Федеральное агентство по образованию
Государственное образовательное учреждение
высшего профессионального образования
«Уральский государственный университет им. А. М. Горького»

ИОНЦ «Информационная безопасность»

Математико-механический факультет

Кафедра алгебры и дискретной математики

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

Математическая логика и теория алгоритмов

Учебное пособие

Автор: доцент кафедры алгебры
и дискретной математики
А. П. Замятин

Екатеринбург
2008

А. П. Замятин

МАТЕМАТИЧЕСКАЯ ЛОГИКА И ТЕОРИЯ АЛГОРИТМОВ

Екатеринбург • 2008

Федеральное агентство по образованию
Уральский государственный университет
им. А.М.Горького

Замятин А.П.

Математическая логика и теория алгоритмов

Учебное пособие

Екатеринбург
2008

Замятин А.П. Математическая логика и теория алгоритмов: Учебное пособие. – Екатеринбург, УрГУ, 2008. – 273 с.

Пособие адресовано студентам специальности “Компьютерная безопасность”. Оно может быть использовано студентами и других специальностей, учебные планы которых предполагают довольно глубокое изучение информатики.

Пособие разбито на семь глав. Оно содержит теоретический материал, подборку задач, а также ответы и указания к ряду задач и решение некоторых из них. В отличие от многих учебников по математической логике и теории алгоритмов, пособие содержит изложение метода резолюций, критерия полноты функций k-значной логики, значительный материал по сложности алгоритмов. В пособии значительное внимание уделено анализу выразительных возможностей языка математической логики, приведены основные результаты теории NP-полноты.

© Уральский государственный университет, 2008
© А.П.Замятин, 2008

Введение

Термином «логика» называется наука, изучающая формы и законы мышления, способы построения доказательств и опровержений различных утверждений. Логика берет начало от работ древнегреческого философа Аристотеля (4 век до нашей эры). Он первым обратил внимание на то, что при выводе одних утверждений из других исходят не из конкретного содержания рассуждений, а из взаимоотношения между их формами. Другой древнегреческий ученый Евклид систематизировал значительное к тому времени количество геометрических утверждений с позиции логики. Он создал аксиоматический метод и положил начало воприятию геометрии как аксиоматической теории, а всей математики как совокупности математических теорий.

Логика Аристотеля усовершенствовалась на протяжении многих веков. Значительный качественный прогресс в развитии логики наступил с применением в логике математических методов. Начало этому положил немецкий философ и математик Г. Лейбниц (17 – 18 век). Он пытался построить универсальный язык, на котором можно было бы формализовать различные рассуждения и все «споры заменить вычислениями».

Возникновение науки, которая называется математической логикой, связывают с работами аглийского математика и логика Д. Буля. Им была создана алгебра логики – результат применения к логике алгебраических методов.

Значительным толчком к новому периоду развития математической логики стало создание неевклидовых геометрий в трудах в трудах русского математика Н. И. Лобачевского и венгерского математика Я. Бойяи (19 век). Обнаруженные к концу 19 века парадоксы в теории множеств поставили перед математической логикой задачу обоснования математики.

Немецкий математик и логик Г. Фреге (19 – 20 век) решение этой задачи видел в сведении математики к логике. Он применил аппарат математической логики для обоснования арифметики. Ему же и американскому математику Ч. Пирсу мы обязаны введением в логику предикатов и кванторов. Сведение математике к логике продолжили в своем трехтомном труде «Осно-

вания математики» американские математики Б. Рассел и А. Уайтхед. Поставленная этими учеными цель в целом достигнута не была, но в результате их деятельности бым создан богатый логический аппарат и математическая логика стала восприниматься как полноценная математическая дисциплина.

Другой немецкий математик Д.Гильберт (20 век) видел путь решения проблем оснований математики в применении аксиоматического метода в более строгой его формулировке. Эта формулировка предполагает не только выделения базовых утверженийматематической теории (аксиом), но и правил комбинирования утверждений (правил вывода). Открытие в тридцатых годах 20 века австрийским математиком К. Геделем неполнимости арифметики показало ограниченность этого пути.

В 20 веке на базе математической логики была разработана теория алгоритмов. В разработку этой теории внесли существенный вклад английский математик А. Тьюринг, американский математик Э. Пост и отечественный математик А. А. Марков.

Математическая логика в течение всего периода развития имела применение как в математике, так и вне ее. Из наиболее значительных применений в математике отметим два. Использование методов математической логики для анализа алгебраических структур привело к возникновению *теории моделей* – области математики, лежащей на стыке алгебры и математической логики. Применение логики в математическом анализе привело к появлению новой научной дисциплины *нестандартный анализ*.

Весьма значительны и “нематематические” применения математической логики в кибернетике и информатике. На одном из применений в информатике остановимся подробнее. В последнее время в рамках этой области знания возникло направление, называемое *искусственный интеллект*, одной из основных задач которого является разработка *моделей представления знаний*. Такая модель должна обладать значительными выразительными средствами – иметь достаточно богатый *язык представления знаний*. Кроме средств описания знаний, модель должна обладать и дедуктивными возможностями – уметь *получать следствия* из некоторой исходной информации. Этим требованиям в полной мере удовлетворяют *логические модели*, в основе которых лежит логика предикатов первого порядка – стержень математической логики.

Пособие предназначено для студентов специальности “Компьютерная безопасность”. Оно может быть использовано при обучении студентов любых специальностей, связанных с обработкой символьной информации.

Пособие состоит из семи глав.

Первая глава посвящена логике высказываний. В ней изучаются основные понятия этой логики, обсуждаются понятия равносильности и логического следствия, приводится список законов логики высказываний. Вводятся нормальные формы и излагается применение к контактным схемам.

Вторая глава посвящена логике предикатов первого порядка. В ней, как и в первой главе, после введения основных понятий, обсуждаются понятия равносильности и логического следствия и приводится список законов логики первого порядка. Вводятся предваренная и сколемовская нормальные формы. На примере транзитивного замыкания демонстрируется способ доказательства невыразимости. Излагается многосортная логика первого порядка, рассматриваются примеры, показывающие значительные выразительные возможности этой логики.

В третьей главе излагается исчисление предикатов. Вводится аксиоматизация гильбертовского типа, доказываются основные теоремы: о дедукции, о непротиворечивости, о полноте исчисления, о компактности. Для исчисления высказываний обсуждается вопрос о независимости схем аксиом. Приведены (для логики высказываний) другие варианты аксиоматизации.

В четвертой главе изучается метод резолюций. Вводятся основные понятия метода для логики высказываний. Приводится теорема о полноте (для логики высказываний). Изучаются подстановка и унификация и основные понятия метода для логики первого порядка. Вводится эрбрановский универсум и семантические деревья. Приводятся теорема Эрбрана и теорема о полноте (для логики первого порядка). Обсуждаются стратегии метода, применения метода для доказательства теорем и решения задачи планирования действий, взаимосвязь метода и логического программирования.

Пятая глава посвящена функциям k-значной логики. Вначале изучаются булевы функции: приводятся понятия замкнутости и полноты класса булевых функций, вводятся основные замкнутые классы функций, приводится теорема Поста и некоторые следствия из нее. Затем рассматривается «общий случай» функций k-значной логики. Доказывается полнота некоторых классов функций, вводится основное для формулировки критерия пол-

ноты понятие сохраняемости функцией данного отношения. Излагается в двух вариантах критерий Розенберга, приводятся примеры применения этого критерия.

