned}
\widehat{T}(U) &= \max_{x\in X} T(U,x) \geq \max_{x\in X} |\mathcal{J}_I(x)|,\\
T(U) &= \mathbf{M}_x\, T(U,x) \geq \mathbf{M}_x\, |\mathcal{J}_I(x)| =\\
&= \int_X |\mathcal{J}_I(x)|\mathbf{P}(dx) = \int_X |\{y\in V : x\rho y\}|\mathbf{P}(dx) =\\
&= \sum_{y\in V}\int_{O(y,\rho)} \mathbf{P}(dx) = \sum_{y\in V} \mathbf{P}(O(y,\rho)).
\end{aligned}$$

А так как это неравенство выполняется для любого графа $U \in \mathcal{U}(I,\mathcal{F})$, то

$$\widehat{T}(I,\mathcal{F}) \geq \max_{x\in X} |\mathcal{J}_I(x)|,$$

$$T(I,\mathcal{F}) \geq \sum_{y\in V} \mathbf{P}(O(y,\rho)),$$

Так как в библиотеке V существует такая запись y, что $O(y,\rho) \neq \emptyset$, то в любом ИГ, решающем ЗИП I, существует хотя бы одна цепь, и соответственно хотя бы одно ребро исходит из корня. Следовательно, $T(I,\mathcal{F}) \geq 1$.

Тем самым теорема доказана. □

Упражнения

2.31. Пусть $X = \{1,2,...,N\}$, $S = \langle X, X, =, \mathbf{P}, \sigma\rangle$ — тип поиска идентичных объектов, где $\sigma = 2^X$, $\mathbf{P}$ — равномерная вероятностная мера, то есть для любого $x \in X$ выполняется $\mathbf{P}(x) = 1/N$, $V = \{3,5,7,11,13,17,19\}$. Приведите мощностную нижнюю оценку для ЗИП $I = \langle X, V, =\rangle$.

2.32. Пусть $S_{dom1} = \langle [0,1], [0,1], \geq, \mathbf{P}, \sigma\rangle$ — тип одномерной задачи о доминировании, где $\mathbf{P}$ — равномерная вероятностная мера на $[0,1]$, $V = \{y_1, y_2, \ldots, y_k\} \subseteq [0,1]$. Приведите мощностную нижнюю оценку для ЗИП $I = \langle [0,1], V, \geq\rangle$.

2.33. Пусть $S_{int} = \langle X_{int}, Y_{int}, \rho_{int}\rangle$ — тип одномерного интервального поиска и на множестве запросов $X_{int} = \{(u,v) : 0 \leq u \leq v \leq 1\}$ задана равномерная вероятностная мера. $V = \{y_1, y_2, \ldots, y_k\} \subseteq [0,1]$. Приведите мощностную нижнюю оценку для ЗИП $I = \langle X_{int}, V, \rho_{int}\rangle$. Оцените сверху полученную величину.

2.4 Поиск идентичных объектов

В данном разделе мы будем рассматривать задачу поиска идентичных объектов, которая в том или ином виде встречается во всех информационных системах и базах данных. Задача поиска идентичных объектов состоит в поиске в информационном массиве объекта, идентичного объекту-запросу.

Для задачи поиска идентичных объектов в рамках АДВ-модели (алгебраическое дерево вычислений) [66] справедлива логарифмическая теоретико-информационная нижняя оценка сложности для худшего случая [70]. Поэтому считается, что бинарный поиск является оптимальным по порядку для задачи поиска идентичных объектов, и это как бы закрывает проблему, но несмотря на это имеется много работ, посвященных исследованию алгоритмов поиска идентичных объектов в худшем случае [7, 71]. Связано это, во-первых, с проблемой поддержки сбалансированности бинарного дерева при операциях вставки и удаления, а во-вторых, с тем, что бинарный поиск хорош только тогда, когда целиком вся библиотека помещается во внутренней памяти (*внутренний поиск* [23]), если же библиотека вся или частично расположена на внешних носителях (*внешний поиск* [23]), то эффективность бинарного поиска сразу падает. Также много работ, посвященных изучению алгоритмов поиска идентичных объектов, имеющих хорошие временные характеристики в среднем [17, 77, 80]. Связаны они в основном с методом хеширования, на котором мы остановимся чуть подробнее.

Хеширование предполагает наличие хеш-функции. Хеш-функция $h(y)$ определена на множестве записей X и переводит его в множество $\{\overline{1,m}\}$, где m — параметр хеш-функции.

Обычно используется два основных метода хеширования. Первый — предложенный А. Думи [77] и называемый методом цепочек, предполагает наличие m списков. Тогда для поступившего запроса x (он же запись) вычисляется значение хеш-функции $h(x)$, и если оно равно i, то просматривается i-ый список и в нем ищется запись x. Если это поиск с занесением, то в случае неудачного поиска запись x добавляется к i-ому списку.

Второй метод хеширования предложен А. П. Ершовым [17] и называется открытой адресацией. Он предполагает наличие

закольцованного массива для m записей и наличие понятия пустой записи. После этого для поступившего запроса x вычисляется значение хеш-функции $h(x)$, и если оно равно i, то просматривается с i-ой позиции массив записей пока запись x будет найдена или пока не встретится пустая запись. Если это поиск с занесением, то в случае неудачного поиска на место встреченной пустой записи помещается запись x.

Методы хеширования хороши тем, что при удачном выборе хеш-функции, равномерно рассеивающей поступающие записи, время поиска в среднем будет очень малым. Вместе с тем Д. Кнут [23, стр. 641–642] усматривает три основных недостатка метода хеширования.

а) После неудачного поиска мы знаем лишь то, что нужной записи нет, тогда как с помощью бинарного поиска мы обнаруживаем ближайших соседей ненайденной записи, что часто бывает важно во многих приложениях.

б) Часто довольно трудно распределить память под хеш-таблицу. Если выделить слишком мало, то она может переполниться, и потребуется тягостное "рехеширование". Если выделить слишком много, то это расточительно.

в) "Наконец, при использовании методов хеширования нужно свято верить в теорию вероятностей, ибо они эффективны лишь в среднем, а худший случай просто ужасен!" — цитата из Д. Кнута [23, стр. 642]. Поэтому они не всегда подходят для работы в реальном масштабе времени, например, для управления движением транспорта, поскольку на карту поставлены человеческие жизни. Алгоритмы, использующие сбалансированные деревья, гораздо безопаснее, ведь они имеют гарантированную верхнюю границу времени поиска.

В данном разделе предлагается метод поиска идентичных объектов, который в среднем эффективен, как хорошие методы хеширования, то есть обеспечивает мгновенное решение, а в худшем случае такой же, как метод бинарного поиска, и плюс к этому не обладает недостатком а), то есть в случае неудачного поиска выходит к ближайшим соседям ненайденной записи, что позволяет использовать этот алгоритм для решения задач о близости.

Опишем формально задачу поиска идентичных объектов.

Пусть нам дано множество X, на котором задано отношение линейного порядка $\preceq$, то есть такое бинарное отношение на $X \times X$, которое для любых $x, y, z \in X$ удовлетворяет условиям

- рефлексивности $x \preceq x$;
- транзитивности $(x \preceq y)\&(y \preceq z) \to (x \preceq z)$;
- антисимметричности $(x \preceq y)\&(y \preceq x) \to (x = y)$;
- связности $(x \preceq y) \vee (y \preceq x)$.

Рассмотрим следующий тип задач поиска $S_{id} = \langle X, X, \rho_{id} \rangle$, где отношение поиска ρ_{id} есть отношение идентичности, то есть

$$x \rho_{id} y \Longleftrightarrow x = y.$$

Тип S_{id} будем называть *типом поиска идентичных объектов.*

2.4.1 Бинарный поиск

Если V — конечное подмножество X, то через $\max\limits_{y \in V} y$ будем обозначать такое $y_{max} \in V$, что для любых $y \in V$ $y \preceq y_{max}$, и через $\min\limits_{y \in V} y$ обозначим такое $y_{min} \in V$, что для любых $y \in V$ $y_{min} \preceq y$.

Пусть

$$g_{\preceq,a}(x) = \begin{cases} 1, & \text{если } x \preceq a \\ 2 & \text{в противном случае} \end{cases}, a \in X, \tag{2.13}$$

$$f_{=,a}(x) = \begin{cases} 0, & \text{если } x \neq a \\ 1, & \text{если } x = a \end{cases}, a \in X. \tag{2.14}$$

$$G_1 = \{g_{\preceq,a}(x) : a \in X\}, \tag{2.15}$$

$$F = \{f_{=,a}(x) : a \in X\}, \tag{2.16}$$

$$\mathcal{F}_{bin} = \langle F, G_1 \rangle. \tag{2.17}$$

Справедлива следующая теорема.

Теорема 13. *Если $I = \langle X, V, \rho_{id} \rangle$ — задача поиска идентичных объектов, то есть задача типа S_{id}, $\mathcal{F}_{bin}$ — базовое множество, определяемое соотношениями (2.13)–(2.17), то*

$$T(I, \mathcal{F}_{bin}, 2|V| - 1) \leq]\log_2 |V|[+ 1,$$

$$\widehat{T}(I, \mathcal{F}_{bin}, 2|V| - 1) \leq]\log_2 |V|[+ 1.$$

Доказательство. Пусть $|V| = k$.

Построим ИГ $D^1(V)$, имеющий вид дерева, следующим образом.

Возьмем бинарное сбалансированное дерево с k концевыми вершинами, высоты $]\log_2 k[$, все ребра которого ориентированы от корня к концевым вершинам. Если в этом дереве есть внутренняя вершина, из которой исходят ребра, ведущие одно в концевую вершину, а другое во внутреннюю (если такая вершина есть, то она только одна), то из концевой вершины, в которую ведет одно из этих ребер, выпустим одно ребро. Полученное дерево обозначим D. Объявим концевые вершины этого дерева листьями и сопоставим им слева направо в порядке возрастания элементы из V. (Говоря о дереве, мы подразумеваем некоторую его укладку, и направления "влево", "вправо" определяются уже на этой укладке.)

Пусть β — произвольная внутренняя вершина дерева D. Обозначим через V_β множество записей, соответствующих листьям, схемно достижимым из β. Пусть β' — вершина, в которую ведет левое (если оно одно, то единственное) ребро из β. Пусть

$$y_\beta = \max_{y \in V_{\beta'}} y.$$

Объявим все внутренние вершины дерева D, из которых исходят ребра, ведущие во внутренние вершины, вершинами переключения и для каждой такой вершины β левому ребру, из нее выходящему, припишем 1, а правому — 2, а самой вершине припишем переключатель $g_{\preceq, y_\beta}(x)$.

Все ребра дерева D, входящие в листья, объявим предикатными и припишем каждому такому ребру, ведущему в лист с записью y, предикат $f_{=,y}(x)$.

ИГ, полученный из дерева D после определения таким образом нагрузки, обозначим $D^1(V)$ и будем называть первым деревом бинарного поиска.

Покажем, что $D^1(V)$ решает задачу $I = \langle X, V, \rho_{id} \rangle$.

Возьмем произвольную запись $y \in V$. Пусть α — лист $D^1(V)$, которому приписана запись y. В этот лист ведет единственная цепь, состоящая из не более чем $]\log_2 k[$ ребер, причем все эти ребра, кроме последнего, переключательные. Последнее ребро в

цепи — предикатное с предикатом $f_{=,y}(x)$, и его проводимость на запросе y равна $f_{=,y}(y) = 1$. Покажем, что проводимость остальных ребер цепи на запросе y равна 1. Возьмем произвольно одно из этих ребер (β, β'). Если (β, β') — левое, исходящее из β , то $y \in V_{\beta'}$ и предикат, приписанный вершине β, $g_{\preceq, y_\beta}(y) = 1$, так как $y \preceq y_\beta = \max\limits_{y' \in V_{\beta'}} y'$. Если (β, β') — правое ребро, исходящее из β, то $y \not\preceq y_\beta$ и $g_{\preceq, y_\beta}(y) = 2$, тем самым проводимость ребра (β, β') в обоих случаях равна 1. Следовательно, $\varphi_\alpha(y) = 1$. Так как ребро, ведущее в лист α, имеет нагрузку $f_{=,y}$, то для любого запроса $x \neq y$ $\varphi_\alpha(x) = 0$.

Таким образом, мы показали, что $\varphi_\alpha(x) = f_{=,y}(x)$. Учитывая произвольность листа α и теорему 9, получим, что ИГ $D^1(V)$ решает задачу $I = \langle X, V, \rho_{id} \rangle$.

Подсчитаем объем ИГ $D^1(V)$.

Поскольку $D^1(V)$ есть дерево, в котором полустепень исхода любой вершины равна 2, кроме, быть может, одной вершины, то если в дереве $D^1(V)$ нет вершины с полустепенью исхода 1, то в $D^1(V)$ будет ровно $2 \cdot k - 2$ ребра, а если в дереве $D^1(V)$ есть вершина с полустепенью исхода 1, то в $D^1(V)$ будет $2 \cdot k - 1$ ребро. Отсюда следует, что

$$Q(D^1(V)) \leq 2k - 1. \tag{2.18}$$

Подсчитаем сложность ИГ $D^1(V)$.

Рассмотрим произвольный запрос $x \in X$. Как мы показали выше, активные на этом запросе вершины графа $D^1(V)$ (вершина β активна на запросе, если $\varphi_\beta(x) = 1$) образуют единственную цепь. Причем в этой цепи не более чем $]\log_2 k[-1$ переключательных вершин и одна вершина, из которой исходит не более двух ребер с предикатами. Тем самым

$$T(D^1(V), x) \leq]\log_2 k[+1.$$

Отсюда с учетом (2.18) сразу следует, что

$$T(I, \mathcal{F}_{bin}, 2k - 1) \leq]\log_2 k[+1,$$
$$\widehat{T}(I, \mathcal{F}_{bin}, 2k - 1) \leq]\log_2 k[+1.$$

Тем самым теорема доказана. □

Отметим, что первое дерево бинарного поиска соответствует стандартному алгоритму бинарного поиска в версии Боттенбрука [71, с. 214].

Упражнения

2.34. Пусть $I = \langle X, V, \rho_c \rangle$ — задача о близости, где $X = \{1, 2, ..., N\}$, $V \subseteq X$, ρ_c — отношение поиска, задаваемое соотношением на $X \times V$ и определяемое соотношением (2.11). По аналогии с поиском идентичных объектов постройте ИГ, решающий задачу о близости I методом бинарного поиска. Получите соответствующие оценки сложности.

2.4.2 Константный в среднем алгоритм поиска

Пусть $\mathbf{N}_0 = \mathbf{N} \cup \{0\}$ — множество целых неотрицательных чисел.

Пусть

$$L_1(l) = \begin{cases} 0, & \text{если } l = 0 \\ ]\log_2 l[+1, & \text{если } l = 1, 2, 3 \\ \log_2 l + 2, & \text{если } l \geq 4 \end{cases} - \qquad (2.19)$$

функция, определенная на множестве $\mathbf{N}_0$.

Докажем следующее вспомогательное утверждение.

Лемма 6. *Пусть $L_1(l)$ — функция, определенная выше, пусть $k, m \in \mathbf{N}$, пусть*

$$r_1(k,m) \stackrel{\text{def}}{=} \max\{\sum_{i=1}^{m} L_1(l_i) : l_1 \in \mathbf{N}_0, \ldots, l_m \in \mathbf{N}_0, \sum_{i=1}^{m} l_i = k\}.$$

Тогда

$$r_1(k,m) = \left(k - \left[\frac{k}{m}\right] \cdot m\right) \cdot L_1\left(\left[\frac{k}{m}\right] + 1\right) + \left(m - k + \left[\frac{k}{m}\right] \cdot m\right) \cdot L_1\left(\left[\frac{k}{m}\right]\right).$$

Доказательство. Если доопределить функцию $L_1(l)$ на положительную полуось числовой прямой, например, следующим образом:

$$L_1(x) = \begin{cases} x, & \text{если } 0 \leq x \leq 4 \\ \log_2 x + 2, & \text{если } x \geq 4 \end{cases},$$

то полученная функция будет непрерывной и вогнутой. И, вообще говоря, вогнутость этой функции объясняет результат леммы.

Более подробное доказательство будем вести от противного. Предположим, что лемма неверна, то есть

$$r_1(k,m) = \sum_{i=1}^{m} L_1(l'_i) > \left(k - \left[\frac{k}{m}\right]\cdot m\right)\cdot L_1\left(\left[\frac{k}{m}\right]+1\right) + \\ + \left(m - k + \left[\frac{k}{m}\right]\cdot m\right)\cdot L_1\left(\left[\frac{k}{m}\right]\right), \qquad (2.20)$$

и среди чисел $l'_i (i = \overline{1,m})$ существует 2 числа, разность которых не меньше двух.

Без ограничения общности можем считать, что $l'_1 - l'_2 \geq 2$.

Пусть

$$l''_1 = \left]\frac{l'_1 + l'_2}{2}\right[, \; l''_2 = \left[\frac{l'_1 + l'_2}{2}\right].$$

$l''_1 + l''_2 = l'_1 + l'_2$ и $l''_1 - l''_2 \leq 1$.

Так как функция $L_1(x)$ вогнутая, то по свойству вогнутых функций

$$L_1(l''_1) + L_1(l''_2) \geq L_1(l'_1) + L_1(l'_2). \qquad (2.21)$$

Если в неравенстве (2.21) неравенство строгое, то получили противоречие, так как $\sum_{i=1}^{m} L_1(l'_i)$ — не максимально; если же нет, то обозначим $l''_i = l'_i$ $(i = \overline{3,m})$ и перейдем к рассмотрению суммы

$$\sum_{i=1}^{m} L_1(l''_i) = \sum_{i=1}^{m} L_1(l'_i) = r_1(k,m).$$

Если среди чисел l''_i $(i = \overline{1,m})$ нет пары чисел, разность которых превышает 1, то получим противоречие с неравенством (2.20), если же есть, то опять выполним операцию, описанную выше, и избавимся от такой пары, и так будем делать до тех пор, пока либо не получим строгое неравенство в неравенстве (2.21), либо не придем к разбиению $l_1^{(n)}, \ldots, l_m^{(n)}$, такому, что

$$\sum_{i=1}^{m} L_1(l_i^{(n)}) = r_1(k,m),$$

и все они отличаются не более чем на 1, и тем самым получим противоречие с неравенством (2.20), что доказывает лемму 6. □

Пусть на X задано вероятностное пространство $\langle X, \sigma, \mathbf{P}\rangle$.

Пусть $g_m^1(x)$ — переключатель такой, что

$$g_m^1(x) = i, \text{ если } x \in X_i \ (i = \overline{1,m}), \tag{2.22}$$

где $X_1, \ldots, X_m$ — разбиение множества X (то есть $X = X_1 \cup \cdots \cup X_m$ и $X_i \cap X_j = \emptyset$, если $i \neq j$) такое, что

$$\mathbf{P}(X_i) \leq c/m \ (i = \overline{1,m}), \tag{2.23}$$

где c — постоянная, не зависящая от m.

Так как

$$\mathbf{P}(X) = \sum_{i=1}^{m} \mathbf{P}(X_i) \leq c,$$

то отсюда сразу следует, что $c \geq 1$.

Пусть

$$G_2 = \{g_m^1(x) : m \in \mathbf{N}\}, \tag{2.24}$$

$$\mathcal{F} = \langle F, G_1 \cup G_2\rangle. \tag{2.25}$$

Справедлива следующая теорема.

Теорема 14. *Пусть $I = \langle X, V, \rho_{id}\rangle$ — задача поиска идентичных объектов, то есть задача типа S_{id}, где $|V| = k$, $\mathcal{F}$ — базовое множество, определяемое соотношениями (2.13)–(2.16), (2.22)–(2.25), m — натуральное число, c — константа, определяемая соотношением (2.23), $L_1(l)$ — функция, определяемая соотношением (2.19). Тогда*

$$\begin{aligned} 1 \ \leq \ & T(I, \mathcal{F}, 2 \cdot k + m - 1) \leq \\ \leq \ & \frac{c}{m}\left(\left(k - \left[\frac{k}{m}\right] \cdot m\right) \cdot L_1\left(\left[\frac{k}{m}\right] + 1\right) + \right. \\ & \left. + \left(m - k + \left[\frac{k}{m}\right] \cdot m\right) \cdot L_1\left(\left[\frac{k}{m}\right]\right)\right) + 1. \end{aligned}$$

В частности,

$$1 \leq T(I, \mathcal{F}, (2 + c) \cdot k) \leq 2$$

и $T(I, \mathcal{F}) \sim 1$ при $k \to \infty$. Кроме того, для ИГ $U \in \mathcal{U}(I, \mathcal{F})$, на котором достигается верхняя оценка, $\widehat{T}(U) \leq 2 + \rceil \log_2 k[$.

Доказательство. Построим ИГ U_m^0, имеющий вид дерева, следующим образом.

Возьмем вершину β_0 и объявим ее корнем графа U_m^0. Выпустим из β_0 m ребер, припишем им числа от 1 до m, объявим β_0 точкой переключения и припишем ей переключатель $g_m^1(x)$.

Пусть $V_i = X_i \cap V$, $l_i = |V_i|$, $i = \overline{1,m}$.

Конец ребра с номером i обозначим β_i.

Для всех таких i, что $V_i \neq \emptyset$, выпустим из вершины β_i первое дерево бинарного поиска, описанное в разделе 2.4.1, то есть дерево $D^1(V_i)$.

Полученный ИГ с $|V| = k$ листьями обозначим U_m^0. Это граф над базовым множеством $\mathcal{F}$.

Покажем, что U_m^0 решает задачу $I = \langle X, V, \rho_{id} \rangle$.

Возьмем произвольную запись $y \in V$. Пусть $y \in V_i$ ($i \in \{\overline{1,m}\}$), то есть лист α, которому приписана запись y, принадлежит дереву $D^1(V_i)$. Так как $g_m^1(y) = i$, запрос y пройдет по i-му ребру, исходящему из корня, в корень дерева $D^1(V_i)$. Так как $D^1(V_i)$ решает задачу поиска идентичных объектов для библиотеки V_i, то только запрос y пройдет в лист α.

Таким образом, мы показали, что $\varphi_\alpha(x) = f_{=,y}(x)$. Учитывая произвольность выбора y и теорему 9, получим, что ИГ U_m^0 решает задачу $I = \langle X, V, \rho_{id} \rangle$.

Подсчитаем объем графа U_m^0.

$$Q(U_m^0) \leq m + \sum_{i=1}^{m} \max(0, 2 \cdot l_i - 1) \leq 2 \cdot k + m - 1.$$

Подсчитаем сложность ИГ U_m^0.

Рассмотрим произвольный запрос $x \in X$. Пусть $x \in X_i$ ($i = \overline{1,m}$).

$$T(U_0^m, x) = 1 + T(D^1(V_i), x) \leq 2 + \rceil \log_2 l_i [\leq 1 + L_1(l_i), \quad (2.26)$$

где $L_1(l)$ — функция, определяемая соотношением (2.19).

По определению и учитывая лемму 6,

$$\begin{aligned} T(U_0^m) &= \mathbf{M}_x\, T(U_0^m, x) = \int_X T(U_0^m, x)\, \mathbf{P}(dx) = \\ &= \sum_{i=1}^{m} \int_{X_i} T(U_0^m, x)\, \mathbf{P}(dx) \le \sum_{i=1}^{m} (1 + L_1(l_i)) \cdot \mathbf{P}(X_i) = \\ &= 1 + \sum_{i=1}^{m} L_1(l_i) \cdot \mathbf{P}(X_i) \le 1 + \frac{c}{m} \sum_{i=1}^{m} L_1(l_i) \le \\ &\le 1 + r_1(k, m)\, \frac{c}{m} = \\ &= \frac{c}{m} \left(\left(k - \left[\frac{k}{m} \right] \cdot m \right) \cdot L_1 \left(\left[\frac{k}{m} \right] + 1 \right) + \right. \\ &\quad \left. + \left(m - k + \left[\frac{k}{m} \right] \cdot m \right) \cdot L_1 \left(\left[\frac{k}{m} \right] \right) \right) + 1. \end{aligned}$$

Возьмем $m =]c \cdot k[$. При этом

$$Q(U_0^m) \le 2k +]c \cdot k[- 1 \le (2 + c) \cdot k.$$

Так как $c \ge 1$, то $m \ge k$. Если $m = k$, то $k/m = 1$ и

$$r_1(m, k) = 0 \cdot L_1(2) + k \cdot L_1(1) = k \cdot L_1(1).$$

Если $m > k$, то $k/m = 0$ и

$$r_1(m, k) = k \cdot L_1(1) + (m - k) \cdot L_1(0) = k \cdot L_1(1).$$

Следовательно,

$$T(I, \mathcal{F}, (2 + c) \cdot k) \le T(U_0^m) \le 1 + \frac{c}{]c\,k[}\, k \cdot L_1(1) \le 2.$$

Если взять $m =]k \cdot \gamma(k)[$, где $\gamma(k) \to \infty$ при $k \to \infty$ и $\gamma(k) > 1$ при любом k, то

$$T(I, \mathcal{F}) \le 1 + \frac{c \cdot k}{]k \cdot \gamma(k)[} \lesssim 1.$$

С другой стороны, $T(I, \mathcal{F}) \ge 1$, так как из корня должно исходить хотя бы одно ребро, поскольку $k > 0$.

И наконец, оценим В-сложность ИГ U_0^m, воспользовавшись соотношение (2.26),

$$\widehat{T}(U_0^m) = \max_{x \in X} T(U_0^m, x) \le 2 + \max_{1 \le i \le m}] \log_2 l_i [\le 2 +] \log_2 k [.$$

Тем самым теорема доказана. □

Следствие 1. *Если $\mathcal{F}$ — базовое множество, определяемое соотношениями (2.13)–(2.16), (2.22)–(2.25), k — натуральное число, то*

$$1 \le T(k, S_{id}, \mathcal{F}) \le \mathcal{T}(k, S_{id}, \mathcal{F}) \le 2$$

и при $k \to \infty$

$$T(k, S_{id}, \mathcal{F}) \sim \mathcal{T}(k, S_{id}, \mathcal{F}) \sim 1.$$

Приведем два примера задачи поиска идентичных объектов.

Пример 1. Пусть $S_{id1} = \langle X, X, \rho_{id} \rangle$ — тип поиска идентичных объектов, где $X = [0, 1]$, причем задано вероятностное пространство $\langle X, \sigma, \mathbf{P} \rangle$ над X такое, что $\mathbf{P}$ задается функцией плотностью вероятности $p(x)$.

Пусть

$$\begin{aligned} X_1 &= \{x \in X : 0 \le x \le 1/m\}. \\ X_i &= \{x \in X : \frac{i-1}{m} < x \le \frac{i}{m}\},\ i = \overline{2, m}. \end{aligned} \qquad (2.27)$$

Тогда переключатель $g^1_m(x) = \max(1,]x \cdot m[)$ удовлетворяет соотношению (2.22).

Если $p(x) \le c = const$, то $\mathbf{P}(X_i) \le c/m$, и мы находимся в условиях теоремы 14.

Приведем неформальное описание алгоритма поиска, описанного в доказательстве теоремы 14, применительно к задаче $I = \langle X, V, \rho_{id} \rangle$ типа S_{id1}, где $|V| = k$.

Разобьем отрезок $[0, 1]$ на m равных частей в соответствии с соотношениями (2.27).

Каждой части сопоставим подмножество множества V, состоящее из точек, принадлежащих этой части.

Теперь для произвольной точки-запроса x поиск точки из V, идентичной запросу, будем вести следующим образом.

Определим ту часть отрезка, которой принадлежит запрос x. Ее номер равен $\max(1,]x \cdot m[)$.

Далее во множестве, сопоставленном найденной части, осуществим обычный бинарный поиск.

Согласно лемме 6 наибольшее среднее время поиска будет в том случае, когда точки из множества V равномерно распределены по всем m частям отрезка.

Если в качестве m взять $m = k$, то случай, когда в каждую часть попадет по одной точке из V, будет иметь наибольшую сложность. А поскольку найти или не найти объект во множестве мощности 1 можно за 1 шаг, то в среднем за 2 шага (на первом мы определили номер нужной части) мы можем решить задачу поиска идентичных объектов.

Отметим, что в случае неудачного поиска алгоритм выводит к ближайшим соседям ненайденной точки.

Если внимательно присмотреться к описанному алгоритму, то можно заметить, что по своей структуре он напоминает хеширование по методу цепочек, в котором в качестве хеш-функции взято умножение, а цепочки-списки организованы в виде бинарного дерева.

Пример 2. Пусть $S_{id2} = \langle X, X, \rho_{id} \rangle$ — тип поиска идентичных объектов, где $X = \{1, \ldots, N\}$.

Пусть $\langle X, \sigma, \mathbf{P} \rangle$ — вероятностное пространство над X, где σ — множество всех подмножеств, а вероятностная мера $\mathbf{P}$ определяет равномерную вероятность на множестве запросов X, то есть для любого $x \in X$ $\mathbf{P}(x) = 1/N$.

Пусть задано $m \in \mathbf{N}$.

Пусть $r = N - m \cdot [N/m]$.

Пусть $X_1, \ldots, X_m$ — такое разбиение множества, что

$$X_i = \{x \in X : 1 + (i-1) \cdot ([N/m]+1) \le x \le i \cdot ([N/m]+1)\},$$
$$i = \overline{1, r},$$

$$X_i = \{x \in X : r \cdot ([N/m]+1) + 1 + (i-1-r) \cdot [N/m] \le x \le$$
$$\le r \cdot ([N/m]+1) + (i-r) \cdot [N/m]\}, \; i = \overline{r+1, m},$$

и $g_m^1(x) = i$, если $x \in X_i$, $i \in \{\overline{1, m}\}$. Тогда

$$\mathbf{P}(X_i) \le ([N/m]+1)/N < 2/m,$$

и мы опять находимся в условиях теоремы 14 при $c = 2$.

Упражнения

2.35. Пусть $I = \langle X, V, \rho_c \rangle$ — задача о близости, где $X = \{1, 2, ..., N\}$, $V \subseteq X$, ρ_c — отношение поиска, задаваемое соотношением на $X \times V$ и определяемое соотношением (2.11). По аналогии

с поиском идентичных объектов постройте ИГ, решающий задачу о близости I и имеющий константную сложность.

2.5 Одномерный интервальный поиск

Интервальный поиск имеет очевидное приложение к системам баз данных. В базе данных, содержащей записи о служащих некоторой компании, каждая запись имеет несколько атрибутов, таких, как возраст, жалование и т. д., и может рассматриваться как точка в n-мерном пространстве, в котором каждый атрибут соответствует измерению, а число измерений пространства n равно числу атрибутов. Типичная задача интервального поиска для двух измерений заключается в выявлении всех служащих, чьи возраст и жалованье находятся в заданных интервалах. Другой пример можно найти у Д. Кнута [23]. Он рассматривал базу данных городов США с координатами в виде широты и долготы. К такой базе естественен вопрос о перечислении всех городов, попадающих в некоторой прямоугольник-запрос. Это типичная двумерная задача интервального поиска.

Исследованию задачи интервального поиска (в другом переводе с английского — регионального поиска) посвящено большое количество работ [52, 67, 68, 79].

В данном разделе рассматривается следующая задача. Дано конечное множество точек из отрезка $[0,1]$. Запрос на поиск задает некий отрезок $[a,b] \subseteq [0,1]$. Надо перечислить все точки из множества, которые попадают в отрезок $[a,b]$. Это известная геометрическая задача поиска, описанная, например, в [52] и называемая одномерной задачей интервального поиска. В данном разделе исследуется вопрос, какие алгоритмы возникают, если ограничивать набор доступных средств, или, более формально, при различных базовых множествах. Получены также некоторые нижние оценки, с помощью которых показывается, что соответствующие полученные алгоритмы не могут быть существенно улучшены при данных ограничениях на набор доступных средств. Результаты данного раздела были опубликованы в [10].

Итак, в одномерной задаче интервального поиска множество записей Y_{int} есть отрезок $[0,1]$, множество запросов X_{int}

есть множество отрезков $[u,v] \subseteq [0,1]$, или множество пар точек (u,v), таких, что $0 \le u \le v \le 1$, то есть $X_{int} = \{x = (u,v) : 0 \le u \le v \le 1\}$.

На X_{int} задано вероятностное пространство $\langle X_{int}, \sigma, \mathbf{P}\rangle$, где σ — алгебра подмножеств множества X_{int}, содержащая все прямоугольники со сторонами параллельными осям координат и прямоугольные равнобедренные треугольники с катетами также параллельными осям координат, $\mathbf{P}$ — вероятностная мера на σ. Будем считать, что мера $\mathbf{P}$ определяется функцией плотности распределения вероятностей $p(u,v)$, то есть для любого $B \in \sigma$

$$\mathbf{P}(B) = \int_B p(u,v)\,du\,dv.$$

Для удобства будем считать, что $p(u,v)$ определена на всем квадрате $[0,1] \times [0,1]$, но $p(u,v) = 0$ при $(u,v) \notin X_{int}$.

Отношение поиска, которое будем обозначать через ρ_{int}, определяется соотношением

$$(u,v)\rho_{int}y \Longleftrightarrow u \le y \le v,$$

где $(u,v) \in X_{int}$, $y \in Y_{int}$.

Тип $S_{int} = \langle X_{int}, Y_{int}, \rho_{int}, \sigma, \mathbf{P}\rangle$ будем называть типом одномерного интервального поиска.

2.5.1 Логарифмический поиск

Пусть

$$F_1 = \{\chi_{a,\rho_{int}} : a \in Y_{int}\}, \tag{2.28}$$

$$G_1 = \{g_{\le,a}(u,v) = \begin{cases} 1, & \text{если } u \le a \\ 2, & \text{если } u > a \end{cases} : a \in [0,1]\}, \tag{2.29}$$

$$\mathcal{F}_1 = \langle F_1, G_1\rangle. \tag{2.30}$$

Отметим, что $N_f \in \sigma$ для любого предиката $f \in F_1 \cup \widehat{G}_1$, т.е. базовое множество $\mathcal{F}_1$ измеримо. Поскольку для любой записи $y \in Y_{int}$ $\chi_{y,\rho_{int}} \in F_1$, то $\mathcal{F}_1$ полно для типа S_{int}.

Справедлива следующая теорема.

Теорема 15. *Пусть* $I = \langle X_{int}, V, \rho_{int} \rangle$ *— ЗИП типа* S_{int}, *где* $V = \{y_1, \ldots, y_k\}$, *причем* $0 \leq y_1 \leq \ldots \leq y_k \leq 1$; $\mathcal{F}_1$ *— базовое множество, определяемое соотношениями (2.28) – (2.30). Тогда*

$$T(I, \mathcal{F}_1) \leq \sum_{i=1}^{k-1} \mathbf{P}(O(y_i, \rho_{int}))+] \log_2 k[+1.$$

Доказательство. Построим первое дерево бинарного поиска для библиотеки V, описанное в разделе 2.4.1, но только нагрузку переключательных вершин мы будем брать из множества G_2, а вместо предикатов $f_{=,a}$ использовать предикаты $\chi_{a,\rho_{int}} \in F_1$. Обозначим это дерево через D.

Для обозначения V_β будет использован тот же смысл, что и в разделе 2.4.1.

Для удобства дальнейшего изложения назовем множество переключательных ребер дерева D переключательной частью дерева D, а множество предикатных ребер — предикатной частью дерева D.

Обозначим лист, которому соответствует запись y_i, через α_i $(i = \overline{1,k})$. Из каждого листа α_i $(i = \overline{1,k-1})$ выпустим ребро, ведущее в лист α_{i+1}, и припишем ему предикат $\chi_{y_{i+1},\rho_{int}} \in F_1$.

Это множество из $(k-1)$-го ребра назовем правосторонней концевой цепью.

Полученный граф обозначим через U_V.

Отметим одно достаточно очевидное свойство графа U_V. Пусть β — вершина графа U_V, не являющаяся листом. Пусть $V_\beta = \{y_i, y_{i+1}, \ldots, y_j\}$. Тогда

- если $i = 1$ и $j = k$, то $N_{\varphi_\beta} = X_{int}$;
- если $i = 1$ и $j \neq k$, то$N_{\varphi_\beta} = \{(u,v) \in X_{int} : 0 \leq u \leq y_j\}$;
- если $i \neq 1$ и $j = k$, то $N_{\varphi_\beta} = \{(u,v) \in X_{int} : y_{i-1} < u \leq 1\}$;
- если $i \neq 1$ и $j \neq k$, то $N_{\varphi_\beta} = \{(u,v) \in X_{int} : y_{i-1} < u \leq y_j\}$.

Покажем, что U_V решает ЗИП I. По теореме 9, для этого достаточно показать, что $\varphi_{\alpha_i} = \chi_{y_i,\rho_{int}}$ для любого $i \in \{\overline{1,k}\}$.

Доказывать это будем индукцией по i.

Базис индукции. $i = 1$.

В лист α_1 ведет единственное ребро из вершины, которую обозначим через β'. Очевидно, $\varphi_{\alpha_1} = \varphi_{\beta'} \& \chi_{y_1, \rho_{int}}$. Через y_j обозначим максимальную запись в библиотеке $V_{\beta'}$. Примем $c = y_j$, если $j \neq k$, и $c = 1$, если $j = k$. Очевидно, $c \geq y_1$. Тогда

$$\begin{aligned} N_{\varphi_{\alpha_1}} &= \{(u,v) \in N_{\varphi_{\beta'}} : u \leq y_1,\ v \geq y_1\} = \\ &= \{(u,v) \in X_{int} : 0 \leq u \leq c,\ u \leq y_1,\ v \geq y_1\} = \\ &= \{(u,v) \in X_{int} : 0 \leq u \leq y_1,\ v \geq y_1\} = O(y_1, \rho_{int}), \end{aligned}$$

то есть $\varphi_{\alpha_1} = \chi_{y_1, \rho_{int}}$.

Индуктивный переход. Предположим, что для некоторого $i \geq 1$ функция фильтра листа α_i равна $\varphi_{\alpha_i} = \chi_{y_i, \rho_{int}}$.

Рассмотрим лист α_{i+1} $(i+1 \leq k)$. В него ведут два ребра: из листа α_i и некоторой вершины β, не являющейся листом. Пусть y_l и y_j — минимальная и максимальная записи в библиотеке V_β соответственно. Пусть

$$c_1 = \begin{cases} y_{l-1}, & \text{если } l \neq 1, \\ 0, & \text{если } l = 1, \end{cases}$$

$$c_2 = \begin{cases} y_j, & \text{если } j \neq k, \\ 1, & \text{если } j = k, \end{cases}$$

Очевидно, что $c_1 \leq y_i$ и $c_2 \geq y_{i+1}$. Тогда

$$\varphi_{\alpha_{i+1}} = (\varphi_{\alpha_i} \& \chi_{y_{i+1}, \rho_{int}}) \vee (\varphi_\beta \& \chi_{y_{i+1}, \rho_{int}}) = (\varphi_{\alpha_i} \vee \varphi_\beta) \& \chi_{y_{i+1}, \rho_{int}},$$

или

$$\begin{aligned} N_{\varphi_{\alpha_{i+1}}} &= (\{(u,v) \in X_{int} : u \leq y_i,\ v \geq y_i\} \cup \\ &\quad \cup \{(u,v) \in X_{int} : c_1 < u \leq c_2\}) \cap \\ &\quad \cap \{(u,v) \in X_{int} : u \leq y_{i+1},\ v \geq y_{i+1}\} = \\ &= \{(u,v) \in X_{int} : u \leq y_{i+1},\ v \geq y_{i+1}\} = O(y_{i+1}, \rho_{int}). \end{aligned}$$

Тем самым мы показали, что U_V решает I.

Покажем, что

$$T(U_V) \leq \sum_{i=1}^{k-1} \mathbf{P}(O(y_i, \rho_{int})) +] \log_2 k [+ 1.$$

Рассмотрим произвольный запрос $x \in X_{int}$. Поскольку через переключательную часть дерева D пролегает единственная проводящая цепь, и ее длина не превышает $] \log_2 k [- 1$, то на любом

запросе будет вычислено не более чем $]\log_2 k[-1$ переключателей. Далее на запросе x вычисляются один или два предиката, соответствующих предикатным ребрам из предикатной части дерева D, растущим из вершины, в которую ведет проводящая цепь. Таким образом, суммарная сложность дерева D не превышает $]\log_2 k[+1$.

Осталось подсчитать сложность правосторонней концевой цепи. Из каждого листа α_i $(i = \overline{1, k-1})$ исходит одно ребро и его сложность равна $\mathbf{P}(O(y_i, \rho_{int}))$.

Следовательно,

$$T(I, \mathcal{F}_2) \leq T(U_V) \leq \sum_{i=1}^{k-1} \mathbf{P}(O(y_i, \rho_{int}))+]\log_2 k[+1.$$

Тем самым теорема 15 доказана. □

Алгоритм поиска, соответствующий графу U_V, построенному в доказательстве теоремы 15, состоит в следующем. В упорядоченной по возрастанию библиотеке с помощью бинарного поиска за $\log_2 k$ шагов находится самая левая запись, находящаяся не левее левого конца запроса. Затем слева направо, начиная с найденной записи, просматриваются записи и сравниваются с правым концом запроса, и если оказывается, что очередная запись не больше правого конца, то эта запись включается в ответ, а если больше, то поиск прекращается, то есть — это традиционный алгоритм решения данной задачи поиска с помощью структуры данных, называемой прошитым двоичным деревом [52]. Там же [52, стр. 93] утверждается, что описанный алгоритм оптимален как по времени поиска, так и по памяти, но имеется в виду время в худшем случае, а оптимальность понимается по порядку.

2.5.2 Сверхлогарифмический поиск

Пусть

$$G_2 = \{ \; g_{-,m}(u, v) = \\ = \begin{cases} 1, & \text{если } 0 \leq v - u < 1/m \\ 2 & \text{в противном случае} \end{cases} : m \in \mathbf{N}\}, \quad (2.31)$$

$$\mathcal{F}_2 = \langle F_1, G_1 \cup G_2 \rangle, \quad (2.32)$$

где F_1, G_1 определяются соотношениями (2.28), (2.29).

Понятно, что для $N_f \in \sigma$ для любого $f \in \widehat{G}_2$.

Справедлива следующая теорема.

Теорема 16. *Пусть функция плотности распределения вероятностей $p(u,v)$, определяющая меру $\mathbf{P}$ вероятностного пространства над множеством запросов X_{int}, такая, что $p(u,v) \le c = const$; $I = \langle X_{int}, V, \rho_{int}\rangle$ — произвольная ЗИП типа S_{int}; $\mathcal{F}_2$ — базовое множество, определяемое соотношением (2.32). Тогда*

$$T(I,\mathcal{F}_2) \le \sum_{y\in V} \mathbf{P}(O(y,\rho_{int})) + \log_2\log_2 k + 6 + \log_2 c.$$

Доказательство. Упорядочим записи в библиотеке $V = \{y_1,\dots,y_k\}$ так, что $y_1 \le y_2 \le \dots \le y_k$.

Пусть m — некоторый параметр, значение которого определим позже.

Введем множество номеров $S = \{s_1,\dots,s_{m-1}\}$ такое, что s_i — номер записи из V такой, что y_{s_i} — ближайшая слева запись к точке i/m ($i = \overline{1,m-1}$), а если такой записи не существует, то $s_i = 0$.

Возьмем точку, объявим ее корнем графа и обозначим β_0. Выпустим из корня два ребра, одно из которых будем считать левым, а другое правым. Конец левого ребра обозначим β_1, а конец второго — β_2.

Припишем корню переключатель $g_{-,m}(u,v)$ из множества G_3. Левому ребру припишем 1, а правому 2.

Построим для V граф U_V так, как это было сделано в доказательстве теоремы 15 и совместим его корень с вершиной β_1.

Построим ориентированное сбалансированное бинарное дерево с $m-1$ концевыми вершинами с корнем в вершине β_2, все ребра которого ориентированы от β_2 к концевым вершинам. Обозначим это дерево через D_{m-1}. Из каждой внутренней вершины дерева D_{m-1} исходит 2 ребра, одному из них припишем 1, а другому — 2, т.е. все ребра дерева D_{m-1} будут переключательными. Тогда каждой концевой вершине дерева D_{m-1} можно сопоставить слово из символов 1 и 2 в соответствии с ребрами, встречающимися в цепи от корня к данному листу,

причем первый символ слова соответствует ребру, исходящему из корня, а последний ребру, входящему в концевую вершину. Занумеруем концевые вершины в соответствии с лексикографическим порядком слов, соответствующих концевым вершинам, и обозначим их через $\beta'_1, \ldots, \beta'_{m-1}$, т.е. вершине β'_1 соответствует слово $11\ldots1$, а вершине β'_{m-1} слово $22\ldots2$. Нагрузку внутренних вершин дерева D_{m-1} переключателями из G_1 определим следующим образом. Пусть β — произвольная внутренняя вершина дерева D_{m-1}. Пусть β'_j — концевая вершина с максимальным номером в ветви, растущей из вершины, в которое ведет ребро с номером 1, исходящее из β. Тогда вершине β припишем переключатель $g_{\leq,j/m}$. Такую операцию проделаем для всех внутренних вершин дерева, и полностью определим нагрузку вершин и ребер дерева D_{m-1}.

Напомним, что в графе U_V листья обозначаются $\alpha_1, \ldots, \alpha_k$ и им приписаны соответственно записи $y_1, \ldots, y_k$. Возьмем k новых точек, объявим их листьями и обозначим $\alpha'_1, \ldots, \alpha'_k$. Каждому листу α_i $(i = \overline{1,k})$ припишем запись y_i (тем самым каждая запись y_i будет приписана двум листьям — α_i и α'_i). Из каждого листа α'_i $(i = \overline{2,k})$ выпустим ребро, ведущее в лист α'_{i-1}, и припишем ему предикат $\chi_{y_{i-1},\rho_{int}} \in F_1$.

Это множество из $(k-1)$-го ребра назовем левосторонней концевой цепью.

Теперь для каждой вершины β'_i проделаем следующие действия. Если $s_i \neq 0$, то из β'_i выпустим ребро, ведущее в лист α'_{s_i}, и припишем ему предикат $\chi_{y_{s_i},\rho_{int}} \in F_1$. Если $s_i < k$, то из β'_i выпустим ребро, ведущее в лист α_{s_i+1}, и припишем ему предикат $\chi_{y_{s_i+1},\rho_{int}} \in F_1$.

Это множество ребер, исходящих из вершин β'_i $(i = \overline{1,m-1})$, назовем правой предикатной частью.

Полученный таким образом ИГ обозначим через U_m.

Покажем, что граф U_m решает одномерную задачу интервального поиска $I = \langle X_{int}, V, \rho_{int} \rangle$.

Рассмотрим произвольную запись $y_i \in V$. По построению $L_{U_m}(y_i) = \{\alpha_i, \alpha'_i\}$. В каждый из листьев α_i и α'_i ведут только ребра, которым приписан предикат $\chi_{y_i,\rho_{int}}$, следовательно,

$$N_{\varphi_{\alpha_i} \vee \varphi_{\alpha'_i}} \subseteq O(y_i, \rho_{int}). \qquad (2.33)$$

Пусть $x = (u, v)$ — произвольный запрос, такой что $u \le y_i \le v$, то есть $x \in O(y_i, \rho_{int})$. Покажем, что $\varphi_{\alpha_i}(x) \vee \varphi_{\alpha'_i}(x) = 1$.

Обозначим $A_a = \{x = (u, v) : 0 \le v - u \le a\}$.

Рассмотрим сначала случай, когда $x \in A_{1/(m+1)}$.

Тогда проводимость ребра (β_0, β_1) равна единице, а ребра (β_0, β_2) — нулю. Поскольку любая цепь, ведущая в лист α'_i проходит через ребро (β_0, β_2), то $\varphi_{\alpha'_i}(x) = 0$.

Обозначим через G'_β подграф графа U_m, состоящий из цепей, исходящих из вершины β.

Так как проводимость ребра (β_0, β_1) на запросе x равна 1, то мы выходим в вершину β_1. По построению граф G'_{β_1} совпадает с U_V и, следовательно, $\varphi_{\alpha_i}(x) = 1$, так как U_V решает ЗИП I.

Теперь рассмотрим случай, когда $x \notin A_{1/(m+1)}$.