В шестой главе вводится формализация понятия «алгоритм» в виде машины Тьюринга. Приводится значительное число разнообразных примеров машин Тьюринга. Излагается понятие универсальной машины Тьюринга, приводятся некоторые алгоритмически неразрешимые проблемы. Изучаются понятия рекурсивно перечислимого и рекурсивного множеств. Излагаются другие варианты формализации понятия «алгоритм»: рекурсивные функции, алгоритмы Маркова.

Седьмая глава посвящена сложности алгоритмов. В начале главы приводится формулировка довольно большого числа практически важных комбинаторных задач, которые затем постоянно используются в примерах. Вводятся понятие временной сложности (детерминированной) машины Тьюринга и класс ***P****-time*. Изучается понятие полиномиальной сводимости и изглагается значительное число результатов полиномиальной сводимости одной задачи к другой. Вводятся понятия недетерминированной машины Тьюринга, временной сложности такой машины и класс ***NP****-time*. Обсуждается проблема ***P*** = ***NP***? Далее вводится понятие *NP*-полноты и доказывается *NP*-полнота некоторых задач. Глава завершается обсуждением приближенных полиномиальных алгоритмов.

Каждая глава содержит довольно значительную подборку задач, которых достаточно для проведения аудиторных занятий и самостоятельной работы. Для большинства задач приведены ответы, ряд задач имеет указания к решению, для некоторых задач приведены решения.

Глава 1. Логика высказываний

Изучение математической логики мы начнем с наиболее простой ее части – логики высказываний.

§1. Высказывания и операции над ними

Высказывание – это повествовательное предложение, о котором можно сказать истинно оно или ложно. Рассмотрим следующие предложения.

A = “Число $\sqrt{2}$ является иррациональным”.

B = “Неверно, что число $\sqrt{2}$ является иррациональным”.

C = “Число $\sqrt{2}+1$ является иррациональным”.

D = “Если число $\sqrt{2}$ является иррациональным, то число $\sqrt{2}+1$ также является иррациональным”.

E = “Число x является иррациональным”.

F = “Который час?”

G = “Идите решать задачу к доске!”

Первые четыре предложения являются высказываниями, последние три – нет. Предложения F и G не являются повествовательными, а значение истинности повествовательного предложения E зависит от того, какие значения получит переменная x (во второй главе подобные предложения будут названы предикатами). Высказывания A, C и D истинны, высказывание B – ложно. Более точно, *значение истинности* высказываний A, C и D *есть истина*, а *значение истинности* высказывания B *есть ложь*. **В дальнейшем истину будем обозначать символом *1*, а ложь – символом *0*.**

Проанализируем высказывания A – D с точки зрения их “внутреннего строения”. Высказывания A и C можно назвать простыми, высказывания B и D – составными, полученными из простых высказываний A и C. Этот пример показывает, что в языке (в данном случае, в русском языке) существуют способы построения одних высказываний из других, которые мы будем называть *операциями*. В естест-

венных языках существует много таких операций. Мы выделим в качестве основных пять операций.

Определение. Пусть X и Y – некоторые высказывания. Тогда высказывание

1) “X и Y” называется *конъюнкцией* высказываний X и Y;

2) “X или Y” называется *дизъюнкцией* высказываний X и Y;

3) “не X” называется *отрицанием* высказывания X;

4) “если X, то Y”называется *импликацией* высказываний X и Y;

5) “X тогда и только тогда, когда Y” называется *эквиваленцией* высказываний X и Y.

Высказывание B из вышеприведенного примера является отрицанием высказывания A, а высказывание D – импликацией высказываний A и C. Введем следующие обозначения для операций: & – конъюнкция, $\vee$ – дизъюнкция, $\neg$ – отрицание, $\rightarrow$ – импликация, $\leftrightarrow$ – эквиваленция. Так, $B = \neg A$, $D = A \rightarrow C$. Символы &, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$ называются *связками*.

Зависимость значения истинности новых высказываний от значений истинности исходных высказываний определяется *таблицей истинности связок* (см. таблицу 1.1). Напомним, что единица означает, что высказывание истинно, а ноль – ложно.

Таблица 1.1

X	Y	$X \& Y$	$X \vee Y$	$\neg X$	$X \rightarrow Y$	$X \leftrightarrow Y$
1	*1*	*1*	*1*	*0*	*1*	*1*
1	*0*	*0*	*1*	*0*	*0*	*0*
0	*1*	*0*	*1*	*1*	*1*	*0*
0	*0*	*0*	*0*	*1*	*1*	*1*

Можно сказать, что таблица 1.1 содержит пять таблиц истинности, по одной для каждой из связок. Эти пять таблиц для удобства объединены в одну.

Прокомментируем таблицы истинности дизъюнкции и импликации. В русском языке союз “или” понимается в двух смыслах: *разделительном* – или то, или другое, но не оба, и *соединительном* – или то, или другое, или оба. Как видно из таблицы 1.1, союз “или” мы будем понимать в соединительном смысле. Перейдем к импликации. Если дана импликация $X \rightarrow Y$, то высказывание X называется *посылкой* импликации, а Y – *заключением*. Если посылка X импликации ложна, то вся импликация $X \rightarrow Y$ истинна (см. третью и

четвертую строки таблицы 1.1). Это свойство импликации часто формулируют в виде следующего принципа: “из ложного утверждения(имеется в виду X) следует все что угодно (имеется в виду Y)”. В силу этого следующее высказывание “если $2\cdot2 = 5$, то π – иррациональное число” является истинным, поскольку оно представляет собой импликацию, посылка которой ложна. Подчеркнем, что при этом не надо искать доказательство или опровержение того, что π – иррациональное число. Аналогично, первая и третья строки таблицы 1.1 показывают нам, что если заключение Y импликации истинно, то вся импликация $X\to Y$ также истинна. Это свойство импликации тоже формулируют в виде принципа: “истинное утверждение (имеется в виду Y) следует из чего угодно (имеется в виду X)”. Из этого принципа сразу следует истинность высказывания “если π – ирррациональное число, то $2\cdot2 = 4$”, поскольку оно представляет собой импликацию с истинным заключением.

§2. Формулы логики высказываний, интерпретация

В первом параграфе высказывания были введены как повествовательные предложения естественного языка, т.е. как лингвистические объекты. Для изучения этих объектов математическими средствами используется понятие формулы логики высказываний. Дадим соответствующие определения.

Определение. *Атомарными формулами логики высказываний* называются выбранные буквы латинского алфавита с индексами и без них.

В качестве выбранных букв мы будем использовать U, V, W, X, Y, Z.

Определение. *Формулами логики высказываний* называются

1) атомарные формулы;

2) символы истины *1* и лжи *0*;

3) выражения вида $(F)\&(G)$, $(F)\vee(G)$, $\neg(F)$, $(F)\to(G)$, $(F)\leftrightarrow(G)$, где F и G – формулы логики высказываний.

На первый взгляд может показаться, что определение содержит “порочный круг”; понятие формулы логики высказываний определяется само через себя. На самом деле, это определение относится к так называемым индуктивным

определениям. Такие определения вводят сначала базовые объекты (в нашем случае – атомарные формулы и символы 0 и 1) и способы порождения новых объектов из уже полученных (в нашем случае – операции, введенные в первом параграфе).