В этом случае $g_{-,m}(x) = 2$ и проводимость ребра (β_0, β_1) будет равна 0, а ребра (β_0, β_2) равна 1, то есть мы попадем в вершину β_2, а проводимость всех цепей, проходящих через (β_0, β_1) равна 0. Из вершины β_2 растет дерево D_{m-1}, и оно обладает тем свойством, что из каждой его внутренней вершины исходит 2 переключательных ребра, и, следовательно, всегда проводимость ровно одного из ребер равна 1. Отсюда следует, что для любого запроса всегда существует единственная цепь, соединяющая β_2 с некоторой вершиной β'_j, проводимость которой равна 1. Таким образом, после прохождения дерева мы попадем в некоторую вершину β'_j, причем по построению этого дерева эта вершина, такая, что j является минимальным числом, таким что $j/m \ge u$.

Так как $v - u \ge 1/m$, то точка j/m принадлежит отрезку $[u, v]$.

Рассмотрим два случая.

1) $y_i \le j/m$.

Тогда $u \le y_i \le y_{s_j} \le v$, так как y_{s_j} — ближайшая слева к j/m запись из библиотеки V. Отсюда следует, что проводимость ребра, ведущего из β'_j в лист α'_{s_j} равна 1. Осталось заметить, что проводимость части левосторонней концевой цепи, ведущей из α'_{s_j} в α'_i, также будет равна 1, так как $0 \le y_i \le y_{s_j} \le v$.

Отметим также, что в этой ситуации $\varphi_{\alpha_i}(x) = 0$, так как в лучшем случае мы попадем в вершины правосторонней концевой цепи в точке α_{s_j+1}, которая в этой цепи находится после

точки α_i.

2) $y_i > j/m$.

Тогда $u \le y_{s_j+1} \le y_i \le v$. Отсюда следует, что проводимость ребра, ведущего из β'_j в лист α_{s_j+1}, равна 1, и проводимость части правосторонней концевой цепи, ведущей из α_{s_j+1} в α_i, также равна 1 на запросе x.

Аналогично предыдущему случаю $\varphi_{\alpha'_i}(x) = 0$.

Тем самым мы показали, что для любой записи $y_i \in V$ и любого запроса $\in X_{int}$ такого, что $x\rho_{int}y_i$ в графе U_m существует проводящая на запросе x цепь, ведущая из корня в какой-либо (но ровно в один) из листьев α_i и α'_i.

Что и доказывает, что граф U_m решает задачу I.

Подсчитаем сложность графа U_m.

Рассмотрим сначала произвольный запрос $x \in A_{1/m}$.

В этом случае

$$T(U_m, x) \le 1 + (\lceil \log_2 k \rceil - 1) + 2 + |\mathcal{J}_{U_m}(x)|.$$

Здесь первое слагаемое соответствует вычислению переключателя $g_{-,m}$ в вершине β_0. Второе слагаемое дают переключатели, входящие в единственную проводящую цепь, пролегающую через переключательную часть дерева D. Третье слагаемое соответствует вычислению одного или двух предикатов, соответствующих предикатным ребрам из предикатной части дерева D, растущим из вершины, в которую ведет проводящая цепь. Четвертое слагаемое соответствует вычислению предикатов, соответствующих ребрам, исходящим из листьев, записи которых входят в ответ (по одному на каждую запись).

Рассмотрим случай, когда $x \in X \backslash A_{1/m}$.

Тогда

$$T(U_m, x) \le 1 + \lceil \log_2(m-1) \rceil + 2 + |\mathcal{J}_{U_m}(x)|.$$

Здесь первое слагаемое соответствует вычислению переключателя $g_{-,m}$. Второе слагаемое дают переключатели, входящие в единственную проводящую цепь, пролегающую через дерево D_{m-1}. Третье слагаемое соответствует вычислению одного или двух предикатов, приписанных ребрам, исходящим из той вершины β'_j, в которую ведет проводящая цепь дерева D_{m-1}. И, наконец, четвертое слагаемое, как и ранее, соответствует вычислению предикатов, соответствующих ребрам, исходящим из

листьев, записи которых входят в ответ. Как мы показали ранее, для любой записи y_i, вошедшей в ответ, ровно у одного из листьев α_i и α_i' функция фильтра будет равна 1, и, следовательно, каждой записи соответствует ровно одно ребро и соответственно ровно один вычисленный предикат.

Подсчитаем сложность графа U_m.

$$\begin{aligned} T(U_m) &= \mathbf{M}_x\, T(U_m, x) = \mathbf{P}(A_{1/m}) \cdot (2+]\log_2 k[) + \\ &\quad +\mathbf{P}(X \backslash A_{1/m}) \cdot (3+]\log_2(m-1)[) + \mathbf{M}_x\, |\mathcal{J}_{U_m}(x)| \leq \\ &\leq \mathbf{P}(A_{1/m}) \cdot [\log_2 k] + 3 + [\log_2 m] + \\ &\quad + \sum_{y \in V} \mathbf{P}(O(y, \rho_{int})) \leq \\ &\leq c \cdot [\log_2 k] \left(\frac{2}{m} - \frac{1}{m^2} \right) + 3 + [\log_2 m] + \\ &\quad + \sum_{y \in V} \mathbf{P}(O(y, \rho_{int})) \leq \\ &\leq \frac{2c \cdot [\log_2 k]}{m} + 3 + [\log_2 m] + \sum_{y \in V} \mathbf{P}(O(y, \rho_{int})). \end{aligned}$$

Поскольку граф U_m решает задачу I, то

$$\mathcal{J}_{U_m}(x) = \{y \in V : x \rho_{int} y\},$$

и следовательно, используемое выше равенство

$$\mathbf{M}_x\, |\mathcal{J}_{U_m}(x)| = \sum_{y \in V} \mathbf{P}(O(y, \rho_{int}))$$

доказывается простой сменой порядка суммирования.

Установив значение параметра $m =]2c \log_2 k[$, получим

$$T(U_m) \leq \sum_{y \in V} \mathbf{P}(O(y, \rho_{int})) + \log_2 \log_2 k + 6 + \log_2 c.$$

Теорема 16 доказана. □

Алгоритм поиска, соответствующий графу U_m, представляет собой следующее.

Пусть нам дан некий запрос $x = (u, v) \in X_{int}$. Поиск по этому запросу будем осуществлять следующим образом.

Сначала вычислим длину интервала x.

Если $v - u < 1/([\log_2 k] + 1)$, то ответ будем искать по методу, описанному в теореме 15. Если $v - u \geq 1/([\log_2 k] + 1)$, то мы за время $\log_2 \log_2 k$ найдем номер j точки j/m, попадающей в интервал $[u, v]$. Затем, начиная с записи с номером s_j, просматриваем справа налево записи из V и сравниваем с левым концом запроса — точкой u. Как только очередная запись окажется меньше u, мы, начиная с записи с номером s_j+1, просматриваем слева направо записи из V и сравниваем с правым концом запроса — точкой v до тех пор, пока очередная запись не станет больше v.

Таким образом, в первом случае помимо перечисления ответа мы тратим время порядка $\log_2 k$, а во втором — $\log_2 \log_2 k$. Поскольку в подавляющем большинстве интервал x будет достаточно большим, то мы получим оценку теоремы 16.

2.5.3 Мгновенное решение

Пусть

$$G_3 = \{g_{\cdot,m}(u, v) = \max(1,]u \cdot m[) : m \in \mathbf{N}\}, \tag{2.34}$$

$$\mathcal{F}_3 = \langle F_1, G_1 \cup G_2 \cup G_3 \rangle, \tag{2.35}$$

где F_1, G_1, G_2 определяются соотношениями (2.28), (2.29), (2.31).

Понятно, что для $N_f \in \sigma$ для любого $f \in \widehat{G}_3$.

Справедлива следующая теорема.

Теорема 17. *Пусть ЗИП $I = \langle X_{int}, V, \rho_{int} \rangle$ — одномерная задача интервального поиска, то есть задача типа S_{int}, $\mathcal{F}_3$ — базовое множество, определяемое соотношением (2.35), и $R(I) \stackrel{\text{def}}{=} \sum_{y \in V} \mathbf{P}(O(y, \rho_{int}))$. Тогда, если функция плотности вероятности $p(x)$, определяющая вероятностную меру $\mathbf{P}$, ограничена константой c, то*

$$R(I) < T(I, \mathcal{F}_3, 4|V| - 3 + 6c[\log_2 |V|]) \leq R(I) + 5.$$

Доказательство. Нижняя оценка следует из теоремы 12.

Построим ИГ, на котором достигается верхняя оценка одномерной задачи интервального поиска.

Пусть m — некоторый параметр, значение которого определим позже.

Возьмем ИГ U_m, построенный в теореме 16. В нем из вершины β_2 исходит дерево D_{m-1}, концевыми вершинами которого являются вершины $\beta'_1, \ldots, \beta'_{m-1}$. Удалим все ребра дерева D_{m-1} и все вершины, не совпадающие с вершинами β_2 и $\beta'_1, \ldots, \beta'_{m-1}$. Выпустим из вершины β_2 $m-1$ ребро, припишем этим ребрам числа от 1 до $m-1$, и заменим нагрузку вершины β_2 на переключатель $g_{\cdot,m} \in G_3$. Полученный информационный граф обозначим через U'_m.

Отметим, что для любого запроса $x = (u, v) \in X_{int}$ если $j = g_{\cdot,m}(x)$, то j является минимальным числом, таким что $j/m \geq u$. Следовательно, вершина β_2 с исходящими из нее ребрами функционально совпадает с деревом D_{m-1}, и, значит, ИГ U'_m решает ЗИП I.

Что касается сложности графа U'_m, то по сравнению с ИГ U_m на любом запросе $x \in X_{int} \backslash A_{1/m}$ вместо $]\log_2(m-1)[$ вычислений переключателей, соответствующих проводящей цепи дерева D_{m-1}, будет вычислен один переключатель $g_{\cdot,m}$. Поэтому

$$
\begin{aligned}
T(U'_m) &= \mathbf{M}_x\, T(U'_m, x) = \mathbf{P}(A_{1/m}) \cdot (2 +]\log_2 k[) + \\
&\quad + \mathbf{P}(X \backslash A_{1/m}) \cdot 4 + \mathbf{M}_x\, |\mathcal{J}_{U'_m}(x)| \leq \\
&\leq \mathbf{P}(A_{1/m}) \cdot ([\log_2 k] - 1) + 4 + \sum_{y \in V} \mathbf{P}(O(y, \rho_{int})) \leq \\
&\leq c \cdot ([\log_2 k] - 1) \left(\frac{2}{m} - \frac{1}{m^2} \right) + 4 + \sum_{y \in V} \mathbf{P}(O(y, \rho_{int})) \leq \\
&\leq \frac{2c \cdot ([\log_2 k] - 1)}{m} + 4 + \sum_{y \in V} \mathbf{P}(O(y, \rho_{int})).
\end{aligned}
$$

Подсчитаем объем графа U'_m:

$$
Q(U'_m) \leq 2 + (2k - 1) + (k - 1) + (k - 1) + (m - 1) + 2(m - 1).
$$

Здесь первое слагаемое соответствует ребрам, исходящим из β_0. Второе слагаемое есть количество ребер в дереве D. Третье и четвертое слагаемые соответствуют ребрам из правосторонней и левосторонней концевых цепочек. Пятое слагаемое — это

ребра, исходящие из вершины β_2. И, наконец, шестое слагаемое не меньше чем число ребер, исходящих из вершин β'_i $(i = \overline{1,m})$.

Возьмем в качестве параметра $m = [2\,c\,[\log_2 k]]$ и получим

$$T(U'_m) \le 5 + \sum_{y \in V} \mathbf{P}(O(y, \rho_{int})),$$

$$Q(U'_m) \le 4\,k - 3 + 6\,c[\log_2 k],$$

что и доказывает утверждение теоремы 17. □

Дадим неформальное описание алгоритма, приведенного выше.

Пусть нам дано множество $V = \{y_1, \ldots, y_k\}$, в котором мы должны производить поиск. Сначала упорядочим его в порядке возрастания. Если известна оценка сверху c функции плотности вероятности появления запросов, то в качестве параметра m возьмем $m = 2\,c\,[\log_2 k]$, если же c неизвестна, то вместо нее можно взять любое число, например, $c = 2$. Затем для V построим множество номеров $S = \{s_1, \ldots, s_{m-1}\}$, описанное выше. Отметим, что оно строится только однажды. Теперь поиск по произвольно взятому интервалу-запросу $x = (u, v)$ производится следующим образом.

Сначала вычисляется длина запроса x.

Если она меньше чем $1/m$, то в множестве V бинарным поиском находится ближайшая справа к точке u запись. Далее, начиная с этой записи, просматриваются слева направо все записи из V и сравниваются с правым концом запроса — точкой v — до тех пор, пока очередная запись не станет больше v. Тем самым в этом случае, помимо перечисления ответа, производится порядка $\log_2 k$ действий.

Если $v - u \ge 1/m$, то с помощью функции $g_{\cdot,m}$ получаем номер j точки j/m, попадающей в интервал $[u, v]$. Затем, начиная с записи с номером s_j, просматриваем справа налево записи из V и сравниваем с левым концом запроса — точкой u. Как только очередная запись окажется меньше u, мы, начиная с записи с номером $s_j + 1$, просматриваем слева направо записи из V и сравниваем с правым концом запроса — точкой v до тех пор, пока очередная запись не станет больше v. Тем самым в этом случае мы, помимо перечисления ответа, производим 4 лишних

действия (сравниваем $v - u$ с $1/m$, вычисляем функцию $g_{\cdot,m}$, делаем 1 лишнее действие, идя справа налево, и 1 лишнее действие, идя слева направо).

Осталось заметить, что параметр m подобран так, что средняя сложность первого случая не превышает 1, если известна оценка сверху функции плотности вероятности, и не превышает некоторой константы, если эта оценка точно не известна.

И, наконец, заметим, что данный алгоритм требует дополнительную память порядка $\log_2 k$, чтобы хранить множество S.

Глава 3

Автоматизация решения задач. Логический подход

Формализация понятия задачи при разработке систем автоматического решения задач часто приводит к необходимости использования логического языка для представления обрабатываемой информации, а также связанной с этим языком формальной дедуктивной системы, определяющей допустимые процессы логического вывода. Возникающие здесь математические модели мы проиллюстрируем на двух типичных примерах — языке и исчислении логики высказываний, а также языке и исчислении логики предикатов.

По этой тематике может быть рекомендована книга [42].

3.1 Исчисление высказываний

3.1.1 Язык логики высказываний

При задании формального логического языка обычно следуют следующей схеме:

а) Указывается конечный либо бесконечный набор символов,

образующих алфавит языка; при описании алфавита могут вводиться те или иные характеристики его символов, в частности определяться разбиения их на подклассы.

б) Вводится индуктивное описание множества правильных выражений языка — конечных последовательностей символов алфавита.

в) Индуктивным образом определяется интерпретация языка (либо множеств допустимых интерпретаций), сопоставляющая каждому выражению языка обозначаемую им функцию, либо объект из некоторой "области интерпретации".

Пункты а) и б) задают синтаксис логического языка, пункт в) — его семантику

В случае языка логики высказываний эта общая схема реализуется следующим образом:

а) *Алфавит языка логики высказываний* состоит из счетного списка переменных $x_1, x_2, \ldots$; двух логических констант И, Л; заданного набора логических связок (например, связок $\vee, \&, \neg, \rightarrow, \leftrightarrow$), а также скобок (,).

б) *Правильные выражения языка логики высказываний*, называемые *формулами логики высказываний*, вводятся при помощи следующего определения:

1) Однобуквенное слово, состоящее из переменной, есть формула логики высказываний.

2) Если слова A и B суть формулы логики высказываний, то слова $(\neg A)$, $(A \vee B)$, $(A \& B)$, $(A \rightarrow B)$, $(A \leftrightarrow B)$ суть формулы логики высказываний.

в) При задании *интерпретации формул логики высказываний* возможны два различных подхода — эти формулы могут рассматриваться либо как обозначения функции алгебры логики, либо как обозначения единичных высказываний, имеющих определенное истинностное значение. В обоих случаях определяемый формулой объект (функция либо истинностное значения) вводится индуктивным образом, по шагам построения формулы, с использованием хорошо известных таблиц истинности для основных логических связок. В первом случае базис индукции сопоставляет каждой однобуквенной формуле x_i тождественную функцию; во втором случае — некоторое конкретное значение из множества И, Л.

Для большей определенности будем ниже рассматривать второй случай. Заметим, что при этом возникает множество различных интерпретаций языка логики высказываний, отличающиеся друг от друга лишь сопоставлением переменным алфавита $x_1, x_2, \ldots$ различных истинностных констант.

Таким образом, интерпретацией языка логики высказываний мы называем отображение I, определенное на множестве переменных $x_1, x_2, \ldots$ и сопоставляющее каждой переменной x_i некоторую истинностную константу a_i; $a_i \in \{\text{И}, \text{Л}\}$. Значение $(F)_I$ формулы F в интерпретации I определяем индукцией по построению формулы F:

1) Если F есть переменная x_i, то $(F)_I$ есть $I(x_i)$;

2) Если F имеет вид $\neg(G)$ и уже определено $(G)_I = b$, то $(F)_I$ есть истинностная константа, отличная от b;

3) Если F имеет вид $G_1 \vee G_2$ и уже определены $(G_1)_I = b_1$, $(G_2)_I = b_2$, то $(F)_I$ есть константа И, если хотя бы одно из b_1, b_2 равно И, иначе $(F)_I$ есть Л;

4) Если F имеет вид $G_1 \& G_2$ и уже определены $(G_1)_I = b_1$, $(G_2)_I = b_2$, то $(F)_I$ есть константа И, если b_1, b_2 равны И, иначе $(F)_I$ есть Л;

5) Если F имеет вид $G_1 \to G_2$ и уже определены $(G_1)_I = b_1$, $(G_2)_I = b_2$, то $(F)_I$ есть константа Л, если b_1 равно И, а b_2 равно Л; иначе $(F)_I$ есть И;

6) Если F имеет вид $G_1 \leftrightarrow G_2$ и уже определены $(G_1)_I = b_1$, $(G_2)_I = b_2$, то $(F)_I$ есть константа И, если $b_1 = b_2$, иначе $(F)_I$ есть Л.

Введем для логики высказываний несколько общих понятий, связанных с интерпретацией формул и являющихся типичными для логических языков.

Моделью формулы f будем называть произвольную интерпретацию языка логики высказываний в которой f имеет значение И.

Формулу имеющую хотя бы одну модель назовем *выполнимой.*

Формулу не имеющую ни одной модели назовем *невыполнимой.*

Формулу, для которой каждая интерпретация языка логики высказываний является моделью, назовем *общезначимой.*

В качестве примеров отметим, что формула $((x_1 \vee x_2) \to x_2)$ — выполнима, но не общезначима, формула $((x_1 \& x_2) \to x_2)$ — общезначима, а формула $(\neg(x_1 \to (x_2 \to x_1)))$ — невыполнима.

В дальнейшем часто мы будем опускать скобки, подразумевая, что $\neg$ — самая приоритетная операция, и в случае, когда порядок расставления скобок не существенен.

Описание класса всех общезначимых формул, является одной из наиболее важных задач, возникающих при изучении логических языков, и для него обычно используются следующие подходы:

1) Нахождение индуктивного описания класса всех общезначимых формул. Такое описание определяет в базисе индукции некоторое подмножество общезначимых формул (конечное либо бесконечное), называемых аксиомами, и указывает в индуктивном шаге перечень допустимых правил вывода, позволяющих получать новые общезначимые формулы исходя из ранее найденных. Логический язык вместе с индуктивным описанием такого рода образует *дедуктивную систему.*

2) Нахождение алгоритма, перечисляющего все общезначимые формулы — фактически, обеспечивается при построении дедуктивной системы.

3) Нахождение алгоритма, распознающего общезначимые формулы в классе всех правильных выражений языка. В отличие от пунктов 1 и 2, описание такого типа удается найти лишь для весьма немногих логических языков, к числу которых относится и язык логики высказываний.

3.1.2 Полнота исчисления высказываний

Существует много различных дедуктивных систем для описания общезначимых формул логики высказываний, отличающихся множеством используемых логических связок и выбором аксиом; приведем здесь одну из простейших таких систем в которой рассматриваются лишь две логические связки — $\neg$, $\to$. Остальные логические связки легко могут быть выражены через них и трактоваться как своего рода "сокращенные обозначения" для соответствующих формул с $\neg$ и $\to$. Аксиомы данной дедуктивной системы задаются при помощи так называемых

схем аксиом.

A1) Если f, g — формулы, то формула

$$(f \to (g \to f))$$

есть аксиома.

A2) Если f, g, h — формулы, то формула

$$((f \to (g \to h)) \to ((f \to g) \to (f \to h)))$$

есть аксиома.

A3) Если f, g — формулы, то формула

$$((\neg f \to \neg g) \to ((\neg f \to g) \to f))$$

есть аксиома.

В действительности каждая из схем аксиом A1, A2, A3 определяет бесконечное множество аксиом, отличающихся выбором формул f, g, h.

Правило вывода в этой дедуктивной системе одно — если уже получены общезначимые формулы f и $(f \to g)$, то из них выводится общезначимая формула g (согласно классификации предложенной еще Аристотелем, это правило называется modus ponens).

Чтобы проиллюстрировать технику, применяемую при изучении дедуктивных систем, приведем схематические наброски доказательств некоторых важных свойств только, что определенной дедуктивной системы.

Выводом формулы f из списка формул $\Gamma = g_1, \ldots, g_n$ будем называть произвольную последовательность формул $f_1, f_2, \ldots, f_m = f$, такую, что каждое $f_i (i = 1, \ldots, m)$ есть либо аксиома, либо элемент списка Γ, либо получено по правилу modus ponens из некоторых f_j, f_k, при $j, k \in \{1, \ldots, i-1\}$. Если существует вывод f из Γ, то обозначаем это обстоятельство посредством $\Gamma \vdash f$ (в частности, список Γ может быть пуст; те формулы, которые выводимы из пустого Γ, как легко можно видеть, образуют класс всех *выводимых формул* нашей дедуктивной системы). Следующая лемма есть пример выводимой формулы.

Лемма 7. *Если f — формула, то $\vdash (f \to f)$.*

Доказательство. Следующая последовательность формул, представляющая собой вывод из пустого Γ, доказывает утверждение.

$$(f \to ((f \to f) \to f)) \to ((f \to (f \to f)) \to (f \to f));$$

$$(f \to ((f \to f) \to f)); (f \to (f \to f)) \to (f \to f);$$

$$f \to (f \to f); (f \to f).$$

Первый ее элемент — аксиома, возникающая по схеме аксиом A2; второй и четвертый — аксиомы по A1; третий и пятый — получены применением правила вывода. □

Имеет место следующее важное утверждение, известное как теорема дедукции (Эрбран).

Теорема 18. *Если* Γ *— список формул и* f, g *— формулы, то из* $\Gamma, f \vdash g$ *вытекает* $\Gamma \vdash (f \to g)$.

Доказательство. Данное утверждение доказывается индукцией по длине n вывода $g_1, \ldots, g_n = g$ формулы g из Γ, f.

Базис индукции. Если $n = 1$, то g — либо аксиома, либо элемент списка Γ, f. В первом случае вывод $(f \to g)$ из Γ имеет вид $(g \to (f \to g)), g, (f \to g)$. Во втором случае возможны два подслучая — если $g \in \Gamma$, то снова берем указанный выше вывод; если же $g = f$, то используем приведенный выше вывод $(f \to f)$ из пустого списка.

Индуктивный переход. Если $n > 1$ и g — аксиома, либо элемент списка Γ, f, то поступаем так же как выше. Наконец если g получено по правилу modes ponens из формул g_i, g_j, $(i, j < n)$, то согласно предположению индукции имеем $\Gamma \vdash (f \to g_i)$, $\Gamma \vdash (f \to g_j)$. Заметим, что одна из формул g_i, g_j, например g_j, должна иметь вид $(g_i \to g)$. Для получения вывода $(f \to g)$ из Γ теперь достаточно рассмотреть последовательность образованную расположенными подряд выводами $(f \to g_i)$, $(f \to g_j)$ из Γ, и присоединить к ней в конце формулы $((f \to (g_i \to g)) \to (((f \to g_i) \to (f \to g)))$, $((f \to g_i) \to (f \to g))$, $(f \to g)$. Первая из этих формул есть аксиома полученная по схеме аксиом A2; две последние получены применением правила вывода. □

Для доказательства того, что каждая общезначимая формула алгебры логики выводима в рассматриваемой нами дедуктивной системе, нам понадобятся следующие вспомогательные утверждения.

Лемма 8 (доказательство от противного). *Если* Γ *— список формул и* f, g *— формулы, то из* $\Gamma, \neg f \vdash g, \neg g$ *вытекает* $\Gamma \vdash f$.

Доказательство. Так как $\Gamma, \neg f \vdash g, \neg g$, то по теореме дедукции $\Gamma \vdash (\neg f \to g), (\neg f \to \neg g)$.

Вывод f из Γ следующий:

$$((\neg f \to \neg g) \to ((\neg f \to g) \to f), ((\neg \to g) \to f), f.$$

Здесь сначала используется аксиома А3, а затем — дважды правило modus ponens. □

Лемма 9 (снятие двойного отрицания). *Если* f *— формула, то* $\vdash (\neg\neg f \to f)$.

Доказательство. Следующее утверждение очевидно:

$$\neg\neg f, \neg f \vdash \neg f, \neg\neg f.$$

Используя лемму 8, получаем $\neg\neg f \vdash f$. Отсюда по теореме дедукции имеем $\vdash (\neg\neg f \to f)$. □

Лемма 10. *Если* f *— формула, то* $\vdash (f \to \neg\neg f)$.

Доказательство. По лемме 9 имеем $f, \neg\neg\neg f \vdash f, \neg f$. Используя лемму 8, получаем $f \vdash \neg\neg f$. Отсюда по теореме дедукции имеем $\vdash (f \to \neg\neg f)$. □

Доказательства следующих трех утверждений рекомендуются в качестве самостоятельных упражнений.

Лемма 11. *Если* f *— формула, то* $\vdash (\neg f \to (f \to g))$.

Лемма 12. *Если* f *— формула, то* $\vdash (f \to (\neg g \to \neg(f \to g)))$.

Лемма 13. *Если* f *— формула, то*

$$\vdash ((f \to g) \to ((\neg f \to g) \to g)).$$

Для произвольной формулы g и некоторой интерпретации I языка логики высказываний обозначим посредством $(g)_I$ формулу, совпадающую с g, если значение g в интерпретации I есть И, и совпадающую с $(\neg g)$ в противном случае.

Лемма 14. *Пусть f — формула логики высказываний; $x_1, \ldots, x_n$ — все переменные входящие в f, и I — некоторая интерпретация языка логики высказываний. Тогда выполняется $(x_1)_I, \ldots, (x_n)_I \vdash (f)_I$.*

Доказательство. Это утверждение будем доказывать индукцией по числу m логических связок в f.

Базис индукции. Если $m = 0$, то f совпадает с x_1, и утверждение сводится к очевидному факту $(x_1)_I \vdash (x_1)_I$.

Индуктивный переход. Пусть $m > 0$. Рассмотрим следующие два случая.

1) f имеет вид $(\neg h)$. Этот случай разбивается на следующие подслучаи.

   а) Если $(h)_I = h$, то $(f)_I = \neg\neg h$. По предположению индукции имеем $(x_1)_I, \ldots, (x_n)_I \vdash h$. По лемме 10 имеем $h \to \neg\neg h$. Применяя modus ponens, получаем $(\neg\neg h)$.

   б) Если же $(h)_I = (\neg h)$, то $(f)_I = f = (\neg h)$, и утверждение сводится к предположению индукции.

2) f имеет вид $(h_1 \to h_2)$. Этот случай разбивается на следующие подслучаи.

   а) Если $(h_1)_I = (\neg h_1)$, то $(f)_I = f = (h_1 \to h_2)$. По предположению индукции имеем $(x_1)_I, \ldots, (x_n)_I \vdash (\neg h)$. По лемме 11 имеем $(\neg h_1 \to (h_1 \to h_2))$. Применяя modus ponens, получаем $(h_1 \to h_2)$.

   б) Если $(h_2)_I = \neg h_2$, то $(f)_I = f = (h_1 \to h_2)$.По аксиоме A1 имеем $h_2 \to (h_1 \to h_2)$. По предположением индукции $(x_1)_I, \ldots, (x_n)_I \vdash h_2$. Применяя modus, ponens получаем $(h_1 \to h_2)$.

   в) В случае $(h_1)_I = h_1$ и $(h_2)_I = \neg h_2$ имеем $(f)_I = \neg f = \neg(h_1 \to h_2)$. По предположением индукции

$$(x_1)_I, \ldots, (x_n)_I \vdash h_1, (\neg h_2).$$

По лемме 12 имеем $(h_1 \to (\neg h_2 \to \neg(h_1 \to h_2)))$. Дважды применяя modus ponens, получаем $\neg(h_1 \to h_2)$.

□

Теорема 19 (о полноте исчисления высказываний). *Формула логики высказываний является общезначима тогда и только тогда, когда она выводима в рассматриваемой дедуктивной системе со схемами аксиом A1, A2, A3 и правилом вывода modus ponens.*

Доказательство. *Достаточность.* Если f и $(f \to g)$ — общезначимые формулы, то понятно, что и формула g — общезначимая. Следовательно, правило modus ponens из общезначимых формул выводит общезначимые. Поскольку, как легко видеть, схемы аксиом задают общезначимые формулы, то все выводимые формулы — общезначимы.

Необходимость. Будем обозначать посредством f^σ, где $\sigma \in \{\text{И}, \text{Л}\}$, формулу f в случае $\sigma = \text{И}$, и формулу $(\neg f)$ в случае $\sigma = \text{Л}$. Пусть f произвольная общезначимая формула алгебры логики, и $x_1, \ldots, x_n$ — все входящие в нее переменные. Покажем индукцией по k, что для любого целого неотрицательного k, $k \leq n$, и любого набора $(\sigma_1, \ldots, \sigma_k)$ логических констант И, Л выполняется $x_1^{\sigma_1}, \ldots, x_k^{\sigma_k} \vdash f$.

Базис индукции. $k = n$, т.е. задана интерпретация $I = (\sigma_1, \ldots, \sigma_n)$. Так как $(x_i)_I = x_i^{\sigma_i}$, то по лемме 14 имеем $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n} \vdash (f)_I = f$.

Индуктивный переход. Если данное утверждение уже доказано для некоторого k, $k > 0$, то рассмотрим произвольный набор $(\sigma_1, \ldots, \sigma_{k-1})$ констант И, Л. По предположению индукции, имеем $x_1^{\sigma_1}, \ldots, x_{k-1}^{\sigma_{k-1}}, x_k \vdash f$ и $x_1^{\sigma_1}, \ldots, x_{k-1}^{\sigma_{k-1}}, \neg x_k \vdash f$. По теореме дедукции находим отсюда $x_1^{\sigma_1}, \ldots, x_{k-1}^{\sigma_{k-1}} \vdash (x_k \to f)$ и $x_1^{\sigma_1}, \ldots, x_{k-1}^{\sigma_{k-1}} \vdash (\neg x_k \to f)$. По лемме 13 имеем $(x_k \to f) \to ((\neg x_k \to f) \to f)$. Дважды воспользовавшись правилом modus ponens, получаем $x_1^{\sigma_1}, \ldots, x_{k-1}^{\sigma_{k-1}} \vdash f$. Шаг индукции, таким образом, доказан.

При $k = 0$ окончательно получим $\vdash f$. □

3.1.3 Алгоритмы распознавания общезначимости формул логики высказываний

В случае логики высказываний проверка общезначимости формулы, имеющей n переменных, может быть осуществлена путем перебора всех 2^n возможных наборов логических констант, сопоставляемых этим переменным при различных интерпретациях, и установлении того, что все значения формулы на этих наборах равны И. Таким образом здесь имеется не только алгоритм перечисления всех общезначимых формул, но и алгоритм проверки общезначимости. Однако даже для такого простейшего случая остается проблема уменьшения трудоемкости распознавания общезначимости по сравнению с практически малоприемлимым для сколь-нибудь значительных n перебором всех 2^n наборов значений переменных. К проблеме распознавания общезначимости формул логики высказываний тесно примыкает проблема решения систем логических уравнений сводящаяся к нахождению всех наборов значений переменных при которых заданная формула алгебры логики принимает значение И. Последняя проблема часто встречается в различных прикладных задачах дискретной оптимизации и диагностики, и поиску эффективных процедур решения ее, учитывающих в своих эвристических решающих правилах статистические особенности рассматриваемого класса задач, посвящено большое количество исследований. Мы ограничимся здесь лишь несколькими простейшими процедурами подобного рода (применительно к задаче распознавания общезначимости), поскольку они позволят нам развить технику доказательства теорем в логике высказываний, представляющую собой упрощенный аналог техники, развиваемой далее для автоматического доказательства теорем в логике предикатов.

Прежде всего, опишем *алгоритм Квайна*, осуществляющий проверку общезначимости формулы f логики высказываний при помощи построения так называемого семантического дерева. Корню этого дерева — исходной вершине — приписывается формула f. Пусть уже построено некоторое подмножество вершин семантического дерева, каждой из которых сопоставлена некоторая формула. Если каждой концевой вершине данного

дерева оказалась сопоставлена константа И, то процесс завершается, и формула f является общезначимой. Если некоторой концевой вершине дерева сопоставлена константа Л, то формула f не общезначима, и процесс также завершается. В противном случае найдется концевая вершина v, которой сопоставлена формула отличная от констант И, Л. Если эта формула g не содержит переменных, то ее можно преобразовать в логическую константу, используя тождества:

$$\neg И = Л;\ \neg Л = И;\ И \vee f = И;\ Л \vee f = f;$$
$$И \& f = f;\ Л \& f = Л;\ И \to f = f; Л \to f = И;$$
$$f \to И = И;\ f \to Л = \neg f;\ И \leftrightarrow f = f;\ Л \leftrightarrow f = \neg f.$$

Если же она содержит хотя бы одну переменную x, и выполняются следующие действия:

а) Находятся результаты g_1 и g_2 подстановки в формулу g вместо переменной x, соответственно констант И и Л.

б) Находятся результаты g_1' и g_2' упрощения формул g_1 и g_2 при помощи перечисленных выше тождеств, применяемых до тех пор пока это возможно.

в) вводятся две новые вершины v_1 и v_2 семантического дерева, которым приписываются, соответственно, формулы g_1' и g_2'. К этим вершинам от вершины v проводятся ребра, отмеченные соответственно выражениями $x = И$ и $x = Л$.

Далее повторяется описанный выше процесс рассмотрения концевых вершин семантического дерева.

В данной процедуре остался недоопределенным выбор конкретной переменной x формулы g, по которой проводится "разбор случаев". Этот выбор в прикладных программах, решающих системы логических уравнений путем построения семантических деревьев, осуществляется на основании различных эвристических решающих правил. Простейшим таким правилом является выбор переменной, имеющей наибольшее число вхождений в g, для получения наиболее сильного упрощения при переходе к g_1' и g_2', другим полезным соображением является такой выбор переменных x, при котором сопоставленные концевым вершинам семантического дерева формулы g оказывались бы представимы, например, как $h_1 \& h_2$ либо $h_1 \vee h_2$, где формулы h_1 и h_2 не имеют общих переменных. В этом случае вместо

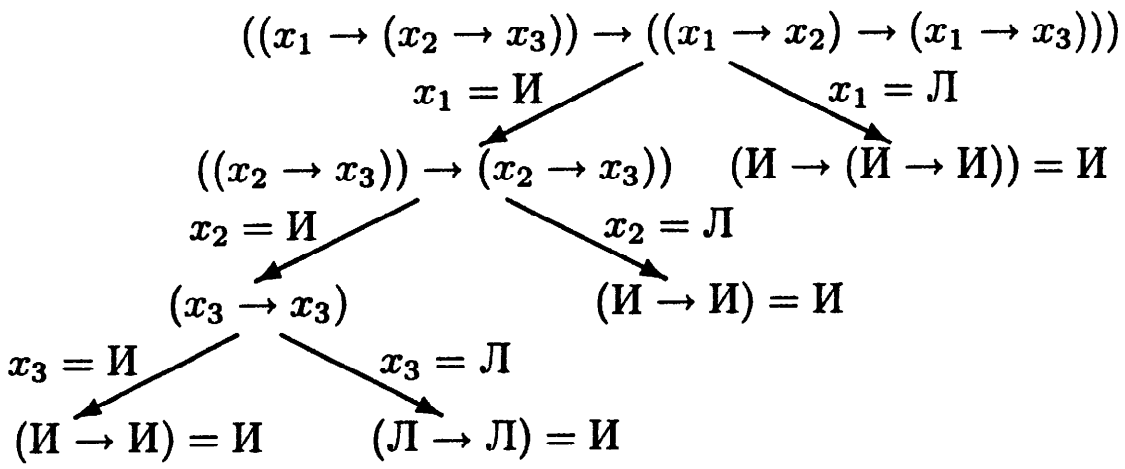

Рис. 3.1: Алгоритм Квайна.

построения ветви дерева для $h_1 \& h_2$ ($h_1 \vee h_2$, $h_1 \to h_2$ и т.п.) можно было бы рассмотреть независимо формируемые деревья для h_1, h_2 и из анализа их концевых вершин сделать вывод относительно общезначимости g.

Пример. В качестве примера применения алгоритма Квайна убедимся в общезначимости формулы, полученной из схемы аксиом А2:

$$((x_1 \to (x_2 \to x_3)) \to ((x_1 \to x_2) \to (x_1 \to x_3))) \qquad (3.1)$$

Дерево разбора случаев для этой формулы приведено на рисунке 3.1.

В прикладных ситуациях логические языки редко используются во всем многообразии своих синтаксических возможностей. Обычно их формулы приводятся эквивалентными преобразованиями к тому или иному "стандартному виду", который и сохраняется в процессе обработки информации. С точки зрения логики высказываний, наиболее типичной логической стандартной формой, отражающей тенденцию к описанию ситуаций в виде конъюнкции условий, является конъюнктивная нормальная форма. Формула, представленная, в этой форме, имеет вид $A_1 \& \ldots \& A_n$, $n \geq 1$, где каждое A_i ($i = 1, \ldots, n$) — формула вида $(B_{i1} \vee \cdots \vee B_{im_i})$, где $m_i \geq 1$, причем $(B_{i1}, \ldots, B_{im_i})$ — переменные или отрицания переменных.

Более точно, для определения конъюнктивной нормальной формы введем сначала понятие дизъюнкта. Если $x_1, \ldots, x_m$ — различные переменные, $m \geq 1$, то формулу вида $(x_1^{\sigma_1} \vee \cdots \vee x_m^{\sigma_m})$

назовем *дизъюнктом*. Логическую константу Л по определению так же считаем *дизъюнктом*.

Если $A_1, \ldots, A_n$ — различные дизъюнкты ($n \geq 1$), то формула $(A_1 \& \ldots \& A_n)$ называется *конъюнктивной нормальной формой*.

Произвольную формулу логики высказываний можно преобразовать к виду конъюнктивной нормальной формы, используя следующие эквивалентные преобразования:

а) логические связки $\rightarrow$, $\leftrightarrow$ устраняются при помощи тождеств

$$(a \rightarrow b) = (\neg a \vee b);\ (a \leftrightarrow b) = (a \& b \vee \neg a \& \neg b);$$

б) отрицания в формуле "опускаются до переменных" при помощи тождеств

$$\neg\neg a = a;\ \neg(a \vee b) = \neg a \& \neg b;\ \neg(a \& b) = \neg a \vee \neg b;$$

в) применяются преобразования дистрибутивности:

$$(a \vee (b \& c)) = (a \vee b) \& (a \vee c);$$

г) устраняются повторные вхождения переменных в дизъюнктивных подформулах, а также константы И, Л (если сама формула не есть И или Л):

$$a \vee a = a;\ a \vee \neg a = \text{И};\ a \vee \text{И} = \text{И};\ a \& \text{И} = a;\ a \& \text{Л} = \text{Л};$$

д) устраняются одинаковые дизъюнкты: $a \& a = a$.

В процедурах автоматического доказательства теорем часто применяется метод доказательства "от противного". В этом случае для доказательства общезначимости формулы f переходят к рассмотрению формулы $\neg f$ и пытаются доказать ее невыполнимость.

Опишем модифицированный алгоритм Квайна, предназначенный для установления невыполнимости формулы логики высказываний g, преобразованной к виду конъюнктивной нормальной формы. Вершинам семантического дерева в этом случае будут сопоставляться не формулы, а их множества дизъюнктов.

Первоначально вводим корень семантического дерева и сопоставляем ему множество дизъюнктов формулы g.

Пусть уже построена часть семантического дерева.

Если концевой вершине дерева сопоставлено пустое множество дизъюнктов (пустое множество соответствует логической константе И), то формула g — выполнима, и алгоритм прекращает работу.

Если каждой концевой вершине семантического дерева соответствует множество дизъюнктов, содержащее дизъюнкт Л, то формула g — невыполнима, и алгоритм прекращает работу.

Если некоторой концевой вершине v сопоставлено непустое множество дизъюнктов S, не содержащее дизъюнкта Л, то в S входят дизъюнкты с переменными. Выбираем одну из таких переменных x и разбиваем S на три подкласса: S_1 — все дизъюнкты, в которые x входит без внешнего отрицания, S_2 — все дизъюнкты, в которые x входит с отрицанием, S_3 — все дизъюнкты в которых x не встречается. Далее рассматриваем множество дизъюнктов $S_x = \{d | d \vee \neg x \in S_2\}$ (если $\neg x \in S_2$, то здесь берется $d =$ Л) и множество дизъюнктов и множество дизъюнктов $S_{\neg x} = \{d | d \vee x \in S_1\}$ (если $x \in S_1$, то здесь берется $d =$ Л). Нетрудно видеть, что невыполнимость конъюнкции дизъюнктов списка S эквивалента одновременной невыполнимости конъюнкций дизъюнктов списков $S_x \cup S_3$ и $S_{\neg x} \cup S_3$. Поэтому для продолжения построения семантического дерева вводим две новых вершины v_1 и v_2, которым сопоставляем, соответственно, множества дизъюнктов $S_x \cup S_3$ и $S_{\neg x} \cup S_3$, причем проводим к v_1 и v_2 ребра от v, отмеченные соответственно выражениями $x =$ И и $x =$ Л.

Далее повторяется описанный выше процесс рассмотрения концевых вершин семантического дерева.

Пример. В качестве примера применения модифицированного алгоритма Квайна обратимся опять к формуле f, заданной выражением (3.1). После устранения логических связок $\rightarrow$ мы получим формулу

$$g = (\neg(\neg x_1 \vee \neg x_2 \vee x_3)) \vee (\neg(\neg x_1 \vee x_2)) \vee \neg x_1 \vee x_3.$$

Далее переходим к рассмотрению формулы $\neg g$, которая имеет вид:

$$\neg g = (\neg x_1 \vee \neg x_2 \vee x_3)\&(\neg x_1 \vee x_2)\&x_1\&(\neg x_3). \qquad (3.2)$$

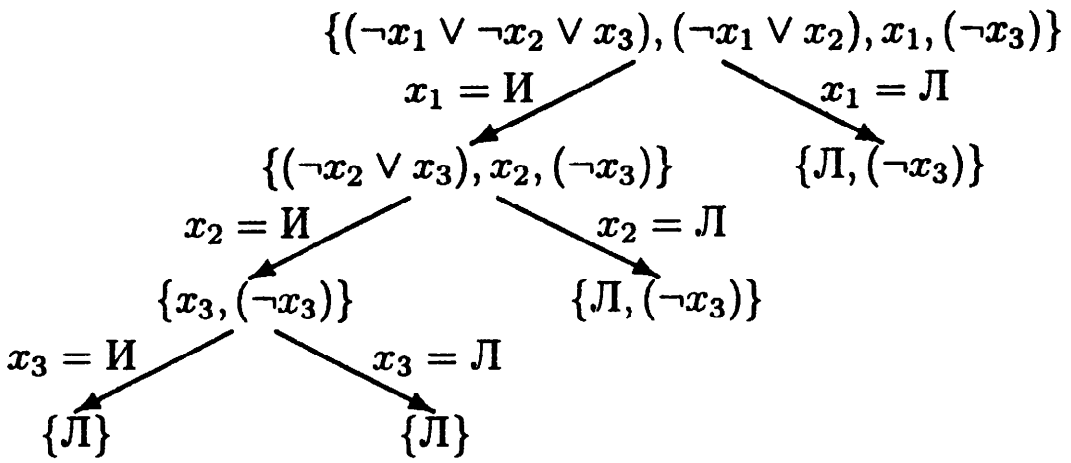

Рис. 3.2: Модифицированный алгоритм Квайна.

Семантическое дерево для формулы $\neg g$ приведено на рисунке 3.2.

Другой алгоритм распознавания невыполнимости конъюнктивной нормальной формы связан с рассмотрением специальной дедуктивной системы, использующей в качестве правила вывода так называемое *правило резолюции*. В этом случае конъюнктивная нормальная форма снова представляется как множество своих дизъюнктов, эти дизъюнкты и образуют перечень аксиом данной дедуктивной системы. *Правило резолюции*, будучи применено к двум дизъюнктам $A \vee B$ и $\neg A \vee C$ создает в качестве следствия новый дизъюнкт $B \vee C$ (здесь возможны вырожденные случаи отсутствия B либо C, если они оба отсутствуют, то результатом служит дизъюнкт Л). Достаточность применения данного правила вывода для распознавания невыполнимости гарантируется следующим утверждением.

Теорема 20. *Конъюнкция конечного семейства дизъюнктов S невыполнима тогда и только тогда, когда за конечное применений правила резолюции из S выводится дизъюнкт Л.*

Доказательство. Доказывать будем индукцией по числу n переменных, встречающихся в дизъюнктах из S.

Базис индукции. Если $n = 0$, то утверждение очевидно, так как в этом случае непустое семейство дизъюнктов может содержать только логическую константу Л.

Индуктивный переход. Пусть утверждение уже доказано

для некоторого n.

Рассмотрим семейство S с $n+1$ переменной: $x_1, x_2, \ldots, x_{n+1}$. Разобьем S на три подкласса: S_1 — все дизъюнкты представимые в виде $x_{n+1} \vee a$, S_2 — все дизъюнкты представимые в виде $\neg x_{n+1} \vee b$, S_3 — дизъюнкты не содержащие x_{n+1}.

Дизъюнкты множества

$$R = \{a \vee b : x_{n+1} \vee a \in S_1, \neg x_{n+1} \vee b \in S_2\}$$

получены из дизъюнктов семейства S применением правила резолюции (здесь допускаются вырожденные случаи, когда a либо b отсутствует, причем при одновременном отсутствии a и b в R включается константа Л).

Рассмотрим семейство дизъюнктов $R \cup S_3$ и покажем, что его невыполнимость эквивалентна невыполнимости S. Если S выполнимо, то интерпретация, обращающая в И дизъюнкты из S дает значения И и для всех дизъюнктов из $R \cup S_3$, т.е. $R \cup S_3$ выполнимо. Пусть S невыполнимо, рассмотрим набор истинностных значений $\alpha_1, \ldots, \alpha_n$ переменных $x_1, \ldots, x_n$ и покажем, что на этом наборе хотя бы один из дизъюнктов семейства $R \cup S_3$ имеет значение Л. Так как S невыполнимо, то на наборе $\alpha_1, \ldots, \alpha_n$, И значений переменных $x_1, \ldots, x_n, x_{n+1}$ некоторый дизъюнкт d из S имеет значение Л. Если $d \in S_3$, то он и является искомым дизъюнктом в $R \cup S_3$. Так как x_{n+1} имеет значение И, то при $d \notin S_3$ остается единственная возможность $d \in S_2$, т.е. $d = \neg x_{n+1} \vee b$. Аналогично на наборе $\alpha_1, \ldots, \alpha_n$, Л значений переменных $x_1, \ldots, x_n, x_{n+1}$ некоторый дизъюнкт d' из S имеет значение Л. Снова, если $d' \in S_3$, то он и есть искомый, а в противном случае (так как x_{n+1} имеет значение Л) необходимо $d' \in S_1$, то есть $d' = x_{n+1} \vee a$. Дизъюнкты a и b не содержат переменной x_{n+1}, и поэтому оба принимают значение Л, если переменные $x_1, \ldots, x_n$ имеют значения $\alpha_1, \ldots, \alpha_n$. Следовательно и дизъюнкт $d'' = a \vee b$, принадлежащий R, будет иметь на наборе $\alpha_1, \ldots, \alpha_n$ значений переменных $x_1, \ldots, x_n$ значение Л. Это и означает доказательство эквивалентности невыполнимости S и $R \cup S_3$.

Но число переменных в $R \cup S_3$ не более n, т.е. для него имеет место эквивалентность невыполнимости и существования вывода константы Л.