Приведем пример. Буквы X,Y,Z – атомарные формулы. В силу первого пункта определения эти буквы являются формулами логики высказываний, а в силу второго формулами являются выражения $(X)\&(Y)$, $((X)\&(Y)) \to (Z)$. Мы видим, что если строго следовать определению, в формуле надо писать много скобок. Это неудобно для восприятия формулы. Чтобы уменьшить количество скобок условимся, во-первых, атомарные формулы в скобки не заключать, во-вторых, ввести приоритет (силу связывания) для связок. Будем считать, что $\neg$ имеет наивысший приоритет, $\&$ и $\vee$ имеют одинаковый приоритет, который выше, чем $y \to$ и $\leftrightarrow$. Последние две связки имеют одинаковый приоритет. Используя эти соглашения, формулу $((X)\&(Y)) \to (Z)$ можно записать так: $X\&Y \to Z$. Отметим, что поскольку мы не упорядочили $\&$ и $\vee$ по силе связывания, то выражение $X\&Y\vee Z$ не является формулой. Надо в этом выражении поставить скобки, определяющие порядок выполнения операций. Получатся две формулы $(X\&Y) \vee Z$ и $X \,\&\, (Y\vee Z)$.

Аналогичная ситуация возникает и при записи алгебраических выражений. Операция умножения имеет здесь более высокий приоритет, нежели операция сложения, поэтому в выражении $a+(b*c)$ скобки можно не ставить. Есть, однако, и различие. Операции сложения и вычитания при записи алгебраических выражений имеют одинаковый приоритет. Тем не менее, в выражении $a+b-c$ скобки можно не ставить, так как предполагается, что операции в этом случае выполняются слева направо. В логике высказываний мы последнего предположения не делаем, поэтому, как отмечалось, в выражении $X\&Y\vee Z$ надо ставить скобки.

В дальнейшем нам понадобится понятие подформулы. Попросту говоря, подформула формулы F – это “слитная” часть, которая сама является формулой. На строгом уровне это понятие вводится следующим образом.

Определение. *Подформулой* атомарной формулы является она сама. *Подформулами* формулы $\neg F$ являются формула $\neg F$ и все ее подформулы. Подформулами формул $F\&G$,

$F \vee G$, $F \to G$, $F \leftrightarrow G$ являются они сами и все подформулы формул F и G.

Например, формула $F = X\&Y \to X \vee Z$ имеет шесть подформул: X, Y, Z, $X\&Y$, $X \vee Z$, $X\&Y \to X \vee Z$.

Теперь необходимо соотнести понятие высказывания и формулы. На самом простом уровне формула – это *форма* для получения высказываний. Пусть, например, дана формула $F = X\&Y \to Z$. Поставим вместо X, Y и Z соответственно высказывания A_1 = “четырехугольник $ABCD$ является параллелограммом”, A_2 = “в четырехугольнике $ABCD$ смежные стороны равны”, A_3 = “в четырехугольнике $ABCD$ диагонали перпендикулярны”, то получим высказывание A_4 = “если четырехугольник $ABCD$ является параллелограммом и его смежные стороны равны, то диагонали перпендикулярны”. Это высказывание получилось “по форме” F. Если вместо X, Y и Z подставить другие высказывания, то получим новое высказывание, имеющее ту же “форму”.

На строгом уровне сказанное в предыдущем абзаце оформляется в виде понятия интерпретации.

Обозначим через $\boldsymbol{A}$ множество атомарных, а через $\boldsymbol{F}$ – множество всех формул логики высказываний. Зафиксируем некоторую совокупность высказываний P, удовлетворяющую следующим условиям:

1) какие бы два высказывания из P мы ни взяли, P содержит их конъюнкцию, дизъюнкцию, импликацию и эквиваленцию,

2) P содержит отрицание каждого из высказываний, принадлежащих P.

Интерпретацией в широком смысле мы будем называть произвольную функцию

$$\varphi: \boldsymbol{A} \to P.$$

Такая функция, определенная на множестве атомарных формул, естественным образом распространяется на множество всех формул. Выше был приведен пример интерпретации в широком смысле. В этом примере совокупность P содержала высказывания $A_1 - A_4$, а интерпретация φ на атомарных формулах X, Y, Z действовала так: $\varphi(X) = A_1$, $\varphi(Y) = A_2$, $\varphi(Z) = A_3$. Естественное расширение φ на множество всех формул будем обозначать той же буквой. Тогда $\varphi(F) = A_4$.

В дальнейшем от высказываний $\varphi(F)$ нам на самом деле будут нужны только их истинностные значения *1* и *0*. Введем поэтому более узкое понятие интерпретации.

Определение. *Интерпретацией в узком смысле* (или просто *интерпретацией*) называется функция

$$\varphi: \boldsymbol{A} \to \{0, 1\}.$$

Используя таблицы истинности связок, интерпретацию можно расширить на множество всех формул. Приведем пример. Пусть $\varphi(X) = 1$, $\varphi(Y) = 0$, $\varphi(Z) = 1$, $F = X \vee Y \to Z$, $G = X \& Y \leftrightarrow X \& Z$. Тогда $\varphi(F) = 1$, $\varphi(G) = 0$.

В заключение параграфа рассмотрим задачу, решение которой состоит в использовании выразительных возможностей логики высказываний. Рассмотрим следующее рассуждение (которое для последующих ссылок будем называть рассуждением молодого человека). “Если я поеду автобусом, а автобус опоздает, то я пропущу назначенное свидание. Если я пропущу назначенное свидание и начну огорчаться, то мне не следует ехать домой. Если я не получу работу, то я начну огорчаться и мне следует поехать домой. Следовательно, если я поеду автобусом, а автобус опоздает, то я получу эту работу”. Задача состоит в том, чтобы перевести это рассуждение на язык логики высказываний, т.е. представить его в виде последовательности четырех формул (поскольку рассуждение содержит четыре предложения).

Обозначим высказывание “я поеду автобусом” буквой X, “автобус опоздает” – буквой Y, “я пропущу свидание” – Z, “я начну огорчаться” – U, “мне следует ехать домой” – V, “я получу работу” – W. Тогда приведенные в рассуждении предложения можно записать в виде следующих формул: $X \& Y \to Z$, $Z \& U \to \neg V$, $\neg W \to U \& V$, $X \& Y \to W$. Мы перевели рассуждение молодого человека на язык логики высказываний.

§3. Равносильность и законы логики высказываний

Нетрудно привести примеры формул, которые “выражают одно и то же”. Таковы, например, формулы $X \vee Y$ и $Y \vee X$. Подобные формулы мы будем называть *равносильными*. Прежде, чем дать соответствующее определение, условимся о следующем обозначении. Если формула F построена из атомарных формул $X_1, \ldots, X_n$, то F будем обозначать через

$F(X_1, \ldots, X_n)$. Более того, мы будем пользоваться последним обозначением, даже если некоторые из атомарных формул отсутствуют в записи формулы F (но всякая атомарная формула, входящая в F, содержится среди $X_1, \ldots, X_n$).

Определение. Формулы F и G называются *равносильными*, если для любой интерпретации φ выполняется равенство $\varphi(F) = \varphi(G)$.

Убедимся в том, что формулы $F = X \to Y$ и $G = \neg X \vee Y$ равносильны. Из соответствующих определений следует, что если интерпретации φ и ψ совпадают на X и Y, то $\varphi(F) = \psi(F)$ и $\varphi(G) = \psi(G)$. Следовательно, для проверки равенства $\varphi(F) = \varphi(G)$ из определения равносильности надо рассмотреть лишь интерпретации, которые различаются на X и Y (а таких интерпретаций четыре), и вычислить соответствующие значения $\varphi(F)$ и $\varphi(G)$. Другими словами, надо составить совместную таблицу истинности формул F и G (см.таблицу 1.2).

Таблица 1.2

X	Y	$F = X \to Y$	$\neg X$	$G = \neg X \vee Y$
1	*1*	*1*	*0*	*1*
1	*0*	*0*	*0*	*0*
0	*1*	*1*	*1*	*1*
0	*0*	*1*	*1*	*1*

В таблице 1.2 для удобства вычисления значения интерпретаций на G введен промежуточный столбец $\neg X$. Мы видим, что столбцы формул F и G совпадают. Это означает, что формулы F и G равносильны.

Близким к понятию равносильности является понятие тождественной истинности.

Определение. Формула F называется *тождественно истинной,* если для любой интерпретации φ выполняется равенство $\varphi(F) = 1$.

Например, формула $F = X\&Y \to X$ является тождественно истинной. Для проверки равенства $\varphi(F) = 1$ не надо рассматривать все интерпретации, а достаточно рассмотреть лишь четыре, которые различаются на атомарных формулах X и Y. Для таких интерпретаций надо вычислить значение формулы F, т.е. составить ее таблицу истинности (см. таблицу 1.3). Таблица 1.3 для удобства вычисления значения $\varphi(F)$ содержит промежуточный столбец $X\&Y$.

Мы видим, что столбец формулы F состоит из одних единиц. Это означает, что формула F тождественно истинна.

Таблица 1.3

X	Y	$X\&Y$	$F= X\&Y\to X$
1	*1*	*1*	*1*
1	*0*	*0*	*1*
0	*1*	*0*	*1*
0	*0*	*0*	*1*

Теорема 1.1. Формулы F и G равносильны тогда и только тогда, когда формула $F\leftrightarrow G$ является тождественно истинной.

Доказательство. Предположим, что формулы F и G равносильны и рассмотрим интерпретацию φ. Ясно, что $\varphi(F\leftrightarrow G) = \varphi(F) \leftrightarrow \varphi(G)$. Поскольку значения истинности $\varphi(F)$ и $\varphi(G)$ совпадают, по таблице истинности эквиваленции имеем равенство $\varphi(F) \leftrightarrow \varphi(G) = 1$. Это означает, что формула $F\leftrightarrow G$ тождественно истинна.

Предположим теперь, что формула $F\leftrightarrow G$ тождественно истинна и рассмотрим интерпретацию φ. Имеем $1 = \varphi(F\leftrightarrow G) = \varphi(F) \leftrightarrow \varphi(G)$. Но из таблицы истинности эквиваленции следует, что если $\varphi(F) \leftrightarrow \varphi(G) = 1$, то $\varphi(F) = \varphi(G)$. □

В логике высказываний довольно часто приходится проводить преобразования формул, сохраняющие равносильность. Для таких преобразований используются так называемые *законы логики высказываний*. Приведем список этих законов.

Пусть F,G и H – некоторые формулы логики высказываний. Тогда следующие формулы равносильны:

1) $F\&1$ и F;
2) $F\vee 1$ и 1;
3) $F\&0$ и 0;
4) $F\vee 0$ и F;
5) $F\&F$ и F;
6) $F\vee F$ и F;
7) $F\&G$ и $G\&F$;
8) $F\vee G$ и $G\vee F$;
9) $F \& (G\&H)$ и $(F\&G) \& H$;
10) $F \vee (G\vee H)$ и $(F\vee G) \vee H$;
11) $F \& (G\vee H)$ и $(F\&G) \vee (F\&H)$;
12) $F \vee (G\&H)$ и $(F\vee G) \& (F\vee H)$;
13) $F \& (F\vee G)$ и F;
14) $F \vee (F\&G)$ и F
15) $F\&\neg F$ и 0;
16) $F\vee\neg F$ и 1;
17) $\neg(F\&G)$ и $\neg F \vee \neg G$;
18) $\neg(F\vee G)$ и $\neg F \& \neg G$;
19) $\neg\neg F$ и F;
20) $F\to G$ и $\neg F \vee G$;
21) $F\leftrightarrow G$ и $(F\to G) \& (G\to F)$.

Доказательство этих равносильностей легко получается с помощью таблиц истинности. Отметим, что в примере на определение равносильности мы фактически доказали закон 20.

Прокомментируем список законов. Законы 5 и 6 называются *идемпотентностью*, 7 и 8 – *коммутативностью*, 9 и 10 – *ассоциативностью* соответственно конъюнкции и дизъюнкции. Ассоциативность конъюнкции означает, что в конъюнкции трех формул скобки можно ставить как угодно, а следовательно – вообще не ставить. Из этого утверждения следует, что в конъюнкции четырех, пяти и любого конечного числа формул скобки можно ставить как угодно и поэтому вообще не ставить. Аналогичное замечание можно сделать и для дизъюнкции.

Законы 11 и 12 называются *дистрибутивностями*. Более точно, закон 11 – дистрибутивность конъюнкции относительно дизъюнкции, а закон 12 – дистрибутивность дизъюнкции относительно конъюнкции. Для применения этих законов в преобразованиях формул удобно иметь в виду следующий аналог. Заменим в законе 11 формулы F, G и H соответственно буквами a, b и c, знак & заменим умножением $*$, а знак $\vee$ – сложением $+$. Мы получим известное числовое тождество

$$a*(b+c) = a*b+a*c$$

Это тождество есть дистрибутивность умножения чисел относительно сложения. В школе применение этого равенства слева направо называется раскрытием скобок, а справа налево вынесением общего множителя. Отличие операций над высказываниями & и $\vee$ от числовых операций $*$ и $+$ состоит в том, что для высказываний выполняются обе дистрибутивности, а для чисел только одна. Сложение не является дистрибутивным относительно умножения.

Закон 15 называется *законом противоречия,* закон 16 – *законом исключенного третьего,* закон 19 – *снятием двойного отрицания*. Законы 13 и 14 называются *законами поглощения*, а законы 17 и 18 – *законами де Моргана* в честь известного шотландского математика и логика 19 века.

Имея законы логики высказываний, мы, наряду с построением совместной таблицы истинности, получаем еще один способ доказательства равносильности двух формул. Этот способ состоит в переходе от одной формулы к другой с помощью законов. В его основе лежит следующее легко

доказываемое утверждение: если в некоторой формуле F заменить подформулу G равносильной ей формулой G', то получим формулу F', равносильную исходной формуле F.

Проиллюстрируем второй способ на следующем примере: доказать равносильность формул

$$F = [X \,\&\, (Z \to Y)] \vee [(X \to Z) \,\&\, Y] \text{ и}$$
$$G = (X \vee Y) \,\&\, (Y \vee \neg Z).$$

В силу закона 20, формулы $Z \to Y$ и $X \to Z$ равносильны соответственно формулам $\neg Z \vee Y$ и $\neg X \vee Z$, поэтому формула F равносильна формуле

$$F_1 = [X \,\&\, (\neg Z \vee Y)] \vee [(\neg X \vee Z) \,\&\, Y].$$

Дважды применив дистрибутивность (закон 11) и пользуясь ассоциативностью связок $\&$ и $\vee$, получим, что формула F_1 равносильна формуле

$$F_2 = (X \vee \neg X \vee Z) \,\&\, (\neg Z \vee Y \vee \neg X \vee Z) \,\&\, (X \vee Y) \,\&\, (\neg Z \vee Y \vee Y).$$

В силу коммутативности дизъюнкции, законов 16 и 2, формулы $X \vee \neg X \vee Z$ и $\neg Z \vee Y \vee \neg X \vee Z$ равносильны 1. Применив теперь законы 1 и 6 и коммутативность дизъюнкции, получим, что формула F_2 равносильна G.