Если константа Л выводима из S, то либо дизъюнкт Л принадлежит S, и тогда S — невыполнимо, либо константа Л выводима из $R \cup S_3$, тогда $R \cup S_3$ — невыполнимо и, следовательно, S — невыполнимо.

Если S невыполнимо, то невыполнимо $R \cup S_3$ и из $R \cup S_3$, а, значит, и из S выводима константа Л.

Тем самым теорема доказана. □

Пример. В качестве примера применения правила резолюции обратимся опять к формуле $\neg g$, заданной выражением (3.2). Исходное множество дизъюнктов имеет вид

$$\{(\neg x_1 \vee \neg x_2 \vee x_3), (\neg x_1 \vee x_2), x_1, (\neg x_3)\}.$$

Из $(\neg x_1 \vee x_2)$ и x_1 по правилу резолюции выводится x_2. Из $(\neg x_1 \vee \neg x_2 \vee x_3)$ и x_1 выводим $\neg x_2 \vee x_3$. Из $(\neg x_2 \vee x_3)$ и x_2 выводим x_3. И наконец, из x_3 и $\neg x_3$ получаем Л. Следовательно, формула $\neg g$ невыполнима.

Упражнения

3.1. Какие из следующих выражений являются правильными выражениями (формулами) логики высказываний:

а) x_1;

б) (x_2);

в) $x_1 \vee x_2$;

г) $(x_1 \vee x_2)$;

д) $x_1 x_2$;

е) $(x_1 \vee x_3)\&(x_1 \vee x_2)$;

ж) $((x_1 \vee \neg x_2 \vee x_3)\&(x_1 \vee x_2))$;

з) $((x_1 \vee (\neg x_2) \vee x_3)\&(x_1 \vee x_2))$;

и) $((x_1 \vee x_2)\&(x_1 \vee x_2))$?

3.2. Расставьте скобки в следующих выражениях (с учетом приоритета операций $\neg$, $\&$, $\vee$, $\rightarrow$, $\leftrightarrow$ и левой ассоциативности всех двухместных операций), так чтобы получились правильные выражения логики высказываний:

а) $x_1 \vee \neg x_3 \to x_1 \& x_2$;

б) $x_1 \vee x_2 \vee x_3 \& x_4 \leftrightarrow x_1$;

в) $\neg\neg x_1 \vee \neg x_2 \to x_1$;

г) $x_2 \leftrightarrow \neg x_1 \to x_3 \& x_1$.

3.3. Какие из следующих формул логики высказываний общезначимы, а какие — выполнимы:

а) $(x_1 \vee x_2)$;

б) $(x_1 \vee (\neg x_1))$;

в) $(x_1 \& (\neg x_1))$;

г) $(((\neg x_1) \vee (\neg x_2)) \& ((x_2 \vee (\neg x_1)) \& x_1))$;

д) $((x_1 \vee ((\neg x_2) \vee x_3)) \& ((x_1 \vee x_2) \& (\neg x_3)))$?

3.4. Построить таблицу истинности для следующих формул логики высказываний:

а) $(x_1 \leftrightarrow ((x_1 \& x_2) \vee (\neg x_3)))$;

б) $((x_1 \to x_2) \& (\neg((\neg x_2 (\to (\neg x_1))))$;

в) x_2;

г) $(((x_1 \vee x_2) \& (x_3 \vee (\neg x_2))) \to (x_1 \vee x_3))$.

3.5. Для каждой выполнимой формулы из упражнений 3.3 и 3.4 найти хотя бы одну ее модель.

3.6. Для каждой формулы из упражнений 3.3 и 3.4 проверить является ли она выполнимой, невыполнимой, общезначимой с помощью метода Квайна, модифицированного метода Квайна и метода резолюции.

3.7. Докажите эквивалентность следующих формул логики высказываний:

а) $(x_1 \vee (x_1 \& x_2))$ и $(x_1 \vee x_2)$;

б) $(x_1 \to x_2)$ и $((\neg x_2) \to (\neg x_1))$;

в) $((\neg x_1) \to x_1)$ и $((x_2 \leftrightarrow (\neg x_2)) \leftrightarrow x_1)$.

3.8. Докажите леммы 11, 12 и 13.

3.9. Пусть f, g, h — формулы логики высказываний. Докажите справедливость следующих утверждений:

а) $(f \to g), (g \to h) \vdash (f \to h)$ (правило силлогизма);

б) $(f \to (g \to h)), g \vdash (f \to h)$;

в) $\vdash ((\neg g \to \neg f) \to (f \to g))$;

г) $\vdash ((f \to g) \to (\neg g \to \neg f))$.

3.2 Исчисление предикатов

3.2.1 Язык логики предикатов

В случае языка логики предикатов наш алфавит будет состоять из следующих групп символов:

1) счетный список предметных переменных $x_1, x_2, \ldots$;
2) счетный список предметных констант $a_1, a_2, \ldots$;
3) счетный список функциональных символов $f_1, f_2, \ldots$;
4) счетный список предикатных символов $P_1, P_2, \ldots$;
5) логические константы И, Л;
6) логические связки $\neg, \vee, \&, \rightarrow, \leftrightarrow$;
7) кванторы $\forall$ и $\exists$;
8) скобки "(", ")" и запятая ",".

Каждый функциональный либо предикатный символ характеризуется своей *арностью* — некоторым натуральным числом. При этом предполагается, что для каждого натурального n имеется бесконечно много как предикатных, так и функциональных символов арности n.

Правильные выражения языка логики предикатов разбиваются на два непересекающихся множества — множество термов и множество формул. Дадим сначала индуктивное определение *множества термов*.

Т1) Однобуквенное слово, состоящее из предметной переменной либо предметной константы, есть терм.

Т2) Если $t_1, \ldots, t_n$ — термы и g — функциональный символ арности n, то слово $g(t_1, \ldots, t_n)$ — есть терм.

Индуктивное определение *формул логики предикатов* опирается на уже введенное понятие терма.

Ф1) Однобуквенное слово состоящее из логической константы И либо Л, есть формула логики предикатов.

Ф2) Если $t_1, \ldots, t_n$ — термы и g — предикатный символ арности n, то слово $g(t_1, \ldots, t_n)$ есть формула логики предикатов.

Ф3) Если f и g — формулы логики предикатов, то слова $(\neg f)$, $(f \vee g)$, $(f \& g)$, $(f \rightarrow g)$, $(f \leftrightarrow g)$ суть формулы логики предикатов (такие формулы будем называть *атомарными* или *атомами*).

Ф4) Если f — формула логики предикатов и x — предметная переменная, то слова $(\forall x f)$ и $(\exists x f)$ суть формулы логики предикатов.

Интерпретацией языка логики предикатов называем четверку (M, F_1, F_2, F_3), такую что:

а) M — непустое множество, называемое областью интерпретации;

б) F_1 — функция, сопоставляющая каждой предметной константе некоторый элемент из M;

в) F_2 — функция, сопоставляющая каждому функциональному символу f арности n некоторую n-местную функцию, определенную на M и принимающую значения из M.

г) F_3 — функция, сопоставляющая каждому предикатному символу P арности n некоторой n-местный предикат определенный на M (функцию со значениями И, Л).

Если задана интерпретация $I = (M, F_1, F_2, F_3)$, то каждый терм t языка логики предикатов определяет в этой интерпретации некоторую функцию $(t)_I$, определенную на M и принимающую значения из M, а каждая формула f — предикат $(f)_I$, определенный на М (в вырожденных случаях $(t)_I$ может отождествляться с элементом M, а $(f)_I$ с логической константой).

Здесь используется следующее индуктивное определение:

1) Если x — предметная переменная, то $(x)_I$ есть тождественная функция от переменной x определенная на M.

2) Если a — предметная константа, то $(a)_I$ есть элемент $F_1(a)$ из M.

3) Если для термов $t_1, \ldots, t_n$ уже определены $(t_1)_I, \ldots, (t_n)_I$, f — функциональный символ арности n и $\phi = F_2(f)$ — функция от n переменных, отображающая M^n в M, то

$$(f(t_1, \ldots, t_n))_I = \phi((t_1)_I, \ldots, (t_n)_I).$$

4) $(\text{И})_I = \text{И}$, $(\text{Л})_I = \text{Л}$.

5) Если для термов $t_1, \ldots, t_n$ уже определены $(t_1)_I, \ldots, (t_n)_I$, P — предикатный символ арности n и $\pi = F_3(P)$ — предикат от n переменных, отображающий M^n в $\{\text{И}, \text{Л}\}$, то

$$(P(t_1, \ldots, t_n))_I = \pi((t_1)_I, \ldots, (t_n)_I).$$

6) Если для f и g уже определены $(f)_I$ и $(g)_I$, то $(\neg f)_I$, $(f \vee g)_I$, $(f\&g)_I$, $(f \to g)_I$, $(f \leftrightarrow g)_I$ получаются применением истинностных функций для $\neg, \vee, \&, \to, \leftrightarrow$ к $(f)_I$ и $(g)_I$.

7) Если уже определен предикат $(f)_I = \phi(x_1, \ldots, x_n)$, то $(\forall x_i f)_I = \psi(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ — предикат, принимающий значение И на таких наборах $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n$ элементов M, что для каждого α_i из M значение $\phi(\alpha_1, \ldots, \alpha_n)$ есть И. Аналогично, $(\exists x_i f)_I = \xi(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ — предикат, принимающий значение И на таких наборах $\alpha_1, \ldots, \alpha_{i-1}$, $\alpha_{i+1}, \ldots, \alpha_n$ элементов M, что существует α_i из M, что значение $\phi(\alpha_1, \ldots, \alpha_n)$ есть И.

Вхождение переменной x в формулу f логики предикатов, расположенное внутри подформул вида $(\forall x g)$ либо $(\exists x g)$, называется *связанным*, вхождения переменных не являющиеся связанными, называются *свободными*. Переменная имеющая хотя бы одно свободное вхождение в формулу либо терм f, называется *свободной переменной* f. Как нетрудно видеть, функция либо предикат $(f)_I$, определенная выше, зависит лишь от свободных переменных f. Формула без свободных переменных называется *замкнутой*.

В формуле $\exists x_1((\forall x_2 P_1(x_2, x_1)) \vee P_2(x_2, x_1))$ первое вхождение переменной x_2 связанное, а второе — свободное, оба вхождения переменной x_1 связанные. Тем самым, у этой формулы одна свободная переменная — x_2. Формула $((\exists x_1 P_1(x_1)) \to (\forall x_2 P_2(x_2)))$ — пример замкнутой формулы.

Терм t называется *свободным для переменной* x_i в формуле логики предикатов f, если никакое свободное вхождение x_i в f не лежит в области действия никакого квантора Qx_j, где $Q \in \{\forall, \exists\}$, x_j — переменная, входящая в терм t.

Например, терм $f_1(x_1, x_3)$ свободен для x_1 в формуле

$$((\forall x_2 P_1(x_1, x_2)) \to P_2(x_1)),$$

но не свободен для x_1 в формуле

$$(\exists x_3(\forall x_2(P_1(x_1, x_2) \to P_2(x_1)))).$$

Если f — есть формула логики предикатов, x — свободная предметная переменная формулы f и t — терм, то через $S_t^x f$

обозначим результат подстановки в f терма t вместо всех свободных вхождений переменной x.

Лемма 15 (об интерпретациях). *Если f — есть формула логики предикатов, x — свободная предметная переменная формулы f и t — терм, свободный для переменной x в f, то для любой интерпретации I формула $(S_t^x f)_I$ определяет предикат, получающийся подстановкой в $(f)_I$ функции $(t)_I$ вместо переменной x.*

Доказательство этой леммы может получено индукцией по построению формулы f и рекомендуется в качестве самостоятельного упражнения.

Если формула f определяет в интерпретации I предикат, тождественно равный И, то она называется *истинной в I*, а I в этом случае называется *моделью для f*.

Формулу логики предикатов f называем *общезначимой*, если она истинна во всех интерпретациях, и *выполнимой*, если хотя бы в одной интерпретации она определяет не тождественно ложный предикат.

Если формула $f_1 \& \ldots \& f_n \to f_0$ общезначима, то f_0 называется *логическим следствием* формул $f_1, \ldots, f_n$, если формула $f_1 \leftrightarrow f_2$ общезначима, то формулы f_1 и f_2 называются *эквивалентными*.

Формула $((\neg P_1(x_1)) \vee P_1(x_1))$ есть пример общезначимой формулы. Формула $((\neg P_1(x_1)) \vee P_1(f_1(x_1)))$ — выполнима, но не общезначима.

Далее, как и в случае логики высказываний, договоримся опускать скобки, подразумевая, что $\neg$ — самая приоритетная операция, и в случае, когда порядок расставления скобок не существенен (например, для ассоциативных операций). Кроме того, договоримся опускать скобки в формулах вида $(Q_1(Q_2 A))$, где Q_1, Q_2 — любые кванторы. Например, вместо

$$(\forall x_1(\exists x_2(\forall x_3 P_1(x_1, x_2, x_3))))$$

будем писать $\forall x_1 \exists x_2 \forall x_3 P_1(x_1, x_2, x_3)$.

3.2.2 Полнота исчисления предикатов

Для указания дедуктивной системы, перечисляющей все общезначимые формулы логики предикатов, воспользуемся имеющимися у нас схемами аксиом А1)–А3), в которых теперь f, g, h — произвольные формулы логики предикатов, и добавим к ним следующие две схемы аксиом:

А4) Если f, g — формулы логики предикатов и x — предметная переменная, не являющаяся свободной переменной формулы f, то формула $(\forall x(f \to g)) \to (f \to (\forall x g))$ есть аксиома.

А5) Если f — есть формула логики предикатов, x — предметная переменная и t — терм, свободный для переменной x в f, то формула $(\forall x f) \to S_t^x f$ есть аксиома.

Правилами вывода в этой дедуктивной системе являются уже известное нам правило modus ponens, а также новое правило (известное как *правило обобщения*), позволяющее из формулы f выводить формулу $(\forall x f)$. Как и в случае логики высказываний, ограничиваемся только логическими связками $\neg$ и $\to$, причем из кванторов используем только квантор общности. Легко видеть, что квантор существования может быть выражен через квантор общности и отрицание: формулы $\exists_x A(x)$ и $\neg(\forall_x(\neg A(x)))$ эквивалентны.

Так как все аксиомы указанной дедуктивной системы общезначимы, а правила вывода сохраняют общезначимость, то все выводимые в ней формулы общезначимы. Обратно, имеет место следующее утверждение.

Теорема 21 (Геделя о полноте исчисления предикатов). *Каждая общезначимая формула логики предикатов является выводимой в приведенной выше дедуктивной системе.*

Таким образом, в исчислении предикатов, также как в исчислении высказываний, существует алгоритмическая процедура, перечисляющая все общезначимые формулы (например, основанная на описанной выше дедуктивной системе для логики предикатов) и позволяющая для любой общезначимой формулы за конечное число шагов установить ее общезначимость. С практической точки зрения, однако, данная процедура является чрезмерно трудоемкой, и развитие работ по автоматическому доказательству теорем в логике предикатов было связа-

но с поиском более приемлемых в "прикладном" отношении ее модификаций.

С другой стороны, в отличие от логики высказываний, для логики предикатов справедливо утверждение.

Теорема 22 (Черч). *Для логики предикатов не существует алгоритма, распознающего общезначимые формулы.*

Доказательства теоремы Геделя о полноте мы здесь не приводим, так как оно близко по своей технике к приводимому ниже доказательству полноты процедуры автоматического доказательства теорем, использующей правило резолюции. По завершении последнего мы вернемся к теореме Геделя и дадим краткий набросок схемы ее доказательства.

3.2.3 Доказательство общезначимости с помощью правила резолюции

Мы опишем здесь одну из процедур, позволяющую для любой общезначимой формулы за конечное число шагов установить ее общезначимость. Эта процедура связана с использованием при поиске доказательства так называемого "правила резолюции" — аналога уже известного нам правила для логики высказываний.

Согласно данной процедуре, доказательство общезначимости замкнутой формулы F складывается из следующих этапов.

Этап 1

Рассматривается формула $F_0 = \neg F$, и далее предпринимаются действия, направленные на установление невыполнимости формулы F_0 (то есть формула F доказывается "от противного").

Этап 2

К формуле F_0 применяется цепочка эквивалентных (в смысле определенной выше эквивалентности формул) преобразований:

а) используются эквивалентные замены $(f \leftrightarrow g) \rightarrow (f\&g \vee (\neg f)\&(\neg g)), (f \rightarrow g) \rightarrow (\neg f \vee g)$ позволяющие устранить логические связки вида $\leftrightarrow$, $\rightarrow$.

б) используются замены вида $(A * QxB) \to Qy(A * S_y^x B)$, где $*$ — связка $\vee$ либо $\&$, Q — квантор $\forall$ либо $\exists$, A и B — формулы , y — предметная переменная не входящая ни в A, ни в B. Кроме того используются замены $\neg\forall xA \to \exists x(\neg A)$, $\neg\exists xA \to \forall x(\neg A)$.

Замены указанные в пункте б), применяются до тех пор пока это возможно, они осуществляют перемещение кванторов к "началу" формулы, в результате чего F_0 преобразуется в формулу F_1 вида $Q_1x_1 \ldots Q_nx_nA$ где A — бескванторная формула, $Q_1, \ldots, Q_n$ — символы кванторов. Про F_1 говорят, что она находится в *предваренной нормальной форме.*

Этап 3

Лемма 16. *Пусть $f = \forall x_1 \ldots \forall x_k \exists x_{k+1} g$ — замкнутая формула логики предикатов, где $k \geq 0$, g — формула в предваренной нормальной форме; ϕ — функциональный символ арности k, не встречающийся в g (если $k = 0$, то ϕ — предметная константа); $f' = \forall x_1 \ldots \forall x_k g'$, где $g' = S^{x_{k+1}}_{\phi(x_1,\ldots,x_k)} g$ — результат замены всех свободных вхождений переменной x_{k+1} в g на $\phi(x_1, \ldots, x_k)$. Тогда f выполнима тогда и только тогда, когда f' выполнима.*

Доказательство. Предположим, что формула f выполнима, пусть I — интерпретация, в которой ее значение есть И. Рассмотрим предикат $(g)_I = h(x_1, \ldots, x_{k+1})$. Так как

$$(\forall x_1 \ldots \forall x_k \exists x_{k+1} g)_I = \text{И},$$

то для любых элементов $a_1, \ldots, a_k$ области M интерпретации I существует элемент a_{k+1} этой области такой, что

$$h(a_1, \ldots, a_{k+1}) = \text{И}.$$

Определим на M функцию $p(x_1, \ldots, x_k)$, полагая в указанной ситуации $p(a_1, \ldots, a_k) = a_{k+1}$. Тогда

$$h(x_1, \ldots, x_k, p(x_1, \ldots, x_k)) \equiv \text{И}$$

на M.

Пусть I' — интерпретация, отличающаяся от I только тем, что в ней символу ϕ сопоставляется функция $p(x_1, \ldots, x_k)$. Используя лемму 15 об интерпретациях, получим, что $(g')_{I'}$ есть

результат подстановки в $(g)_{I'} \equiv (g)_I = h(x_1, \ldots, x_k + 1)$ функции $(\phi(x_1, \ldots, x_k))_{I'} = p(x_1, \ldots, x_k)$ вместо переменной x_{k+1}, то есть $(g')_{I'} \equiv$ И на M. Таким образом, $(f')_{I'} =$ И, и формула f' выполнима.

Обратно, если f' выполнима, то рассматриваем интерпретацию I, в которой $(f')_I =$ И, то есть $(g')_I \equiv$ И на M, снова используя лемму об интерпретациях, находим, что $(g')_I$ получено подстановкой в функцию $(g)_I = h(x_1, \ldots, x_k, x_{k+1})$ функции $(\phi(x_1, \ldots, x_k))_I = p(x_1, \ldots, x_k)$ вместо переменной x_{k+1}, то есть для любых $a_1, \ldots, a_k$ из M выполнено

$$h(a_1, \ldots, a_k, p(a_1, \ldots, a_k)) = \text{И},$$

и $(f)_I =$ И.

Лемма доказана. □

Описанное в лемме 16 преобразование $f \to f'$ позволяет последовательно устранять кванторы $\exists$ из кванторной приставки, применяя его необходимое число раз к формуле F_1, получим в результате формулу F_2 вида $\forall x_1 \ldots \forall x_n A$, где A — бескванторная формула. Говорят, что F_2 находится в *сколемовской нормальной форме*. Заметим, что общезначимость исходной функции F эквивалентна невыполнимости формулы F_2.

Этап 4

Для анализа выполнимости формул f вида $\forall x_1 \ldots \forall x_n g$, где g — бескванторная формула, обычно используется теорема Эрбрана. Чтобы сформулировать эту теорему, введем понятие эрбрановского универсума H_f формулы f указанного вида. H_f есть множество термов, задаваемое посредством следующего индуктивного определения:

а) для каждой предметной константы a, встречающейся в f, однобуквенный терм a входит в H_f, если f не имеет предметных констант, то в H_f включается терм a_1, где a_1 — первая по алфавитному списку предметная константа;

б) если термы $t_1, \ldots, t_n$ принадлежат H_f и функциональный символ ϕ арности n встречается в формуле f, то терм $\phi(t_1, \ldots, t_n)$ входит в H_f.

Иными словами, H_f образовано всеми термами, которые можно построить при помощи предметных констант и функциональных символов, встречающихся в f (если f не имеет предметных констант, то разрешается использовать константу a_1).

Скажем, что *множество формул логики предикатов выполнимо*, если существует такая интерпретация, в которой каждая формула множества определяет не тождественно ложный предикат.

Имеет место следующее утверждение.

Теорема 23 (Эрбрана). *Замкнутая формула f вида $\forall x_1 \ldots \forall x_n g$, где g не содержит кванторов, выполнима тогда и только тогда, когда каждое конечное подмножество множества $\{S^{x_1,\ldots,x_n}_{t_1,\ldots,t_n} g : t_1, \ldots, t_n \in H_f\}$ выполнимо.*

Доказательство. Если в некоторой интерпретации I формула $\forall x_1 \ldots \forall x_n g$ имеет значение И, то в этой же интерпретации (согласно лемме об интерпретациях) истинны и все формулы $S^{x_1,\ldots,x_n}_{t_1,\ldots,t_n} g$.

Предположим теперь, что выполнимо каждое конечное подмножество множества $R = \{S^{x_1,\ldots,x_n}_{t_1,\ldots,t_n} g : t_1, \ldots, t_n \in H_f\}$, и будем доказывать выполнимость f.

Прежде всего, установим существование такой интерпретации I, в которой все формулы множества R имеют значение И.

Если R — конечно, то это сразу вытекает из сделанного предположения.

Пусть R бесконечно. Рассмотрим множество $A = \{A_1, A_2, \ldots\}$ всех встречающихся в формулах из R подформул вида $P(\tau_1, \ldots, \tau_m)$, где P — предикатный символ (напомним, что такие формулы называются атомарными или атомами). Легко видеть, что A бесконечно. Пусть R_i — подмножество множества R, состоящее из всех таких формул, в которых встречаются только атомы из $\{A_1, A_2, \ldots, A_i\}$. Очевидно, что $R = \cup_{i=1}^{\infty} R_i$, причем каждое R_i — конечно.

Так как каждая формула множества R построена из своих атомов при помощи одних только логических связок, то определение истинностных значений этих атомов в какой-либо интер-

претации позволяет однозначно определить и значение самой формулы. Ввиду выполнимости каждого R_i, мы можем рассмотреть теперь непустые множества V_i наборов $\alpha_1, \ldots, \alpha_i$ истинностных значений атомов $A_1, \ldots, A_i$, при которых все формулы из R_i имеют значение И.

Построим индуктивно бесконечное дерево T.

Базис индукции. Возьмем вершину v_0 и объявим ее корнем дерева T. В качестве первого яруса дерева берем элементы из V_1 и от v_0 к каждой вершине из V_1 проводим ребро.

Индуктивный переход. Пусть построено дерево T до яруса i и V_i — вершины этого яруса. Строить $(i+1)$-й ярус будем следующим образом. В качестве вершин возьмем элементы V_{i+1}. Если $(\alpha_1, \ldots, \alpha_{i+1}) \in V_{i+1}$, то понятно, что $(\alpha_1, \ldots, \alpha_i) \in V_i$, в этом случае вершины $(\alpha_1, \ldots, \alpha_i)$ и $(\alpha_1, \ldots, \alpha_{i+1})$ соединяем ребром.

Поскольку для каждого натурального i множество R_i выполнимо, а, значит, V_i не пусто, то дерево T — бесконечно. Поскольку по построению T связно, то согласно известной лемме Кенига из теории графов, в дереве T существует бесконечный путь $\pi = v_0, v_1, v_2, \ldots$, где $v_i \in V_i$ при $i = 1, 2, \ldots$. Согласно построению дерева T, если $v_i = (\alpha_1, \ldots, \alpha_i)$ и $v_{i+1} = (\beta_1, \ldots, \beta_i + 1)$, то $\alpha_1 = \beta_1, \ldots, \alpha_i = \beta_i$.

Сопоставим каждому натуральному i истинностное значение α_i, такое что $v_i = (\gamma_1, \ldots, \gamma_{i-1}, \alpha_i)$. Тогда, очевидно, для каждого $i = 1, 2, \ldots$ будет выполнено $v_i = (\alpha_1, \alpha_2, \ldots, \alpha_i)$.

Определим теперь в качестве области M интерпретации I множество H_f. Каждому символу предметной константы, встречающемуся в f, сопоставляем в качестве значения самого себя, функциональному символу ϕ, встречающемуся в f, сопоставляем функцию ϕ_I над термами из H_f, преобразующую набор $(t_1, \ldots, t_n)$ таких термов в терм $\phi(t_1, \ldots, t_n)$. Для каждого натурального i если $A_i = P(\tau_1, \ldots, \tau_m)$ (здесь $\tau_1, \ldots, \tau_m \in H_f$), то значение предиката P_I на наборе $(\tau_1, \ldots, \tau_m)$ полагаем равным α_i. В остальном интерпретацию I доопределяем произвольно.

По определению I атомы $A_1, A_2, \ldots$ имеют в ней истинностные значения $\alpha_1, \alpha_2, \ldots$. Если h — произвольная формула из R, то она входит в некоторое R_i, и принимает на наборе истинност-

ных значений $(\alpha_1, \ldots, \alpha_i)$ своих атомов $A_1, \ldots, A_i$, значение И, иными словами, $(h)_I$ = И, и существование требуемой интерпретации установлено.

Покажем, что $(f)_I$ = И. Пусть $(g)_I = G(x_1, \ldots, x_n)$. Если $(\tau_1, \ldots, \tau_n)$ — произвольный набор термов из H_f, то согласно лемме об интерпретациях имеем $G((\tau_1)_I, \ldots, (\tau_n)_I) = (S^{x_1,\ldots,x_n}_{\tau_1,\ldots,\tau_n} g)_I$ = И. С другой стороны, согласно определению I имеем $(\tau_1)_I = \tau_1, \ldots, (\tau_n)_I = \tau_n$. Отсюда $G(\tau_1, \ldots, \tau_n)$ = И, то есть G — тождественно истинный предикат, и $(\forall x_1 \ldots \forall x_n g)_I$ = И.

Теорема доказана. □

Согласно доказанной теореме, невыполнимость формулы F_2, полученной на этапе 3 и имеющей вид $\forall x_1 \ldots \forall x_n A$, эквивалентна существованию конечного подмножества множества $\{S^{x_1,\ldots,x_n}_{t_1,\ldots,t_n} A | t_1, \ldots, t_n \in H_{F_2}\}$, являющегося невыполнимым, так как проверка выполнимости конечного подмножества указанного множества формул является, по существу проверкой выполнимости конечного множества формул логики высказываний и алгоритмически реализуема, то для установления невыполнимости F_2 можно применять перебор таких конечных подмножеств. Эта процедура, однако, является чрезмерно трудоемкой, и наши дальнейшие действия будут использовать теорему Эрбрана лишь в неявном виде.

Этап 5

Используя уже описанную для логики высказываний процедуру, преобразуем формулу A к конъюнктивной нормальной форме и получим формулу A' вида $\&_{i=1}^{m} B_i$, где каждое B_i — дизъюнкция атомов либо их отрицаний.

Атомы и их отрицания называются *литерами*, дизъюнкции литер — *дизъюнктами*, логическая константа Л по определению также считается дизъюнктом.

Нетрудно видеть, что существование конечного невыполнимого подмножества множества $\{S^{x_1,\ldots,x_n}_{t_1,\ldots,t_n} A : t_1, \ldots, t_n \in H_{F_2}\}$ эквивалентно существованию конечного невыполнимого подмножества множества $\{S^{x_1,\ldots,x_n}_{t_1,\ldots,t_n} B_i : t_1, \ldots, t_n \in H_{F_2}, i =$

$1, \ldots, m\}$. Последнее нам удобно будет впоследствии формулировать как существование таких формул $f_1, \ldots, f_l$, не имеющих общих переменных и полученных из некоторых формул списка $\{B_1, \ldots, B_m\}$ переобозначением без отождествления их переменных $x_1, \ldots, x_n$, а также такой подстановки σ вместо переменных формул $f_1, \ldots, f_l$ термов множества H_{F_2}, что система $\{\sigma(f_1), \ldots, \sigma(f_l)\}$ — невыполнима. Здесь подстановка σ рассматривается как оператор на множестве формул.

Этап 6

Как и в случае логики высказываний, для установления невыполнимости формулы F_2 мы введем вспомогательную дедуктивную систему, аксиомами которой будут являться определенные выше дизъюнкты $B_1, \ldots, B_m$, и докажем, что выводимость в этой системе константы Л эквивалента невыполнимости F_2. Для формулировки правил вывода данной дедуктивной системы нам понадобится следующая *лемма об унифицирующей подстановке* (подстановку термов $t_1, \ldots, t_n$ вместо переменных $x_1, \ldots, x_n$ рассматриваем как оператор на множестве бескванторных формул и термов).

Лемма 17 (об унифицирующей подстановке). *Если $f_1, g_1, \ldots, f_k, g_k$ — бескванторные формулы либо термы и существует такая подстановка σ, что*

$$\sigma(f_1) = \sigma(g_1), \ldots, \sigma(f_k) = \sigma(g_k),$$

то существует минимальная подстановка σ_1, обладающая этим свойством, то есть

$$\sigma_1(f_1) = \sigma_1(g_1), \ldots, \sigma_1(f_k) = \sigma_1(g_k)$$

и для любой подстановки σ_2, удовлетворяющей условиям

$$\sigma_2(f_1) = \sigma_2(g_1), \ldots, \sigma_2(f_k) = \sigma_2(g_k),$$

выполнено: $\sigma_2 = \sigma_3\sigma_1$ при подходящей подстановке σ_3 (здесь умножение подстановок означает их последовательное применение).

Доказательство. Если рассматривать каждый входящий в бескванторную формулу либо терм f символ ϕ (предикатный

символ, функциональный символ либо логическая связка) арности n как обозначение n-местной операции над термами (в случае логической связки — над бескванторными формулами), преобразующей набор $(t_1, \ldots, t_n)$ в слово $\phi(t_1, \ldots, t_n)$, то f будет определять некоторый оператор f^*, который на наборе $(\tau_1, \ldots, \tau_m)$ значений переменных $x_1, \ldots, x_m$ будет равен $S^{x_1,\ldots,x_m}_{\tau_1,\ldots,\tau_m} f$. Пусть $x_1, \ldots, x_m$ — все переменные, встречающиеся в $f_1, g_1, \ldots, f_k, g_k$ из условия леммы. Тогда выполнение условий

$$\sigma(f_1) = \sigma(g_1), \ldots, \sigma(f_k) = \sigma(g_k)$$

для подстановки σ термов $\tau_1, \ldots, \tau_m$ вместо переменных $x_1, \ldots, x_m$ является решением системы уравнений

$$f_1^* = g_1^*, \ldots, f_k^* = g_k^*.$$

Для описания в явном виде всей совокупности ее решений используем следующие эквивалентные преобразования данной системы:

а) Если при некотором i слова f_i^*, g_i^* имеют, соответственно вид $\phi(t_1, \ldots, t_p)$ и $\psi(t'_1, \ldots, t'_p)$ (ϕ, ψ — предикатный символ, функциональный символ либо логическая связка $\neg$), то при $\phi \neq \psi$ делаем вывод о несовместности системы (согласно предположению леммы, это невозможно), а при $\phi = \psi$ (следовательно, при $p = q$) заменяем уравнение $f_i^* = g_i^*$ на группу уравнений $t_1^* = {t'_1}^*, \ldots, t_p^* = {t'_p}^*$.

б) Если f_i^* совпадает с g_i^*, то уравнение $f_i^* = g_i^*$ удаляется.

в) Если при некотором i одно из слов f_i^*, g_i^* есть переменная x, а другое терм t, $x \neq t$, то в случае, когда t содержит x, делаем вывод о несовместности системы (что невозможно по предположению леммы), иначе — подставляем t вместо x во все оставшиеся уравнения системы. Данное преобразование выполняется только при условии, что существует отличное от $f_i^* = g_i^*$ уравнение содержащее x.

Преобразование в) уменьшает число переменных x, имеющих явное выражение $x = t$ через остальные переменные системы и имеющих более одного вхождения в систему, и оно применимо лишь конечное число раз.

Преобразование а), в случае применения его к уравнениям максимально возможной длины, уменьшает число уравнений данной либо большей длины, и таким образом его возможно

применять лишь конечное число раз по завершении преобразований в)

Таким образом процесс применения к системе уравнений преобразований а), б), в) обрывается за конечное число шагов.

Результирующая система к которой эти преобразования неприменимы может иметь лишь вид $x^*_{i_1} = \theta^*_1, \ldots, x^*_{i_p} = \theta^*_p$, где термы $\theta_1, \ldots, \theta_p$ не содержат переменных $x_{i_1}, \ldots, x_{i_p}$.

Выберем теперь в качестве σ_1 подстановку термов $\theta_1, \ldots, \theta_p$ вместо переменных $x_{i_1}, \ldots, x_{i_p}$. Если переменные $x_{i_1}, \ldots, x_{i_p}$ имеют своими значениями термы $\theta_1, \ldots, \theta_p$, а каждая переменная x_j, $j \in \{1, \ldots, m\} \backslash \{i_1, \ldots, i_p\}$, имеет значением однобуквенный терм x_j, то уравнения $x^*_{i_1} = \theta^*_1, \ldots, x^*_{i_p} = \theta^*_p$ выполняются, а следовательно, $\sigma_1(f_1) = \sigma_1(g_1), \ldots, \sigma_1(f_k) = \sigma_1(g_k)$.

Предположим, что σ_2 — подстановка такая, что $\sigma_2(f_1) = \sigma_2(g_1), \ldots, \sigma_2(f_k) = \sigma_2(g_k)$. Без ограничения общности можно считать, что σ_2 подставляет термы $\tau_1, \ldots, \tau_q$ вместо переменных $x_1, \ldots, x_q$, $q \geq m$. Набор значений $\tau_1, \ldots, \tau_m$ переменных $x_1, \ldots, x_m$ удовлетворяет системе $x^*_{i_1} = \theta^*_1, \ldots, x^*_{i_p} = \theta^*_p$. Это означает, что выполнены соотношения:

$$\tau_{i_1} = S^{x_{j_1}, \ldots, x_{j_r}}_{\tau_{j_1}, \ldots, \tau_{j_r}} \theta_1, \ldots, \tau_{i_p} = S^{x_{j_1}, \ldots, x_{j_r}}_{\tau_{j_1}, \ldots, \tau_{j_r}} \theta_p,$$

где $\{j_1, \ldots, j_r\} = \{1, \ldots, m\} \backslash \{i_1, \ldots, i_p\}$. Если теперь выбрать в качестве σ_3 подстановку термов $\tau_{j_1}, \ldots, \tau_{j_r}, \tau_{m+1}, \ldots, \tau_q$ вместо переменных $x_{j_1}, \ldots, x_{j_r}, x_{m+1}, \ldots, x_q$, то получим $\sigma_2 = \sigma_3 \sigma_1$.

Лемма доказана. □

Подстановка σ_1 существование которой установлено в данной лемме, называется *унифицирующей подстановкой* для пар $(f_1, g_1), \ldots, (f_k, g_k)$.

В качестве примера рассмотрим процесс получения унифицирующей подстановки для для одной пары формул

$$(\neg P(a, y), \neg P(x, g(x))).$$

1) Изначально имеем уравнение $\neg P(a, y) = \neg P(x, g(x))$.

2) Применяя эквивалентное преобразование а), получим $P(a, y) = P(x, g(x))$.

3) Еще раз применяя эквивалентное преобразование а), получим систему уравнений $a = x$, $y = g(x)$.

4) Применяя эквивалентное преобразование в), получим унифицирующую подстановку $x = a$, $y = g(a)$.

Замыканием дизъюнкта D (обозначаемым $[D]$) будем называть формулу $\forall x_1 \forall x_k D$, где $x_1, \ldots, x_k$ — все входящие в D предметные переменные.

Если определенное на множестве дизъюнктов "правило вывода" позволяет выводить из дизъюнктов $D_1, \ldots, D_r$ лишь такие следствия D_0, что формула $[D_0]$ есть логическое следствие формул $[D_1], \ldots, [D_r]$, то такое правило вывода назовем *корректным.*

Как легко видеть, если при использовании некоторого набора корректных правил вывода из дизъюнктов $B_1, \ldots, B_m$, введенных на этапе 5, удается за конечное число шагов извлечь логическую константу Л, то формула F_2 невыполнима, а F — общезначима.

Предполагаемая здесь процедура автоматического доказательства теорем использует при поиске такого вывода константы Л следующие (как легко видеть, корректные) правила вывода:

R0 (*переобозначения переменных*). Из произвольного дизъюнкта D выводится дизъюнкт D', полученный из D переобозначением без отождествлений переменных, входящих в D.

R1 (*правило резолюции*). Если $f_1 \vee \cdots \vee f_s$ и $g_1 \vee \cdots \vee g_r$ — дизъюнкты, множества переменных которых не пересекаются, причем f_i имеет вид $P(t_1, \ldots, t_k)$, а g_j — вид $\neg P(t'_1, \ldots, t'_k)$ и σ — унифицирующая подстановка для пары

$$(P(t_1, \ldots, t_k), P(t'_1, \ldots, t'_k)),$$

$i \in \{1, \ldots, s\}$, $j \in \{1, \ldots, r\}$, то выводится дизъюнкт

$$\sigma(f_1) \vee \cdots \vee \sigma(f_{i-1}) \vee \sigma(f_{i+1}) \vee \cdots \vee \sigma(f_s) \vee \sigma(g_1) \vee \ldots \cdots \vee \sigma(g_{j-1}) \vee \sigma(g_{j+1}) \vee \cdots \vee \sigma(g_r)$$

(если $s = r = 1$, то выводится константа Л).

Дизъюнкт получаемый согласно правилу R1, называется *резольвентой* исходных дизъюнктов.

R2 (*правило склеивания*). Если $f_1 \vee \cdots \vee f_s$ — дизъюнкт и σ — унифицирующая подстановка для (f_i, f_j), $i \neq j$, $i, j \in \{1, \ldots, s\}$, то выводится дизъюнкт $\sigma(f_1) \vee \cdots \vee \sigma(f_{i-1}) \vee \sigma(f_{i+1}) \vee \cdots \vee \sigma(f_s)$.

Теорема 24. *Если формула F_2 невыполнима и ее конъюнктивная нормальная форма имеет вид $\&_{i=1}^{m} B_i$, то за конечное число шагов из дизъюнктов $B_1, \ldots, B_m$ при помощи правил вывода R0,R1,R2 может быть выведена логическая константа Л.*

Доказательство. Как было замечено в конце этапа 5, невыполнимость F_2 влечет существование таких формул $f_1, \ldots, f_l$, не имеющих общих переменных и полученных из некоторых формул списка $\{B_1, \ldots, B_m\}$ переобозначением без отождествления их переменных $x_1, \ldots, x_n$, а также такой подстановки σ вместо переменных формул $f_1, \ldots, f_l$ термов множества H_{F_2}, что система $\{\sigma(f_1), \ldots, \sigma(f_l)\}$ невыполнима.

Каждая формула $\sigma(f_i)$ имеет вид $A_1^{\alpha_1} \vee \cdots \vee A_s^{\alpha_s}$, где $A_1, \ldots, A_s$ — атомы без переменных, $\alpha_1, \ldots, \alpha_s$ — логические константы. Если некоторые $A_j^{\alpha_j}$ в этой формуле совпадают между собой, то после их отождествления формула $\sigma(f_i)$ преобразуется в некоторый дизъюнкт, который обозначим d_i.

Пусть $A_1, \ldots, A_n$ — все различные атомы, встречающиеся в дизъюнктах $d_1, \ldots, d_l$, $n \geq 1$. Множество $M = \{d_1, \ldots, d_l\}$ представим в виде $M_1 \cup M_2 \cup M_3$, где M_1 — все дизъюнкты, в которые не входит A_1, M_2 — все дизъюнкты вида $A_1 \vee B$, M_3 — все дизъюнкты вида $\neg A_1 \vee B$.

Рассмотрим множество M_4, образованное всеми дизъюнктами вида $B \vee C$, где $A_1 \vee B \in M_2$, $\neg A_1 \vee C \in M_3$ (если $A_1 \in M_2$, $\neg A_1 \in M_3$, то в M_4 включается Л). Обозначим $M' = M_1 \cup M_4$. Понятно, что множество M' не содержит атома A_1.

Так как $\{d_1, \ldots, d_l\}$ — невыполнимое множество дизъюнктов, то для любых истинностных значений $a_1, \ldots, a_n$ атомов $A_1, \ldots, A_n$ существует формула $d \in M$, имеющая значение Л. Если $d \in M_1$, то $d \in M'$. Если такой формулы d в M_1 нет, то при $a_1 =$ И существует формула $d' = \neg A_1 \vee C$ имеющая значением Л, а при $a_1 =$ Л — формула $d'' = A_1 \vee B$, имеющая значение Л, $d' \in M_3$, $d'' \in M_2$. Таким образом, варьируя при фиксированных $a_2, \ldots, a_n$, значение a_1, находим указанные формулы d', d'' и получаем, что формула $d''' = B \vee C$, принадлежащая M_4, а следовательно и M', имеет на наборе значений $a_2, \ldots, a_n$ атомов $A_2, \ldots, A_n$ значение Л.

Мы получили, что для любых истинностных значений ато-

мов $A_2, \ldots, A_n$ в M' имеется формула, значение которой есть Л, т.е. M' невыполнимо и имеет не более $n-1$ атома.

Пусть дизъюнкт $B \vee C$ из M_4 получен из дизъюнктов $A_1 \vee B \in M_2$ и $\neg A_1 \vee C \in M_3$. $A_1 \vee B$ возникло из некоторого $\sigma(f_i)$ отождествлением одинаковых литер. Следовательно, $\sigma(f_i) = \sigma_1'\sigma_1''(f_i)$, где σ_1'' унифицирует те литеры из f_i, которые совпадают после применения к ним σ. Аналогично, $\sigma(f_j) = \sigma_2'\sigma_2''(f_j)$. Так как переменные в f_i и f_j различны, то можно считать $\sigma_2'' = \sigma_1'' = \sigma''$, $\sigma_2' = \sigma_1' = \sigma'$. Нетрудно видеть, что $\sigma''(f_i)$ и $\sigma''(f_j)$ получены из f_i, f_j несколькими применениями правила R2.

Выделяя в $\sigma''(f_i)$ и $\sigma''(f_j)$ те атомы, применение подстановки σ' к которым дает, соответственно, A_1 и $\neg A_1$, будем иметь для $\sigma''(f_i)$ и $\sigma''(f_j)$ следующие представления:

$$\sigma''(f_i) = P(\theta_1, \ldots, \theta_s) \vee B',$$

$$\sigma''(f_j) = \neg P(\theta_1', \ldots, \theta_s') \vee C',$$

$$\sigma'(P(\theta_1, \ldots, \theta_s)) = A_1,\ \sigma'(B') = B,$$

$$\sigma'(P(\theta_1', \ldots, \theta_s')) = A_1,\ \sigma'(C') = C.$$

Рассматривая далее подстановку ρ, унифицирующую пару $(P(\theta_1, \ldots, \theta_s), P(\theta_1', \ldots, \theta_s'))$, получим, что для некоторой подстановки ρ' выполняется $\sigma' = \rho'\rho$. При этом дизъюнкт $\rho(B') \vee \rho(C')$ получен из $\sigma''(f_i)$, $\sigma''(f_j)$ по правилу R1, причем $B \vee C = \rho'(\rho(B') \vee \rho(C'))$.

Проведенные рассуждения показывают, что применяя к $\{f_1, \ldots, f_l\}$ правила R0, R1, R2 можно получить такие формулы $\{g_1, \ldots, g_r\}$, не имеющие общих переменных, что $M_1 \cup M_4 = \{\tilde{\sigma}(g_1), \ldots, \tilde{\sigma}(g_r)\}$ для подходящей подстановки $\tilde{\sigma}$, то есть для $g_1, \ldots, g_r$ выполнены те же условия, что и для $f_1, \ldots, f_l$, и число атомов в формулах $\{\tilde{\sigma}(g_1), \ldots, \tilde{\sigma}(g_r)\}$, хотя бы на один меньше, чем в $\{\sigma(f_1), \ldots, \sigma(f_l)\}$.

Продолжая описанный процесс логического вывода, за конечное число шагов получим систему $\{h_1, \ldots, h_q\}$, удовлетворяющую указанным условиям, причем такую, что для нее число атомов в соответствующем семействе $\{\tilde{\sigma}(h_1), \ldots, \tilde{\sigma}(h_q)\}$ окажется равно 0. Это означает, что $q = 1$ и $\sigma(h_1) = \text{Л}$, а следовательно, и $h_1 = \text{Л}$, и утверждение доказано. □

Приведем простой пример применения описанной выше процедуры автоматического доказательства теорем логики предикатов. Формула F, общезначимость которой требуется установить имеет вид:

$$(\forall x\exists y(P(x,y)\vee Q(x,y))\&\forall x\forall y(P(x,y)\to R(x,f(x,y)))\&$$
$$\&\forall x\forall y(Q(x,y)\to R(x,f(x,y)))\to\forall x\exists yR(x,y)).$$

Отрицание F_0 формулы F представляет собой конъюнкцию следующих формул:

$$\forall x\exists y(P(x,y)\vee Q(x,y));$$
$$\forall x\forall y(P(x,y)\to R(x,f(x,y)));$$
$$\forall x\forall y(Q(x,y)\to R(x,f(x,y)));$$
$$\exists x\forall y\neg R(x,y).$$

Нетрудно проверить, что описанные на этапах 2)–4) преобразования, позволяющие определить исходную совокупность дизъюнктов $B_1,\ldots,B_m$, можно применять к каждой из этих формул по отдельности. В первой формуле избавляемся от квантора существования, вводя в рассмотрение новый функциональный символ g: $\forall x(P(x,g(x))\vee Q(x,g(x)))$, в последней формуле для этой цели используем новую предметную переменную a: $\forall y\neg R(a,y)$. Окончательно, получаем следующую систему дизъюнктов:

1) $P(x,g(x))\vee Q(x,g(x))$,
2) $\neg P(x,y)\vee R(x,f(x,y))$,
3) $\neg Q(x,y)\vee R(x,f(x,y))$,
4) $\neg R(a,y)$.

Далее переобозначаем переменную y в дизъюнкте 4 на y' и унифицируем $R(a,y')$ с $R(x,f(x,y))$ из дизъюнкта 2: $a=x$, $y'=f(x,y)$, $y'=f(a,y)$. Таким образом, унифицирующая подстановка имеет вид $S^{x,y'}_{a,f(a,y)}$, и с ее помощью получаем по правилу R1 новый дизъюнкт:

5) $\neg P(a,y)$

Далее из дизъюнктов 3, 4 получаем аналогичным образом:

6) $\neg Q(a,y)$

Унифицируем $P(a,y)$ с $P(x,g(x))$ из дизъюнкта 1: $x=a$, $y=g(a)$, и получаем следствие дизъюнктов 5, 1:

7) $Q(a, g(a))$

Наконец, унифицируем $Q(a, g(a))$ и $\neg Q(a, y)$ (то есть дизъюнкты 6,7) и выводим константу Л.

3.2.4 Связь с теоремой Геделя о полноте

В описанной выше процедуре автоматического доказательства теорем была использована некоторая вспомогательная дедуктивная система, использующая три правила вывода — переобозначение переменных, правило резолюции и правило склеивания. Роль аксиом в ней играли дизъюнкты, полученные в процессе преобразований отрицания исходной теоремы. "Полнота" этой дедуктивной системы заключалась в возможности получения за конечное число шагов константы "ложь" для любого невыполнимого множества дизъюнктов. Оказывается, что использованная при установлении данной полноты техника весьма близка к технике, применяемой для доказательства теоремы Геделя о полноте приведенной в начале раздела классической дедуктивной системы логики предикатов. Чтобы провести сопоставление, дадим здесь краткий набросок схемы доказательства теоремы Геделя.