§4. Логическое следствие

Одна из основных целей изучения логики состоит в получении формального аппарата для доказательства того, является ли данное утверждение следствием других. Введем необходимые понятия.

Определение. Формула G называется *логическим следствием* формул $F_1, F_2, \ldots, F_k$, если для любой интерпретации φ из того, что $\varphi(F_1) = \varphi(F_2) = \ldots = \varphi(F_k) = 1$, следует, что $\varphi(G) = 1$.

В качестве примера рассмотрим следующую задачу: выяснить, логично ли рассуждение молодого человека из §2. Напомним, что это рассуждение мы перевели на язык логики высказываний, т.е. представили в виде последовательности формул: $F_1 = X\&Y \to Z$, $F_2 = Z\&U \to \neg V$, $F_3 = \neg W \to U\&V$, $F_4 = X\&Y \to W$. Это означает, что поставленная задача переходит в следующую задачу: выяснить, будет ли формула F_4 логическим следствием формул F_1, F_2, F_3? Чтобы ответить на этот вопрос, предположим, что при некоторой интерпретации φ формула F_4 принимает значение 0, а формулы F_1, F_2, F_3 – значение 1. Если $\varphi(F_4) = 0$, то $\varphi(X) = \varphi(Y) =$

1 и $\varphi(W) = 0$. Из того, что $\varphi(F_1) = \varphi(F_3) = 1$, следуют равенства $\varphi(Z) = \varphi(U) = \varphi(V) = 1$. Но это противоречит тому, что $\varphi(F_2) = 1$. Полученное противоречие показывает, что если $\varphi(F_1) = \varphi(F_2) = \varphi(F_3) = 1$, то $\varphi(F_4) = 1$, т.е. что формула F_4 является логическим следствием формул F_1, F_2, F_3 и рассуждение молодого человека логично.

Приведем противоположный пример. Докажем, что формула $G = Y \to X$ не является логическим следствием формул $F_1 = X \vee Y$, $F_2 = X \to Y$, $F_3 = Y$. Для этого построим совместную таблицу истинности формул F_1, F_2, F_3 и G.

Таблица 1.4

X	Y	$F_1 = X \vee Y$	$F_2 = X \to Y$	$F_3 = Y$	$G = Y \to X$
1	*1*	*1*	*1*	*1*	*1*
1	*0*	*1*	*0*	*0*	*1*
0	*1*	*1*	*1*	*1*	*0*
0	*0*	*0*	*1*	*0*	*1*

Мы видим (см. таблицу 1.4), что если взять интерпретацию φ, для которой $\varphi(X) = 0$, $\varphi(Y) = 1$, (т.е. взять третью строку таблицы), то $\varphi(F_1) = \varphi(F_2) = \varphi(F_3) = 1$, но $\varphi(G) = 0$. Следовательно, формула G не является логическим следствием формул F_1, F_2, F_3.

Понятие логического следствия тесно связано с понятием выполнимости.

Определение. Множество формул $\{F_1, F_2, \ldots, F_l\}$ называется *выполнимым*, если существует интерпретация φ такая, что $\varphi(F_1) = \varphi(F_2) = \ldots = \varphi(F_l) = 1$.

Проверку выполнимости множества формул $\{F_1, F_2, \ldots, F_l\}$ можно провести построением совместной таблицы истинности этих формул. Если найдется хотя одна строка, в которой в столбцах формул F_1, F_2, …, F_l стоят единицы, то это множество формул выполнимо. Если такой строки нет, то множество формул невыполнимо. Так, множество формул $\{F_1, F_2, F_3, G\}$ из предыдущего примера выполнимо, поскольку в таблице 1.4 в первой строке в столбцах этих формул стоят единицы.

В главе 4 нам понадобится следующее утверждение.

Теорема 1.2 Формула G является логическим следствием формул F_1, F_2, …, F_k тогда и только тогда, когда множество формул $L = \{F_1, F_2, \ldots, F_k, \neg G\}$ невыполнимо.

Доказательство. Пусть формула G является следствием множества формул F_1, ..., F_k . Предположим, что множество L выполнимо. Это означает, что существует интерпретация ψ такая, что $\psi(F_1) = \ldots = \psi(F_k) = \psi(\neg G) = 1$. Но если $\psi(F_1) = \ldots = \psi(F_k) = 1$, то $\psi(G) = 1$, поскольку G – логическое следствие формул F_1, ..., F_k. Полученное противоречие $\psi(\neg G) = 1$ и $\psi(G) = 1$ доказывает, что множество формул $\{F_1, \ldots, F_k, \neg G\}$ невыполнимо.

Пусть теперь множество формул L невыполнимо. Рассмотрим интерпретацию φ такую, что $\varphi(F_1) = \ldots = \varphi(F_k) = 1$. Поскольку L невыполнимо, имеем $\varphi(\neg G) = 0$. Если $\varphi(\neg G) = 0$, то $\varphi(G) = 1$. Следовательно, из равенств $\varphi(F_1) = \ldots = \varphi(F_k) = 1$ следует равенство $\varphi(G) = 1$. Это означает, что G – логическое следствие множества формул F_1, ..., F_k. □

§5. Нормальные формы в логике высказываний

Среди множества формул, равносильных данной, выделяют формулы, имеющие ту или иную нормальную форму. Дадим необходимые определения.

Определение. *Литералом* называется атомарная формула, кроме *1* и *0*, или ее отрицание. *Элементарной конъюнкцией* называется литерал или конъюнкция литералов.

Определение. Формула G имеет *дизъюнктивную нормальную форму* (сокращенно: ДНФ), если она является элементарной конъюнкцией или дизъюнкцией элементарных конъюнкций.

Например, формулы X, $\neg Y$, $X\&\neg Y$, $(X\&\neg Y) \vee (\neg X\&Z)$ имеют ДНФ, а формулы $\neg(X\&Y)$, $X\vee Y\vee 1$, $X\to Y$ не имеют.

Теорема 1.3. Для всякой формулы F существует формула G, равносильная F и имеющая дизъюнктивную нормальную форму.

Теорема легко следует из рассмотрения следующего алгоритма, который по данной формуле F выдает одну из формул G, удовлетворяющую условию теоремы.

Прежде, чем привести алгоритм, условимся не различать формулы, которые получаются одна из другой применением коммутативности и ассоциативности конъюнкции и дизъюнкции, т.е. законов 7–10.

Алгоритм приведения к ДНФ

Шаг 1. Используя законы 21 и 20, исключить из исходной формулы эквиваленцию и импликацию.

Шаг 2. С помощью законов 17–19 занести отрицание к атомарным формулам.

Шаг 3. Если формула содержит подформулу вида

$$H_1 \,\&\, (H_2 \vee H_3),$$

то заменить ее на равносильную формулу

$$(H_1 \& H_2) \vee (H_1 \& H_3).$$

Пример 1. Применение алгоритма проиллюстрируем на примере формулы

$$F = \neg(X \leftrightarrow Y) \,\&\, X.$$

Выполним первый шаг. Для этого, используя закон 21, заменим $X \leftrightarrow Y$ равносильной ей формулой $(X \to Y) \,\&\, (Y \to X)$. Затем в полученной формуле с помощью закона 20 исключим связку $\to$. Мы получим формулу

$$F_1 = \neg[(\neg X \vee Y) \,\&\, (\neg Y \vee X)] \,\&\, X.$$

Перейдем ко второму шагу. Применение закона 17, приведет к формуле

$$F_2 = [\neg(\neg X \vee Y) \vee \neg(\neg Y \vee X)] \,\&\, X.$$

Затем дважды воспользуемся законом 18 и снимем двойное отрицание (закон 19), получим формулу

$$F_3 = [(X \& \neg Y) \vee (Y \& \neg X)] \,\&\, X.$$

Шаг 2 выполнен.