Прежде всего, нам понадобится понятие *теории первого порядка*. Это — произвольная дедуктивная система, полученная из дедуктивной системы логики предикатов добавлением некоторого (конечного либо бесконечного) множества аксиом. Теорию первого порядка называют *противоречивой*, если в ней выводимы одновременно какая-либо формула и ее отрицание. Очевидно, в таком случае в ней выводима вообще любая формула. Одна теория первого является *расширением* другой, если каждая формула, выводимая во второй теории, выводима также в первой. Наконец, введем понятие *полной* теории, т.е. такой, в которой для любой замкнутой формулы выводима либо она, либо ее отрицание.

Первым шагом доказательства теоремы Геделя является установление следующей леммы Линденбаума:

Лемма 18. *Если теория первого порядка непротиворечива, то существует непротиворечивое полное ее расширение.*

Доказательство леммы основано на рассмотрении последо-

вательности $F_0, F_1, \ldots$, перечисляющей все замкнутые формулы логики предикатов. Начиная с исходной непротиворечивой теории T, по этой последовательности строится последовательность теорий $T = T_0, T_1, \ldots$. Каждый раз, как оказывается, что формула $\neg F_i$ не выводима в теории T_i, теория T_{i+1} получается из T_i добавлением F_i в качестве аксиомы. В противном случае полагается $T_{i+1} = T_i$. Затем рассматривается теория T', множество аксиом которой является объединением множеств аксиом всех теорий $T_0, T_1, \ldots$. Как легко видеть, T' представляет собой непротиворечивое полное расширение теории T. Данная конструкция вполне аналогична конструкции, применявшейся при доказательстве теоремы Эрбрана, когда определялась бесконечная последовательность истинностных значений термов эрбрановского универсума.

Если в некоторой интерпретации логики предикатов истинны все аксиомы теории первого порядка T, то называем ее *моделью* теории T. Далее доказывается следующая лемма:

Лемма 19. *Всякая непротиворечивая теория первого порядка имеет модель со счетной областью.*

Пусть T — непротиворечивая теория первого порядка. Рассмотрим последовательность $F_1(x_1), F_2(x_2), \ldots$, в которой перечисляются все формулы логики предикатов, содержащие не более одной свободной переменной. Пусть также $a_1, a_2, \ldots$ — последовательность различных предметных констант, таких, что a_i не содержится в формулах $F_1(x_1), \ldots, F_i(x_i)$. Введем в рассмотрение формулу S_n вида $\neg\forall_{x_i} F_i(x_i) \to \neg F_i(a_i)$. Определяем теорию T_n как результат присоединения к T аксиом $S_1, \ldots, S_n$, а теорию T' — как результат присоединения к T аксиом $S_1, S_2, \ldots$. Очевидно, что для установления непротиворечивости T' достаточно установить непротиворечивость каждой теории T_n; $n = 1, 2, \ldots$. Последнее нетрудно установить индукцией по n. Заметим, что формула S_n эквивалентна формуле $\exists_{x_i} G_i(x_i) \to G_i(a_i)$, где $G_i(x_i)$ есть $\neg F_i(x_i)$. Такое представление объясняет смысл добавления аксиом S_n: они позволяют перейти от формулы с внешним квантором существования к формуле, полученной из нее отбрасыванием квантора и подстановкой "новой" константы вместо подкванторной переменной. Этот переход аналогичен переходу, выполнявшемуся выше при

получении сколемовской нормальной формы.

После установления непротиворечивости теории T' применяем лемму Линденбаума, согласно которой она имеет непротиворечивое полное расширение T''. Используя теорию T'', непосредственно указываем интерпретацию для теории T, в которой истинны все ее аксиомы. Эта интерпретация строится аналогично тому, как строилась интерпретация в доказательстве теоремы Эрбрана. Областью интерпретации здесь служит множество M всех термов, не имеющих переменных. Очевидно, оно счетно. Интерпретацией константы служит она сама; функциональный символ f арности n интерпретируется операцией, переводящей термы $t_1, \ldots, t_n$ в терм $f(t_1 \ldots t_n)$. Отношение, являющееся интерпретацией предикатного символа P арности n, истинно на наборе термов $t_1, \ldots, t_n$ тогда и только тогда, когда формула $P(t_1 \ldots t_n)$ выводима в теории T''. Доказательство того, что в указанной интерпретации будет истинной любая формула F, выводимая в теории T'', осуществляется сравнительно несложной индукцией по построению формулы F. Так как любая формула, выводимая в T, выводима также в теории T'', получаем отсюда, что построенная интерпретация является моделью для T.

Чтобы доказать теперь теорему Геделя о полноте, рассмотрим произвольную замкнутую общезначимую формулу F логики предикатов. Если эта формула не выводима в логике предикатов, то можно присоединить ее отрицание к аксиомам и получить непротиворечивую теорию первого порядка. По доказанной лемме, она имеет модель. В этой модели одновременно должны быть истинны и общезначимая формула F, и ее отрицание. Полученное противоречие и устанавливает выводимость формулы F.

3.2.5 Неполнота формальной арифметики

Хотя для любой непротиворечивой теории первого порядка T существует непротиворечивое полное ее расширение T', это еще не означает существования алгоритма, распознающего по формуле, является ли она аксиомой теории T' или хотя бы перечисляющего все аксиомы этой теории. Гедель установил, что любая

непротиворечивая теория первого порядка, для которой имеется алгоритм, распознающий ее аксиомы, и притом включающая в себя формализацию простейших свойств целых неотрицательных чисел, обязательно является неполной. Это утверждение, известное как теорема Геделя о неполноте формальной арифметики, часто упоминается в связи философскими аспектами искусственного интеллекта, и мы приведем здесь краткую схему рассуждений, лежащих в основе его доказательства.

Прежде всего, приведем список аксиом, присоединение которых к общим аксиомам логики предикатов дает теорию первого порядка, называемую формальной арифметикой. В этих аксиомах выделены следующие специальные символы: символ предметной константы, обозначаемый далее 0; функциональный символ арности 1, обозначаемый штрихом; функциональные символы арности 2, обозначаемые знаками сложения и умножения; предикатный символ арности 2, обозначаемый символом равенства. Напомним, что язык логики предикатов имеет счетное множество символов предметных констант и счетное множество функциональных либо предикатных символов произвольной арности. Аксиомами служат следующие формулы:

1. $x_1 = x_2 \to (x_1 = x_3 \to x_2 = x_3)$
2. $x_1 = x_2 \to x_1' = x_2'$
3. $\neg(0 = x_1')$
4. $x_1' = x_2' \to x_1 = x_2$
5. $x_1 + 0 = x_1$
6. $x_1 + x_2' = (x_1 + x_2)'$
7. $x_1 \cdot 0 = 0$
8. $x_1 \cdot x_2' = x_1 \cdot x_2 + x_1$
9. $A(0) \to (\forall_x(A(x) \to A(x')) \to \forall_x A(x))$

Здесь x, x_1, x_2, x_3 — произвольные предметные переменные; $A(x)$ — произвольная формула логики предикатов.

Областью стандартной интерпретации для формальной арифметики служит множество целых неотрицательных чисел; предметной константе 0 соответствует число 0; функциональному символу "штрих" соответствует операция прибавления единицы; сложение, умножение и равенство понимаются обычным образом. Последняя из приведенных выше схем аксиом представляет собой формализацию принципа математической индукции. Для любого целого неотрицательного числа k будем обозначать $T(k)$ терм, полученный последовательным k-кратным применением к 0 операции $'$. В стандартной интерпретации значением такого терма будет служить число k.

Отношение $R(x_1 \dots x_n)$, определенное на множестве наборов целых неотрицательных чисел, называется *выразимым в формальной арифметике*, если существует формула логики предикатов $A(x_1 \dots x_n)$, такая, что для любых целых неотрицательных чисел $k_1, \dots, k_n$:

а) Если $R(k_1 \dots k_n)$ истинно, то выводима формула

$$A(T(k_1) \dots T(k_n));$$

б) Если $R(k_1 \dots k_n)$ ложно, то выводима формула

$$\neg A(T(k_1) \dots T(k_n)).$$

Выводимость здесь понимается относительно аксиом формальной арифметики.

Функция $f(x_1 \dots x_n)$, определенная на множестве наборов целых неотрицательных чисел и принимающая целые неотрицательные значения, называется *представимой в формальной арифметике*, если существует формула $A(x_1 \dots x_n x_{n+1})$, такая, что для любых целых неотрицательных чисел $k_1, \dots k_n, k_{n+1}$:

а) Если $f(k_1 \dots k_n) = k_{n+1}$, то выводима формула

$$A(T(k_1) \dots T(k_n) T(k_{n+1}));$$

б) Выводима формула $\exists! x_{n+1} A(T(k_1) \dots T(k_n) x_{n+1})$, где знак ! после квантора существования — сокращенное обозначение для формулы, выражающей существование и единственность.

Выделим специальный класс арифметических (т.е. определенных на множестве целых неотрицательных чисел и принимающих целые неотрицательные значения) функций, определяе-

мый индуктивным образом. Этот класс дает формализацию понятия эффективно вычислимой арифметической функции, т.е. такой функции, для вычисления значений которой существует алгоритм. Базис индукции составляют следующие функции:

1. Нуль-функция $Z(x)$, тождественно равная 0;
2. Функция прибавления единицы: $N(x) = x + 1$;
3. Проектирующие функции: $U_{i,n}(x_1 \dots x_n) = x_i$

Операции, порождающие новые функции из ранее построенных, таковы:

1. Подстановка — получение функции
$$f(x_1 \dots x_n) = g(h_1(x_1 \dots x_n), \dots, h_m(x_1 \dots x_n))$$
из функций $g(y_1 \dots y_m), h_1(x_1 \dots x_n), \dots, h_m(x_1 \dots x_n)$;
2. Примитивная рекурсия — получение функции
$$f(x_1 \dots x_n, x_{n+1})$$
из функций $g(x_1 \dots x_n), h(x_1 \dots x_{n+2})$ с помощью соотношений:
$f(x_1 \dots x_n\ 0) = g(x_1 \dots x_n)$;
$f(x_1 \dots x_n\ x_{n+1} + 1) = h(x_1 \dots x_n\ x_{n+1}\ f(x_1 \dots x_{n+1}))$;
3. Минимизация — получение функции $f(x_1 \dots x_n)$ из функции $g(x_1 \dots x_{n+1})$, для которой уравнение $g(x_1 \dots x_n\ y) = 0$ имеет хотя бы один корень y при любом наборе целых неотрицательных $x_1, \dots, x_n$. Значением функции $f(x_1 \dots x_n)$ является наименьший из указанных корней y.

Функции, принадлежащие заданному таким индуктивным определением классу, называются *рекурсивными*. Если ограничиться лишь операциями подстановки и примитивной рекурсии, то возникает собственный подкласс класса рекурсивных функций, называемый классом *примитивно-рекурсивных функций*. Оказывается, что каждая рекурсивная функция представима в формальной арифметике, а каждое рекурсивное арифметическое отношение (т.е. отношение на множестве наборов целых

неотрицательных чисел, имеющее рекурсивную характеристическую функцию) выразимо в ней. Доказываются эти утверждения индукцией по определению класса рекурсивных функций. Чтобы с помощью арифметических отношений и функций можно было определять различные свойства конечных последовательностей символов в алфавите языка логики предикатов, вводится специальная нумерация этих последовательностей. Сначала символам s алфавита сопоставляются взаимнооднозначно их номера $N(s)$, после чего последовательности s_1, $s_2, ..., s_k$ сопоставляется номер $2^{N(s_1)} \cdot 3^{N(s_2)} \cdot \ldots \; cdotp_k^{N(s_k)}$. Здесь p_i — простое число с номером i; $i = 1, \ldots, k$. Легко видеть, что по своему номеру последовательность восстанавливается однозначно. Теперь для таких свойств слов в алфавите языка логики предикатов, как, например, свойство быть термом, формулой, аксиомой, последовательностью формул, представляющей собой вывод, и т.п., можно рассматривать соответствующие арифметические отношения для номеров слов. Этот прием перевода свойств конечных последовательностей символов на арифметический язык был предложен Геделем и называется *геделевой нумерацией*. Номер, сопоставляемый слову указанным выше образом, называем *геделевым номером* этого слова.

Используя утверждение о выразимости в формальной арифметике любых рекурсивных отношений, далее можно доказать выразимость в ней следующих двух отношений $W_1(u, v)$ и $W_2(u, v)$:

1. $W_1(u, v)$ истинно тогда и только тогда, когда u — геделев номер формулы $A(x_1)$, имеющей свободную переменную x_1 (первую переменную списка предметных переменных алфавита языка логики предикатов), и v есть геделев номер некоторого вывода в формальной арифметике формулы $A(T(u))$.

2. $W_2(u, v)$ истинно тогда и только тогда, когда u есть геделев номер формулы $A(x_1)$, имеющей свободную переменную x_1, и v есть геделев номер вывода в формальной арифметике формулы $\neg A(T(u))$.

Оба эти свойства, очевидным образом, допускают алгоритмическую проверку, и доказательство рекурсивности отноше-

ний W_1, W_2 сводится, по существу, к обычному программированию.

Рассмотрим теперь формулы $U_1(x_1, x_2), U_2(x_1, x_2)$, с помощью которых выразимы отношения W_1, W_2. Построим формулу U следующего вида: $\forall_{x_2}(U_1(x_1, x_2) \to \exists_{x_3}(x_3 \leq x_2 \,\&\, U_2(x_1, x_3)))$. Пусть n — геделев номер этой формулы. Строим далее формулу V, получаемую подстановкой в U терма $T(n)$ вместо переменной x_1: $\forall_{x_2}(U_1(T(n), x_2) \to \exists_{x_3}(x_3 \leq x_2 \,\&\, U_2(T(n), x_3)))$.

Теорема Геделя о неполноте формальной арифметики (в так называемой форме Россера, несколько усиливающей первоначальное утверждение Геделя) утверждает, что если формальная арифметика непротиворечива, то в ней не выводима ни формула V, ни ее отрицание, т.е. она неполна.

Смысл утверждения V достаточно прозрачен: оно говорит, что если x_2 есть геделев номер вывода утверждения V, то существует вывод отрицания утверждения V, имеющий геделев номер, не превосходящий x_2. Это — удобная в техническом отношении версия утверждения, которое утверждает свою собственную ложность. Как известно, оба допущения — об истинности либо о ложности такого рода утверждений, сразу приводят к противоречию. На этой стандартной схеме и построено доказательство теоремы. Завершающая его часть (после установления рекурсивности W_1, W_2) несложна, и подробности мы опустим.

Заметим, что приведенные рассуждения по аналогии могут быть воспроизведены и для любого расширения формальной арифметики, у которого множество аксиом рекурсивно, т.е. все такие расширения неполны.

Конечно, философская интерпретация теоремы Геделя о неполноте как доказательства "принципиальной ограниченности аксиоматического подхода" требует большой осторожности. Рассматриваемое в ее доказательстве утверждение не несет совершенно никакой содержательной информации, утверждая лишь свою собственную ложность. Оно может быть воспринято как пример утверждения совершенно бессмысленного и именно из-за своей бессмысленности оказывающегося вне рамок действия аксиоматизации. С другой стороны, из математической практики известны конкретные примеры вполне содержательных утверждений, таких, как континуум-гипотеза, оказавших-

ся вне области действия общепринятой аксиоматизации и таким образом выявившие ее неполноту. Теорема Геделя оставляет открытым вопрос о том, сколь значительной может оказаться неполнота аксиоматизаций, если ограничиваться рассмотрением в них лишь достаточно осмысленных и содержательных утверждений, подобных континуум-гипотезе.

С точки зрения искусственного интеллекта, который лишь моделирует логические процессы естественного интеллекта, теорема Геделя вообще не накладывает никаких дополнительных ограничений на первый, которые отсутствовали бы для второго. Точно так же, как человек, выявив независимость какого-либо утверждения от имеющейся аксиоматики, начинает рассматривать две независимых ветви теории либо, опираясь на практику, выбирает одну из них, так и компьютерная модель, воспроизводящая шаг за шагом логику рассуждений человека, в принципе, могла бы выполнять аналогичные действия.

3.2.6 Эвристики в управлении выводом

Вообще говоря, процесс поиска вывода константы Л из исходных дизъюнктов $B_1, \ldots, B_m$ при помощи правил вывода R0, R1, R2 может оказаться неприемлемо трудоемким, даже в рассматриваемом выше простом примере можно было бы указать достаточное количество альтернативных способов применения правил вывода (например, получить резольвенты дизъюнктов 1 и 2, либо 1 и 3).

Трудоемкость поиска вывода для описанной процедуры растет экспоненциально с ростом глубины вывода, и это заставляет предпринимать исследования, направленные на нахождение принципов управления выводом следствий. Мы приведем здесь краткий обзор такого рода принципов, которые были найдены при практическом использовании указанной процедуры, а также при теоретическом ее исследовании. Эти принципы можно разбить на две группы — принципы исключения из рассмотрения некоторых "избыточных" следствий, с доказательством сохранения полноты процедуры при их применении (они получили название "стратегии очищения"), и эвристические принципы упорядочения перебора резольвент ("стратегии упорядо-

чения"). Начнем с простейших примеров таких стратегий.

1) **Стратегия предпочтения одночленов**. Как нетрудно заметить, применение правила резолюции к дизъюнктам $A \vee K_1 \vee \cdots \vee K_{m-1}$ и $\neg A' \vee K'_1 \vee \cdots \vee K'_{n-1}$ длины m и n соответственно дает резольвенту $K''_1 \vee \cdots \vee K''_{m-1} \vee K'''_1 \vee \cdots \vee K'''_{n-1}$ длины $m+n-2$. Длина эта лишь в том случае окажется меньшей длины одного из исходных дизъюнктов, когда другой дизъюнкт имел длину 1, то есть являлся "одночленом". Исходя из такого, вообще говоря, эвристического соображения получения следствий, более простых чем исходные формулы, стратегия предпочтения одночленов рекомендует первоочередное применение правила резолюции к одночленам.

2) **Исключение тавтологий и уникальных литер**: в процессе вывода следствий отбрасываются дизъюнкты вида $(A \vee \neg A \vee B_1 \vee \cdots \vee B_k)$ ("тавтологии"). Если предикатный символ P встречается во всех дизъюнктах только с отрицанием, либо только без отрицания, то все содержащие P дизъюнкты отбрасываются.

Стратегия предпочтения одночленов дает пример стратегии упорядочения, принцип исключения тавтологий и уникальных литер — пример стратегии очищения.

3) **Стратегия первоочередного применения правила склеивания**: правило R1 применяется лишь после того, как были получены все возможные в текущей ситуации следствия с применением правила R2.

4) **Стратегия использования подслучаев**. Дизъюнкт D называется *подслучаем* дизъюнкта C, если существует такая подстановка σ, что C имеет вид $\sigma(D) \vee C'$. Стратегия заключается в отбрасывании всех возникающих при выводе дизъюнктов C, для которых уже найден был ранее некоторый их подслучай (данная стратегия является стратегией очищения и гарантирует получение за конечное число шагов константы Л).

5) **Стратегия применения несущего множества дизъюнктов**. Эта стратегия связана с выделением в исходном множестве дизъюнктов M такого, возможно большего подмножества M', про которое известно, что оно выполнимо (например M' может быть образовано всеми аксиомами, из которых требуется извлечь заданное следствие). При выводе следствий допус-

кается лишь такое применение правила резолюции, когда хотя бы один из несущих дизъюнктов принадлежит *несущему множеству*. Последнее определяется следующим индуктивным образом:

а) Каждый элемент из $M \backslash M'$ относится к несущему множеству.

б) Если резольвента получена при участии хотя бы одного элемента несущего множества, то она входит в несущее множество.

в) Результаты применения правила R0 либо R2 к элементу несущего множества дают элемент несущего множества.

6) **стратегия использования линейных выводов**. Линейный вывод представляет собой последовательность применений правила резолюции определяемую диаграммой следующего вида:

$$
\begin{array}{lll}
D_0 & & C_1 \\
\downarrow & \swarrow & \\
D_1 & & C_2 \\
\downarrow & \swarrow & \\
D_2 & & C_3 \\
\downarrow & \swarrow & \\
\vdots & & C_n \\
\downarrow & \swarrow & \\
D_n & &
\end{array}
$$

Здесь каждое C_i — либо элемент исходного множества дизъюнктов M, либо есть некоторое D_j при $j \in \{0, 1, \ldots, i-1\}$. Имеет место утверждение (Андерсон-Бледсоу), согласно которому для получения константы Л достаточно использовать лишь линейные выводы (здесь и далее мы рассматриваем лишь встречающиеся в выводах правила R1, так как именно они ответственны за экспоненциально нарастающую с увеличением глубины вывода трудоемкость поиска доказательства, не упоминая о возможных применениях "одноместных" правил R0 и R2). Заметим, что само по себе ограничение линейности вывода не устраняет древовидной схемы поиска линейного вывода при переборе

различных возможностей для $C_1, C_2, \ldots, C_n$.

7) **Совместное применение стратегий использования подслучаев и слияния**. Введем сначала вспомогательные понятия *слияния* и *литеры слияния*. Резольвента $\sigma(D'_1) \vee \cdots \vee \sigma(D'_n) \vee \sigma(D''_1) \vee \cdots \vee \sigma(D''_m)$ двух дизъюнктов $A \vee D'_1 \vee \cdots \vee D'_n$ и $\neg A \vee D''_1 \vee \cdots \vee D''_m$ называется *слиянием* этих дизъюнктов, если существуют такие $i \in \{1, \ldots, n\}$ и $i \in \{1, \ldots, m\}$, что $\sigma(D'_i) = \sigma(D''_j)$. Формула $\sigma(D'_i)$ при этом называется *литерой слияния*. Андерсону и Бледсоу принадлежит утверждение о том, что при поиске вывода константы Л достаточно ограничится лишь такими линейными выводами, у которых каждое C_i (обозначения те же, что в пункте 6) либо принадлежит исходному множеству дизъюнктов M, либо является некоторым D_j, $j < i$, полученным при слиянии двух дизъюнктов, причем литерой (унифицируемой при применении правила резолюции формулой A либо $\neg A'$)в C_i при получении D_i является некоторая литера слияния, а само D_i является подслучаем дизъюнкта D_{i-1}. Несмотря на существенное ограничение в использовании формул C_i, не принадлежащих M (по сравнению с пунктом 6), данное утверждение практически не устраняет эффект экспоненциального нарастания трудоемкости, так как достаточное количество альтернативных применений правил вывода на каждом шаге дает уже исходное множество M.

8) **Стратегия упорядочения литер**. Возможно применение при поиске вывода константы Л другого ограничения на вид линейного вывода. В этом случае разрешающей литерой в каждом D_i является первая литера, то есть D_{i+1} возникает из $D_i = A \vee D'_1 \vee \cdots \vee D'_n$ и $C_i = D''_1 \vee \cdots \vee D''_{j-1} \vee \neg A' \vee D''_{j+1} \vee \cdots \vee D''_n$ как

$$\sigma(D''_1) \vee \cdots \vee \sigma(D''_{j-1}) \vee \sigma(D''_{j+1}) \vee \cdots \vee \sigma(D''_n) \vee \sigma(D'_1) \vee \cdots \vee \sigma(D'_n),$$

причем указанное здесь упорядочение литер в дизъюнкте D_{i+1} сохраняется. Как и в пункте 7, предполагается, что каждое C_i — либо из исходного множества дизъюнктов M, либо совпадает с некоторым D_j, $j < i$, являющимся дизъюнктом слияния. Полнота данной стратегии также была установлена в работах Андерсона и Бледсоу.

Для иллюстрации применения перечисленных выше стратегий рассмотрим сравнительно простой пример доказательства

теорем из теории групп. Будет доказываться утверждение о том, что в ассоциативной системе в которой уравнения вида $xa = b$ и $ay = b$ имеют левое и правое решение x и y, существует правый единичный элемент. при формулировке этого утверждения на языке логики предикатов используем трехместный предикатный символ $P(x, y, z)$, интерпретируемый как равенство $xy = z$. После преобразования к виду дизъюнктов аксиомы существования решений уравнений $xa = b$ и $ay = b$ принимают вид:

D_1 – $P(g(x,y),x,y)$ существование решения для $xa = b$;

D_2 – $P(x,h(x,y),y)$ существование решения для $ay = b$.