Выполнение шага 3 состоит из применения дистрибутивности к формуле F_3. Это дает нам формулу

$$F_4 = (X \& \neg Y \& X) \vee (Y \& \neg X \& X).$$

Алгоритм на этом завершен. Формула F_4 имеет дизъюнктивную нормальную форму. Но эту формулу можно упростить. Действительно, формула $Y \& \neg X \& X$ в силу законов 15 и 3 равносильна *0*, а формула $X \& \neg Y \& X$ равносильна $X \& \neg Y$ (закон 5). Следовательно, формула F_4 равносильна формуле

$$F_5 = X \& \neg Y.$$

Формула F_5, как и F_4, имеет ДНФ и равносильна исходной формуле F. □

Рассмотренный пример показывает, что формула F может иметь не одну равносильную ей формулу, имеющую ДНФ. Иногда это обстоятельство является неудобным. Чтобы его исключить, вводится более узкое понятие, нежели ДНФ.

Определение. Формула G имеет *совершенную дизъюнктивную нормальную форму* (сокращенно: СДНФ) *относительно атомарных формул* $X_1,\ldots,X_n$, если выполнены следующие условия:

1) $F = F(X_1, \ldots, X_n)$, т.е. в записи формулы участвуют только $X_1, \ldots, X_n$;

2) F имеет дизъюнктивную нормальную форму, т.е. $F = C_1 \vee C_2 \vee \ldots \vee C_k$, где $C_1, \ldots, C_k$ – элементарные конъюнкции;

3) каждая элементарная конъюнкция содержит один и только один из литералов X_i или $\neg X_i$ для любого $i = 1, \ldots, n$;

4) F не содержит одинаковых элементарных конъюнкций.

Например, формулы X, $\neg X \& Y$, $(\neg X \& Y) \vee (X \& \neg Y)$ имеют СДНФ относительно содержащихся в них атомарных формул. Формулы $\neg(X \& Y)$, $(X \& Y) \vee (\neg X \& Z)$, $(X \& Y \& X) \& (\neg X \& \neg Y)$, $(X \& Y) \vee (X \& \neg Y) \vee (Y \& X)$ не имеют СДНФ (относительно содержащихся в них атомарных формул). Для первой формулы не выполняется второе условие, для второй и третьей – третье условие, для четвертой формулы не выполняется последнее условие из определения СДНФ.

Теорема 1.4. Для любой выполнимой формулы F существует равносильная ей формула G, имеющая совершенную дизъюнктивную нормальную форму.

Как и теорема 1.3, эта теорема легко следует из соответствующего алгоритма, который по формуле F выдает формулу G, удовлетворяющую требуемому условию

Алгоритм приведения к СДНФ

Шаги 1–3 – те же, что и в алгоритме приведения к ДНФ.

Шаг 4. Если элементарная конъюнкция C не содержит ни атомарной формулы X_i, ни ее отрицания для некоторого $i = 1, \ldots, n$, то заменить C на две элементарные конъюнкции $(C \& X_i) \vee (C \& \neg X_i)$.

Шаг 5. Если элементарная конъюнкция C содержит два вхождения одного литерала, то одно из них вычеркнуть. Если же C содержит X_i и $\neg X_i$ для некоторого $i = 1,\ldots,n$, то вычеркнуть всю элементарную конъюнкцию.

Шаг 6. Если формула содержит одинаковые элементарные конъюнкции, то вычеркнуть одну из них.

Напомним, что “одинаковость” понимается с точностью до коммутативности и ассоциативности конъюнкции и дизъюнкции.

Пример 2. В качестве примера рассмотрим ту же формулу $F = \neg(X \leftrightarrow Y)\ \&\ X$, что и в примере для предыдущего алгоритма. Как мы видели, выполнение шагов 1 – 3 приводит к формуле F_4. Эта формула имеет ДНФ, но не имеет СДНФ, поскольку для нее не выполняется третье условие. Если для F_4 выполнить шаг 4, то в первой элементарной конъюнкции будет зачеркнуто одно из вхождений литерала X, а вторая эдементарная конъюнкция будет вычеркнута вся. В результате мы получим формулу F_5. Она имеет СДНФ относительно X и Y. □

Пример 3. Рассмотрим еще один пример. Пусть $G = (X\&Y) \vee (X\&\neg Z)$. Эта формула имеет ДНФ, поэтому выполнение алгоритма приведения к СДНФ начинается с шага 4. При выполнении этого шага элементарная конъюнкция $X\&Y$ будет заменена на $(X\&Y\&Z) \vee (X\&Y\&\neg Z)$, а $X\&\neg Z$ – на $(X\&\neg Z\&Y) \vee (X\&\neg Z\&\neg Y)$. В результате получим формулу

$$G_1 = (X\&Y\&Z) \vee (X\&Y\&\neg Z) \vee (X\&\neg Z\&Y) \vee (X\&\neg Z\&\neg Y).$$

Условия шага 5 для формулы G_1 не выполняются, поэтому этот шаг формулы G_1 не изменяет. Формула G_1 содержит одинаковые элементарные конъюнкции – вторую и третью. При выполнении шестого шага будет зачеркнута одна из них и получится формула

$$G_2 = (X\&Y\&Z) \vee (X\&Y\&\neg Z) \vee (X\&\neg Y\&\neg Z).$$

Это и есть формула, равносильная G и имеющая СДНФ относительно входящих в G атомарных формул. □

Ответим на естественно возникающий вопрос о том, зачем в формулировке теоремы 1.4 требуется выполнимость формулы F? Нетрудно доказать, что если формула F невыполнима, т.е. при любой интерпретации φ выполняется равенство $\varphi(F) = 0$, то после приведения F к ДНФ каждая элементарная конъюнкция будет содержать хотя бы одну пару противоположных литералов X и $\neg X$. Но в таком случае на шаге 5 все элементарные конъюнкции будут вычеркнуты.

Имеется еще один способ приведения к СДНФ, основанный на построении таблицы истинности исходной формулы. Изложим этот способ на примере формулы

$$F = X_1\ \&\ (X_2 \rightarrow X_3).$$

Составим таблицу истинности формулы F (таблицу 1.5).

Таблица 1.5

X_1	X_2	X_3	$X_2 \to X_3$	$F = X_1 \& (X_2 \to X_3)$
1	*1*	*1*	*1*	*1*
1	*1*	*0*	*0*	*0*
1	*0*	*1*	*1*	*1*
1	*0*	*0*	*1*	*1*
0	*1*	*1*	*1*	*0*
0	*1*	*0*	*0*	*0*
0	*0*	*1*	*1*	*0*
0	*0*	*0*	*1*	*0*

Выделим строки, в которых в столбце F стоит *1*. (Хотя бы одна такая строка должна быть, так как формула F выполнима.) Это будут первая, третья и четвертая строки. Каждой из выделенных строк поставим в соответствие элементарную конъюнкцию $X_1^{\alpha 1} X_2^{\alpha 2} X_3^{\alpha 3}$, где $X_i^{\alpha i} = X_i$, если в столбце X_i этой строки стоит *1*, и $X_i^{\alpha i} = \neg X_i$, если в столбце X_i этой строки стоит *0*, где $i = 1, 2, 3$. Так, первой строке будет поставлена в соответствие элементарная конъюнкция $X_1 \& X_2 \& X_3$, третьей – $X_1 \& \neg X_2 \& X_3$, четвертой – $X_1 \& \neg X_2 \& \neg X_3$. Формула

$$G = (X_1 \& X_2 \& X_3) \vee (X_1 \& \neg X_2 \& X_3) \vee (X_1 \& \neg X_2 \& \neg X_3)$$

имеет СДНФ относительно X_1, X_2, X_3. В то же время G имеет ту же таблицу истинности, что и F. Это означает, что G равносильна F. Следовательно, G – искомая формула.

Из других нормальных форм рассмотрим конъюнктивную нормальную форму. Она получается из ДНФ заменой & на $\vee$ и $\vee$ на &. Дадим точные определения.

Определение. *Элементарной дизъюнкцией* (или *дизъюнктом*) называется литерал или дизъюнкция литералов.

Определение. Формула G имеет *конъюнктивную нормальную форму* (сокращенно: КНФ), если она является элементарной дизъюнкцией или конъюнкцией элементарных дизъюнкций.

Например, формулы X, $\neg Y$, $X \vee \neg Y$, $X \& \neg Y$, $(X \vee \neg Y) \& (X \vee Z)$ имеют КНФ, а формулы $X \to Y$, $\neg(X \vee Y)$, $(X \& \neg Y) \vee (X \& \neg Z)$ не имеют.

Для конъюнктивных нормальных форм справедливо утверждение, аналогичное теореме 1.3.

Теорема 1.5 Для всякой формулы F существует формула G, равносильная F и имеющая конъюнктивную нормальную форму.

Доказательство теоремы легко следует из анализа алгоритма приведения к КНФ, который в свою очередь получается из алгоритма приведения к ДНФ, если шаг 3 заменить на следующий

Шаг 3′. Если формула содержит подформулу вида

$$H_1 \vee (H_2 \& H_3),$$

то заменить ее на равносильную ей формулу

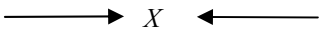
$$(H_1 \vee H_2) \ \& \ (H_1 \vee H_3).$$

В силу очевидной аналогии между ДНФ и КНФ примеров приведения к КНФ здесь приводить не будем.

§6. Контактные схемы

В этом параграфе рассматривается применение логики высказываний к анализу так называемых контактных схем.

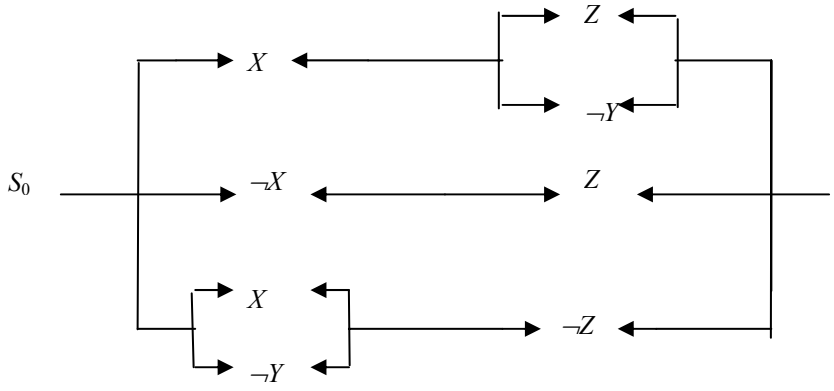
Контактом будем называть устройство, которое в процессе работы может быть в двух состояниях: замкнутом или разомкнутом. Контакт X на чертеже будем изображать следующим образом:



Контакты можно соединять между собой различными способами:

Первое соединение называется *параллельным*, второе – *последовательным*. Контакты, соединенные между собой, будем называть *контактной схемой*. Будем предполагать наличие у схемы двух выделенных точек: входа и выхода. Схему назовем *замкнутой*, если существует последовательность замкнутых контактов X_1, X_2, …, X_n такая, что X_i соединен с X_{i+1}, X_1 соединен с входом, X_n – с выходом. Схему, не являющуюся замкнутой, назовем *разомкнутой*. Каждому контакту поставим в соответствие высказывание, которое истинно тогда и только тогда, когда контакт замкнут. Такие высказывание и контакт будем обозначать одной буквой. Пусть схема S построена из контактов X_1, X_2, …, X_n с помощью параллельного и последовательного соединений. Тогда по схеме S можно построить формулу логики высказываний F_S так, что параллельному соединению соответствует дизъюнкция, последовательному – конъюнкция. На-

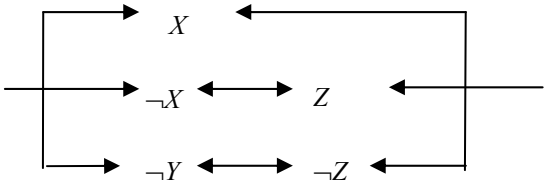
пример, приводимой ниже схеме S_0 соответствует указанная затем формула F_{S0}. (Через $\neg V$ обозначается контакт, который замкнут тогда и только тогда, когда V разомкнут.)



$$F_{S0} = [X \,\&\, (Z \vee \neg Y)] \vee [\neg X \& Z] \vee [(X \vee \neg Y) \,\&\, \neg Z] \,.$$

Формула F_S "представляет" схему в следующем смысле: схема S замкнута в том и только в том случае, если Fs принимает значение *1*. Контактным схемам соответствуют формулы, в построении которых участвуют лишь связки &, $\vee$, $\neg$, причем отрицание применяется только к атомарным формулам. Нетрудно понять, что по всякой такой формуле F можно восстановить схему, которую формула F "представляет".

Пусть схемам S и T соответствуют формулы F_S и F_T в описанном выше смысле. Тогда если схемы S и T эквивалентны (т.е. замкнуты и разомкнуты одновременно), то F_S и F_T равносильны, и обратно. Этот факт используется для решения задачи минимизации контактных схем, которая состоит в том, чтобы по данной схеме S найти схему T, эквивалентную S и содержащую меньше контактов. Один из путей решения этой задачи состоит в переходе к формуле F_S и в отыскании формулы G, равносильной F_S и содержащей меньше вхождений атомарных формул (разумеется, имеется ввиду, что G построена только с помощью &, $\vee$ и $\neg$ причем $\neg$ применяется лишь к атомарным формулам). Так, например, формула F_S равносильна формуле $X \vee (\neg X \& Z) \vee (\neg Y \& \neg Z)$. Следовательно, приведенная выше схема эквивалентна следующей схеме, которая содержит на три контакта меньше.



Заметим, что приведенную контактную схему можно еще упростить.

Задачи

1. Определить, какая логическая связка используется в следующих словесных выражениях: "A, если B", "коль скоро A, то B", "в случае A имеет место B", "как A, так и B", "для A необходимо B", "для A достаточно B", "A вместе с B", "A не имеет места", "A, только если B", "A, пока B", "или A, или B", "A одновременно с B", "A – то же самое, что и B".

2. Записать следующие рассуждения в виде последовательности формул логики высказываний.

2.1. Профсоюзы штата будут поддерживать губернатора, если он подпишет этот закон. Фермеры окажут ему поддержку, если он наложит на него вето. Очевидно, что он или не подпишет закон, или не наложит на него вето. Следовательно, губернатор потеряет голоса рабочих, объединенных в профсоюзы, или голоса фермеров.

2.2. Если мы не будем продолжать политику сохранения цен, то мы потеряем голоса фермеров. Если же мы будем продолжать эту политику и не прибегнем к котролю над производством, то продолжится перепроизводство. Без голосов фермеров нас не переизберут. Значит, если нас переиберут и мы не прибегнем к контролю над производством, то продолжится перепроизводство.