Ассоциативность выражается двумя дизъюнктами:

D_3 – $\neg P(x,y,z) \vee \neg P(y,v,w) \vee \neg P(x,w,u) \vee P(z,v,u)$;

D_4 – $\neg P(x,y,z) \vee \neg P(y,v,w) \vee \neg P(z,v,u) \vee P(z,w,u)$.

Наконец, отрицание утверждения о существовании правого единичного элемента приводится к виду:

$$D_5 \quad - \quad \neg P(f(y),y,f(y)).$$

В исходной ситуации несущее множество состоит из единственного элемента D_5, при применении правил вывода будем строить лишь линейные выводы. Стратегия предпочтения одночленов дает лишь две возможности — применение правила R1 к D_5 и D_3, либо к D_5 и D_4. Выберем, например, первую из них, получим дизъюнкт:

$$D_6 \quad - \quad \neg P(x,y,f(z)) \vee \neg P(y,z,v) \vee \neg P(x,v,f(z))$$

(по мере надобности в новых дизъюнктах проводим переобозначение переменных). Теперь несущее множество имеет уже два элемента — D_5 и D_6, и стратегия предпочтения одночленов вовлекает в рассмотрение также дизъюнкты D_1 и D_2. Применяя правило R1 к D_1 и D_6, получаем дизъюнкт:

$$D_7 \quad - \quad \neg P(x,y,z) \vee \neg P(g(x,f(y)),z,f(y))$$

Далее из D_7 и D_1 выводим:

$$D_8 \quad - \quad \neg P(x,y,z)$$

и, наконец, унифицируя D_8 с отрицанием D_2, выводим константу Л.

Разбирая пример, мы привели лишь ведущую к цели цели цепочку вывода. Вместе с тем, здесь имелось уже достаточно большое множество альтернативных выводов, не отсекаемых перечисленными выше стратегиями: применение правила резолюции к парам (D_5, D_4), (D_2, D_6), (D_2, D_7), а также альтернативные применения этого правила к (D_1, D_6) и (D_1, D_7) дают новые дизъюнкты, которые в свою очередь приводят к новым следствиям. Таким образом, в целом сохраняется ветвящийся характер рассматриваемого в описанной процедуре логического вывода, и это ограничивает область эффективной ее применимости классом сравнительно простых задач, где проведение полного перебора является все еще реализуемым.

Попытки применения изложенной выше процедуры автоматического доказательства теорем в различных прикладных задачах, связанных с логическим выводом, привели к созданию языка ПРОЛОГ. Не останавливаясь на технических подробностях, изложенных в руководствах по программированию на этом языке, мы приведем здесь лишь "общелогическую" схему организации и функционирования программ на ПРОЛОГ'е. каждая такая программа представляет собой упорядоченный набор дизъюнктов $(D_1, D_2, \ldots, D_n)$, причем входной информацией для программ также служит некоторый дизъюнкт D_0. Программа пытается найти такую подстановку σ, для которой совокупность дизъюнктов $\{\sigma(D_0), D_1, D_2, \ldots, D_n\}$ становится невыполнимой, причем она не останавливается, найдя первую такую σ, а продолжает поиск и перечисляет все найденные σ вплоть до исчерпания всех возможностей, заложенных в ее схеме перечисления. Сама эта схема исключительно проста — по существу, это полный перебор всех возможных линейных выводов, определяемых дизъюнктами $D_1, \ldots, D_n$, реализуемый по принципу "сначала вглубь". Более подробно, функционирование программы происходит следующим образом. Текущая ситуация описывается при помощи набора троек $S = (S_1, \ldots, S_m)$, где каждое S_i, $i = 1, \ldots, m$, имеет вид (σ_i, D_i^*, k_i), σ_i — подстановка, D_i^* — дизъюнкт, $k_i \in \{1, 2, \ldots, n+1\}$. В исходной ситуации $m = 1$, $\sigma_1 = e$ — тождественная подстановка, $D_1^* = D_0$,

$k_1 = 1$. На очередном шаге преобразований происходит рассмотрение "текущей" тройки $S_m = (\sigma_m, D_m^*, k_m)$, $D_m^* = K_1 \vee \cdots \vee K_s$, и перебор всех дизъюнктов номер которых не меньше k_m. Для каждого D_j, $D_j = Q_1 \vee \cdots \vee Q_p$, предпринимается попытка унификации литер K_1 и $\neg Q_1$. Как только это удалось сделать, определяется резольвента D' дизъюнктов D_m^* и D_j, и к концу набора S присоединяется новая тройка $(\sigma_m\sigma', D', 1)$, где σ' — унифицирующая подстановка. Одновременно индекс k_m в тройке S_m заменяется на $j+1$, и процесс возобновляется. Если просмотр всех дизъюнктов D_j закончился безрезультатно и новая тройка S_{m+1} не возникла, то либо $m = 1$, и тогда программа завершает работу (то есть завершает процесс перечисления результирующих подстановок), либо $m > 1$, и тогда тройка S_m отбрасывается, после чего — переход к следующему шагу. Исключительным является случай, когда дизъюнкт D' оказался константой Л. В этом случае тройка S_{m+1} не вводится, а лишь предпринимается замена k_m на $j+1$ и происходит выдача очередного результата — подстановки $\sigma_m\sigma'$. Затем — переход к очередному шагу.

Дизъюнкт $D_i = Q_1 \vee \cdots \vee Q_p$ ПРОЛОГ-программы можно интерпретировать как последовательность "операторов" $\neg Q_2, \ldots,$ $\neg Q_p$, выполняемых в качестве подпрограммы для реализации условия Q_1. Здесь возникают два принципиально разных свойства, отсутствующих у операторов программирования "обычных" алгоритмических языков. Во-первых, отсутствует четкое разделение программных переменных, встречающихся в операторе, на входные и выходные, и в различных ситуациях в одной и той же программе роли таких переменных могут меняться. Во-вторых, оператор реализуется как бы в режиме перечисления своих "выходных" переменных: сначала находится первая версия набора этих значений, реализуются последующие операторы, и если при их обработке возникает в некоторый момент ложное условие, то происходит возвращение к ранее рассмотренному оператору, который выдает очередную версию набора значений выходных переменных, и так далее.

Такой "режим перечисления" существенно упрощает программирование: исчезают не только затрудняющие отладку операторы "goto", но и становятся крайне редкими в явно выделенные в программе конструкции с циклами, так как эти цик-

лы реализуются автоматически. По-видимому, указанное обстоятельство и предопределило в значительной степени дальнейшее развитие ПРОЛОГ'а — в первую очередь появление в нем большого числа встроенных предикатов, реализуемых интерпретатором (либо компилируемых) без привлечения механизма логического вывода, но с сохранением указанного принципа перечисления выходных значений. В ПРОЛОГ были введены также некоторые дополнительные средства управления ходом выполнения программы, и в конечном итоге программирование на нем значительно приблизилось к программированию на традиционных алгоритмических языках, хотя бы в том смысле, что написанию программы здесь также должна предшествовать разработка алгоритма, учитывающего существенным образом специфику конкретной задачи и включающая в себя оптимизацию применяемой схемы вычислений. Практическая значимость описанной выше процедуры автоматического доказательства теорем как универсального средства решения задач оказалась в результате весьма скромной, будучи явно отодвинутой на второй план предоставляемой ПРОЛОГ'ом возможностью программировать "почти без циклов".

Важный этап в развитии логических методов автоматического решения задач, связанный с попытками обнаружить универсальные, предметно-независимые способы борьбы с перебором, возникающим при поиске решения, привел к пониманию невозможности таких способов и необходимости использования в каждой конкретной предметной области своих алгоритмических конструкций для решения задач, учитывающих статистические особенности как данной предметной области в целом так и рассматриваемых в прикладных ситуациях "потоков задач". По-видимому выработка решающих правил в предметной области является результатом достаточно трудоемкой и сложной в логическом отношении адаптации, и лишь на уровне анализа общих принципов процессов такой адаптации можно надеяться на обнаружение предметно-независимых процедур, используемых интеллектуальными системами.

Упражнения

3.10. Какие из следующих выражений являются правильными выражениями (формулами и термами) логики предикатов:

а) x_1;

б) $(f_1(x_2))$;

в) $x_1 \vee x_2$;

г) $(x_1 \vee x_2)$;

д) $(P_1(x_1) \vee P_2(x_3))\&(P_1(x_1) \vee P_2(x_2))$;

е) $((P_1(x_1) \vee (\neg P_1(x_2)) \vee P_1(x_3))\&(P_1(x_1) \vee P_1(x_2)))$;

ж) $((\forall x_1(P_1(x_1) \vee P_1(x_2)))\&(\forall x_1(\forall x_2((P_2(x_1) \vee P_1(x_2)))))$?

3.11. Индукцией по построению формулы докажите лемму 15 об интерпретациях.

3.12. Указать свободные и связанные вхождения переменных в следующие формулы:

а) $(\forall x_3(\forall x_1(P_1(x_1, x_2) \to P_1(x_3, x_1))))$;

б) $((\forall x_1 P_1(x_1, x_3)) \to (\forall x_3 P_1(x_3, x_1)))$;

в) $((\forall x_2(\exists x_1 P_1(x_1, x_2, f_1(x_1, x_2)))) \vee (\neg(\forall x_1 P_2(x_2, f_2(x_1)))))$.

3.13. Свободен ли терм $f_1(x_1, x_2)$ для x_1 в формулах

а) $(P_1(x_1, x_2) \to (\forall P_2(x_2)))$;

б) $((\forall x_2 P_1(x_2, a_1)) \vee (\exists P_1(x_1, x_2)))$?

3.14. Показать, что следующие формулы не являются общезначимыми:

а) $(((\forall x_1 P_1(x_1)) \to (\forall x_1 P_2(x_1))) \to ((\forall x_1(P_1(x_1) \to P_2(x_1)))))$;

б) $((\forall x_1(P_1(x_1) \vee P_2(x_1))) \to ((\forall x_1 P_1(x_1)) \vee (\forall x_1 P_2(x_1))))$.

3.15. Показать, что если A и B — формулы логики предикатов, то следующие формулы являются общезначимыми:

а) $(A(t) \to (\exists x A(x)))$, если терм t свободен для x в A;

б) $((\forall x A) \to (\exists A))$;

в) $((\forall x_i(\forall x_j A)) \leftrightarrow (\forall x_j(\forall x_i A)))$;

г) $((\forall x A) \leftrightarrow (\neg(\exists x(\neg A))))$;

д) $((\forall x(A \to B)) \to ((\forall x A) \to (\forall x B)))$;

е) $(((\forall x A)\&(\forall x B)) \leftrightarrow (\forall x(A\&B)))$;

ж) $(((\forall x A) \vee (\forall x B)) \to (\forall x(A \vee B)))$;

з) $((\exists x_i(\exists x_j A)) \leftrightarrow (\exists x_j(\exists x_i A)))$;

и) $((\exists x_i(\forall x_j A)) \to (\forall x_j(\exists x_i A)))$.

3.16. Доказать, что всякая формула A со свободными переменными $x_1, \ldots x_n$ выполнима тогда и только тогда, когда выполнима формула $\exists x_1 \ldots \exists x_n A$.

3.17. Доказать, что всякая формула A со свободными переменными $x_1, \ldots x_n$ общезначима тогда и только тогда, когда общезначима формула $\forall x_1 \ldots \forall x_n A$.

Литература

[1] *Алешин С.В.* Распознавание динамических образов. — М.: Изд-во МГУ, 1996.

[2] *Альсведе Р., Вегенер И.* Задачи поиска. — М.: Мир, 1982.

[3] *Андреев А.Е.* Некоторые вопросы тестового распознавания образов // ДАН СССР. — 1980. — Т.255, №4. — С. 781–784.

[4] *Андреев А.Е.* О тупиковых и минимальных тестах // ДАН СССР. — 1981. — Т. 256, №3. — С. 521–524.

[5] *Андреев А.Е.* Об асимптотическом поведении числа тупиковых тестов и минимальной длины теста для почти всех таблиц // Проблемы кибернетики. — 1984. — Т.41. — С. 117–141.

[6] *Ансель Ж.* О числе монотонных булевых функций n переменных. — Кибернетический сборник, новая серия, вып. 5, 1968, С. 53–57.

[7] *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979.

[8] *Вапник В.Н., Червоненкис А.Я.* Теория распознавания образов. — М.: Наука, 1974.

[9] *Васильев В.И.* Распознающие системы. — Киев: Наукова думка, 1983.

[10] *Гасанов Э. Э.* Об одномерной задаче интервального поиска // Дискретная математика. — 1995. — Т. 7, № 2. — С. 40–60.

[11] *Гасанов Э.Э., Кудрявцев В.Б.* Теория хранения и поиска информации. — М.: Физматлит, 2002.

[12] *Гасанов Э.Э.* Теория сложности информационного поиска. — М.: Изд-во МГУ, 2005.

[13] *Гильберт Э.Н.* Теоретико-структурные свойства замыкающих переключательных функций // Кибернетический сборник. — М.: ИЛ, 1960. — Т.1.

[14] *Дмитриев А.Н., Журавлев Ю.И., Кренделев Ф.П.* О математических принципах классификации предметов и явлений // Дискретный анализ. — 1966. — №7.

[15] *Дюкова Е.В.* Об асимптотически оптимальном алгоритме построения тупиковых тестов // ДАН СССР. — 1977. — Т.233, №4. — С. 527–530.

[16] *Дюкова Е.В.* Об асимптотически оптимальном алгоритме построения тупиковых тестов для бинарных таблиц // Проблемы кибернетики. — 1978. — Т. 34. — С. 169–186.

[17] *Ершов А. П.* О программировании арифметических операторов // ДАН СССР. — 1958. — Т. 118. — С. 427–430.

[18] *Ершов А. П.* Операторные алгоритмы. III (Об операторных схемах Янова) // Проблемы кибернетики. — 1968. — Т. 20. — С. 181–200.

[19] *Журавлев Ю.И.* Об алгебраическом подходе к решению задач распознавания или классификации // Проблемы кибернетики. — 1978. — Т. 33. — С. 5–68.

[20] *Кибкало А.А.* Об алгоритмах распознавания образов, использующих короткие тесты // Вестник Московского Университета, Математика, механика. — 1988.

[21] *Ким Д. П.* Методы поиска и преследования подвижных объектов. — М.: Наука, 1989.

[22] *Клейн Ф.* Геометрия. — М,: Наука, 1987.

[23] *Кнут Д.* Искусство программирования для ЭВМ. Т. 3. Сортировка и поиск. — М.: Мир, 1978.

[24] *Козлов В.Н.* Математическое моделирование зрительского восприятия // Математические вопросы кибернетики. — 1996. — Т. 6. — С. 321–338.

[25] *Козлов В.Н.* Элементы математической теории зрительного восприятия. — М.: Изд-во МГУ, 2001.

[26] *Константинов Р.М., Королева З.Е., Кудрявцев В.Б.* Комбинаторно-логический подход к задачам прогноза рудоносности // Проблемы кибернетики. — 1976. — Т. 31. — С. 5–33.

[27] *Коробков В.К.* О монотонных фунциях алгебры логики // Проблемы кибернетики. — 1965. — Т. 13. — С. 5–28.

[28] *Константинов Р.М., Королева З.Е.* Применение тестовых алгоритмов к задачам геологического прогнозирования // Распознавание образов. Тр. Международн. симпозиума 1971 г. по практическим применениям методов распознавания образов. — М.: ВЦ АН СССР, 1973. — С. 194–199.

[29] *Королева З.Е.* О сравнении тестовых алгоритмов распознавания // Журнал выч. матем. и матем. физики. — 1975. — Т. 15, №3. — С. 749–756.

[30] *Коршунов А.Д.* О длине минимальных тестов для прямоугольных таблиц // Кибернетика. — 1971. — №1. — С. 1–11.

[31] *Коршунов А.Д.* О числе монотонных булевых функций // Проблемы кибернетики. — 1981. — Т. 38. — С. 5–109.

[32] *Котов В. Е., Сабельфельд В. К.* Теория схем программ. — М.: Наука, 1991.

[33] *Крушинский Л.В., Козлов В.Н., Кудрявцев В.Б.* О некоторых результатах применения математики к моделированию в биологии. // Математические вопросы кибернетики. — 1988. — Т. 1. — С. 52–88.

[34] *Кудрявцев В.Б., Кудрявцев Вит.Б.* О тестовом подходе к задаче о перспективности населенных пунктов // Исследование операций. — 1972, №3.

[35] *Кудрявцев В.Б.* Теория тестового распознавания // Интеллектуальные системы. — 2006. — Т. 10.

[36] *Ли Д., Препарата Ф.* Вычислительная геометрия. Обзор // Кибернетический сб. — 1987. — Т. 24. — С. 5–96.

[37] *Лупанов О. Б.* О синтезе некоторых классов управляющих систем // Проблемы кибернетики. — 1963. — Т. 10. — С. 63–97.

[38] *Малпас Дж.* Реляционный язык ПРОЛОГ и его применение. — М.: Наука, 1990.

[39] *Мальцев А. И.* Алгоритмы и рекурсивные функции. — М.: Наука, 1986.

[40] *Мартин Дж.* Организация баз данных в вычислительных системах. — М.: Мир, 1980.

[41] *Мейер Д.* Теория реляционных баз данных. — М.: Мир, 1987.

[42] *Мендельсон Э.* Введение в математическую логику. — М.: Наука, 1971.

[43] *Метакидес Г., Нероуд А.* Принципы логики и логического программирования. — М.: Факториал, 1998.

[44] *Носков В.Н., Слепян В.А.* О числе тупиковых тестов для некоторого класса таблиц // Кибернетика. — 1972. — №1. — С. 60–65.

[45] *Носков В.Н.* Диагностические тесты для входов логических устройств // Дискретный анализ. — Новосибирск: ИМ СО АН СССР, 1974. — Т. 26. — С. 72-83.

[46] *Носков В.Н.* О сложности тестов, контролирующих работу входов логических схем // Дискретный анализ. — Новосибирск: ИМ СО АН СССР, 1975. — Т. 27. — С. 23–51.

[47] *Переяславский В.И.* Об одном линейном методе распознавания образов // Комбинаторно-алгебраические методы в прикладной математике (Межвузовский сб.) — 1982.

[48] *Переяславский В.И.* О линейном тестовом алгоритме распознавания // ДАН СССР. — 1983. — Т. 271, №5.

[49] *Погосян Г.Р.* О длине проверяющих тестов для логических схем // ДАН СССР. — 1982. — Т.263, №3.

[50] *Подколзин А.С.* Компьютерный решатель математических задач // ДАН РФ. — 1994. — Т.335, №4.

[51] *Подколзин А.С.* Компьютерное моделирование процессов решения математических задач — М.: Изд-во ЦПИ при мех.-мат. факультете МГУ, 2001.

[52] *Препарата Ф., Шеймос М.* Вычислительная геометрия: Введение. — М.: Мир, 1989.

[53] *Решетников В. Н.* Алгебраическая теория информационного поиска // Программирование. — 1979. — № 3. — С. 68–74.

[54] *Саломаа А.* Криптография с открытым ключом. — М.: Мир, 1996.

[55] *Селезнев О.В., Тальхайм Б.* Ключевые системы в случайных базах данных // Интеллектуальные системы. — 1998. — Т. 3, № 1–2. — С. 307–326.

[56] *Селтон Г.* Автоматическая обработка, хранение и поиск информации. — М.: Советское радио, 1973.

[57] *Ульман Дж.* Основы систем баз данных, пер. с англ. — М.: Мир, 1983.

[58] *Фу К., Гонсалес Р., Ли К.* Робототехника. — М.: Мир, 1989.

[59] *Хант Э.* Искусственный интеллект. — М.: Мир, 1978.

[60] *Хорн Б.К.П.* Зрение роботов. — М.: Мир, 1989.

[61] *Хелльман О.* Введение в теорию оптимального поиска. — М.: Наука, 1985.

[62] *Чегис И.А, Яблонский С.В.* Логические способы контроля электрических схем // Труды матем. ин-та им. В.А. Стеклова, АН СССР. — 1958. — Т. 51. — С. 270–360.

[63] *Яблонский С.В., Чегис И.А.* О тестах для электрических схем // Успехи матем. наук. — 1955. — Т. 10, вып.4(66). — С. 182–184.

[64] *Яблонский С.В., Демидова Н.Г., Константинов Р.М., Королева З.Е., Кудрявцев В.Б., Сиротинская С.В.* Тестовый подход к количественной оценке геолого-структурных факторов и масштабов оруднения (на примере ртутных месторождений) // Геология рудных месторождений. — 1971. — Т. 13, №2. — С. 30–42.

[65] *Atkinson M., Bancilhon F., DeWitt D., Dittrich K., Maier D., Zdonic S.* The Object-Oriented Database System Manifesto // Proc. 1st DOOD, Kyoto 1989.

[66] *Ben-Or M.* Lower bounds for algebraic computation trees // Proc. 15th ACM Annu. Symp. Theory Comput. (Apr. 1983) 80–86.

[67] *Bentley J. L., Friedman J. H.* Data structures for range searching // Comput. Surveys (1979), **11** 397–409.

[68] *Bentley J. L., Maurer H. A.* Efficient worst-case data structures for range searching // Acta Inform. (1980), **13** 155–168.

[69] *Bentley J. L., Saxe J. B.* Decomposable searching problems. I. Static-to-dynamic transformation // J. Algorithms — 1980. — v. 1. — p. 301–358.

[70] *Borodin A. B.* Computational complexity — Theory and practice // Currents in Theory of Computings. Englewood Cliffs, NJ: Printice-Hall, 1973, 35–89.

[71] *Bottenbruch H.* Structure and use of ALGOL 60 // JACM (1962) **9**, 161-221.

[72] CODASYL Systems Committee, Feature Analysis of Generalized Data Base Management Systems, ASM, New York, 1971b.

[73] *Codd E. F.* A Relation Model of Data for Large Shared Data Banks // Comm. ACM 13, № 6, ACM, New York, London, Amsterdam, June 1970, 377–387.

[74] *Codd E. F.* Further Normalization of the Data Base Relational Model // Courant Computer Sci. Symposia (vol. 6: "Data-Base System"), ed. by R. Rustin, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972.

[75] *Data Language/I-System/370 DOS/VS, General Information Manual* GH20-1246, IBM, White Plains, New York, 1974.

[76] *Dedekind R.* Ueber Zrlegungen von Zahlen durch ihre grossten gemeinsamen Teiler. Festschrift Hoch. Braunschweig, 1987 u. ges. Werke, II, 103–148.

[77] *Dumey A.* Indexing for Rapid Random Access Memory Systems // Computers and Automation. (1956) **4**, № 12, 6–9.

[78] *Information Management System Virtual Storage (IMS/VS), General Information Manual* GH20-1260, IBM, White Plains, New York, 1974.

[79] *Lueker G. S., Willard D. E.* A data structure for dynamic range queries // Inform. Processing Lett. (Dec. 1982) **15**, № 5, 209–213.

[80] *Maurer W. D., Lewis T. G.* Hash table methods // Commputing Surveys (1975) **7**, 5–20.

[81] *Minker J. (ed.)* Foundations of deductive databases and logic programming // Morgan Kaufman, Los Altos, 1988.

[82] *Thalheim B.* Entity-Relationship Modeling. Foundations of Database Technology . — Springer, Berlin, 2000.

Учебное пособие

КУДРЯВЦЕВ Валерий Борисович
ГАСАНОВ Эльяр Эльдарович
ПОДКОЛЗИН Александр Сергеевич

ВВЕДЕНИЕ В ТЕОРИЮ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Издательский отдел
Факультета вычислительной математики и кибернетики
МГУ им. М.В.Ломоносова
Лицензия ИД № 05899 от 24.09.2001 г.

119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В.Ломоносова
2-й учебный корпус

Напечатано с готового оригинал-макета
в издательстве ООО "МАКС Пресс"
Лицензия ИД № 00510 от 01.12.99 г.
Подписано к печати 20.09.2006 г.
Формат 60x90 1/16. Усл.печ.л. 13,0. Тираж 500 экз. Заказ 626.
Тел. 939-3890, 939-3891, 928-1042. Тел./Факс 939-3891. 119992, ГСП-2,
Москва, Ленинские горы, МГУ им. М.В.Ломоносова
2-й учебный корпус, 627 к.