2.3. Если завтра будет хорошая погода, то я буду кататься на коньках или пойду на лыжах. Если я пойду на лыжах, то лучше поехать за город, а если буду кататься на конъках, то останусь в городе. Мне не хочется завтра в выходной день оставаться в городе. Следовательно, если завтра будет хорошая погода, то я пойду на лыжах.

3. Выяснить, будут ли тождественно истинны следующие формулы:

а) $X\&Y \to X$,
б) $X \vee Y \to X$,
в) $X\&Y \to X \vee Y$,
г) $X \vee Y \to X\&Y$,
д) $(X \to Y) \to (Y \to X)$,
е) $(X \to \neg X) \to X$,
ж) $(\neg X \to X) \to X$,
з) $(X \to Y) \to (\neg Y \to \neg X)$,
и) $\neg(X \leftrightarrow Y) \to (\neg X \leftrightarrow \neg Y)$.

4. Выяснить, существует ли формула F такая, что формула G тождественно истинна:

а) $G = X\&Y \to F\&Z$;

б) $G = (F\&Y \to \neg Z) \to (Z \to \neg Y)$;

в) $G = (F\&Z) \vee (\neg F\&\neg Y\&\neg Z)$.

5. Выяснить, будут ли следующие формулы равносильны:

а) $X \to Y$ и $\neg Y \to \neg X$,

б) $\neg X \to Y$ и $\neg Y \to X$

в) $X \to (Y \to Z)$ и $(X \to Y) \to Z$,

г) $X \to (Y \to Z)$ и $X\&Y \to Z$,

д) $\neg(X \to Y)$ и $\neg X \to \neg Y$,

е) $X \leftrightarrow Y$ и $\neg X \leftrightarrow \neg Y$.

6. Доказать равносильность формул:

а) $\neg[(X \vee Y)\ \&\ (X\&\neg Z)]$ и $X \to Z$,

б) $(X\&\neg Y) \vee \neg(X\&Y)$ и $\neg(X\&Y)$,

в) $\neg[(X \vee \neg Y)\ \&\ Y]\ \&\ \neg(\neg X\&Y)$ и $\neg Y$,

г) $\neg[(X\&Y) \vee \neg Z]$ и $\neg\ (Z \to X) \vee \neg(Z \to Y)$,

д) $(X\&Y) \vee (\neg X\&Y) \vee (X\&\neg Y)$ и $X \vee Y$,

е) $(\neg X\&Y\&Z) \vee (\neg X\&\neg Y\&Z) \vee (Y\&Z)$ и $(\neg X \vee Y)\ \&\ Z$

ж) $[(X \to Y)\ \&\ Z] \to \neg(X \to Y)$ и $\neg X \vee \neg Y \vee \neg Z$,

з) $[\neg(\neg X\ \&\ Z)\ \&\ \neg(Y\ \&\ Z)] \vee \neg X \vee \neg Y$ и $\neg(X\ \&\ Y\ \&\ Z)$,

и) $\neg(X \leftrightarrow Y) \vee [(X \to Y)\ \&\ Z]$ и $[\neg X\ \&\ (Y \vee Z)] \vee (X\ \&\ \neg Y) \vee (Y\ \&\ Z)$,

к) $[\neg(X\ \&\ \neg Y)\ \&\ Z] \vee \neg(X \leftrightarrow Y)$ и $(X\ \&\ \neg Y) \vee (\neg X\ \&\ Z) \vee [Y\ \&\ (\neg X \vee Z)]$.

7. Доказать, что формула G является логическим следствием формул $F_1, \ldots, F_n$:

а) $F_1 = X \to Y \vee Z$, $F_2 = Z \to W$, $F_3 = \neg W$, $G = X \to Y$;

б) $F_1 = X \vee Y \vee \neg Z$, $F_2 = X \to X_1$, $F_3 = Y \to Y_1$, $F_4 = Z$, $G = X_1 \vee Y_1$;

в) $F_1 = X \to Y\&Z$, $F_2 = Y \to Z_1 \vee Z_2$, $F_3 = Z \to Z_1$, $F_4 = \neg Z_1$, $G = X \to Z_2$;

г) $F_1 = Z \to Z_1$, $F_2 = Z_1 \to Y$, $F_3 = X \to Y \vee Z$, $G = X \to Y$.

8. Доказать, что формула G не является логическим следствием формул $F_1, F_2, \ldots, F_n$:

а) $F_1 = X \to Y \vee Z$, $F_2 = Y \to W$, $F_3 = Z \to X$, $G = X \to W$;

б) $F_1 = X \to Y$, $F_2 = Y \to Z$, $F_3 = Z \to Z_1 \vee Z_2$, $G = X \to Z_1$;

в) $F_1 = X \vee Y \vee Z$, $F_2 = X \to X_1$, $F_3 = Y \to X_1 \vee Y_1$, $F_4 = \neg Y_1$, $G = Z \to X_1$.

9. Логичны ли рассуждения из задач 2.1, 2.2, 2.3 ?

10. Логичны ли следующие рассуждения ?

10.1. Если Джонс не встречал этой ночью Смита, то Смит был убийцей или Джонс лжет. Если Смит не был убийцей, то Джонс не встречал Смита этой ночью, и убийство произошло после полуночи. Если убийство произошло после полуночи, то Смит был убийцей или Джонс лжет. Эксперты установили, что убийство произошло до полуночи. Следовательно, Смит был убийцей.

10.2. В бюджете возникнет дефицит, если не повысят пошлины. Если в бюджете возникнет дефицит, то расходы на социальные нужды сократятся. Следовательно, если повысят пошлины, то расходы на социальные нужды не сократятся.

10.3. Намеченная атака удастся, если захватить противника врасплох или его позиции плохо защищены. Захватить противника врасплох можно только, если он беспечен. Он не будет беспечен, если его позиции плохо защищены. Следовательно, намеченная атака не удастся.

10.4. Если губернатор не имеет соответствующего авторитета или если он не желает принимать на себя ответственность, то порядок не будет восстановлен и волнения не прекратятся до тех пор, пока участникам волнений это не надоест и власти не начнут примирительные действия. Следовательно, если губернатор не желает взять на себя ответственность и участникам волнений это не надоест, то волнения не прекратятся.

11. Привести формулы к ДНФ:

а) $(X\to Y)\ \&\ (Y\to X)$, б) $\neg[(X\to Y)\ \&\ (Y\to X)]$,

в) $\neg(X\vee Z)\ \&\ (X\to Y)$, г) $\neg(X\&Y\to X)$.

12. Привести формулы к СДНФ:

а) $X\vee(Y\&Z)$, б) $(X\to Y)\ \&\ (Y\to X)$,

в) $\neg(X\vee Y)\ \&\ (X\to Z)$, г) $X\&Y\to\neg\,(X\vee Y)$.

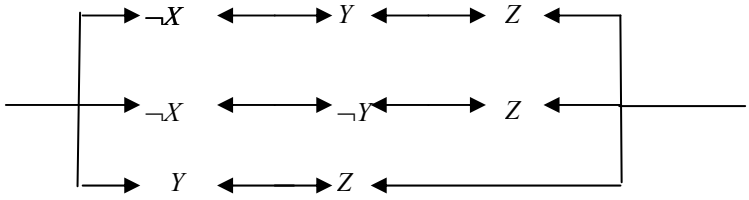
13. Привести формулы к КНФ:

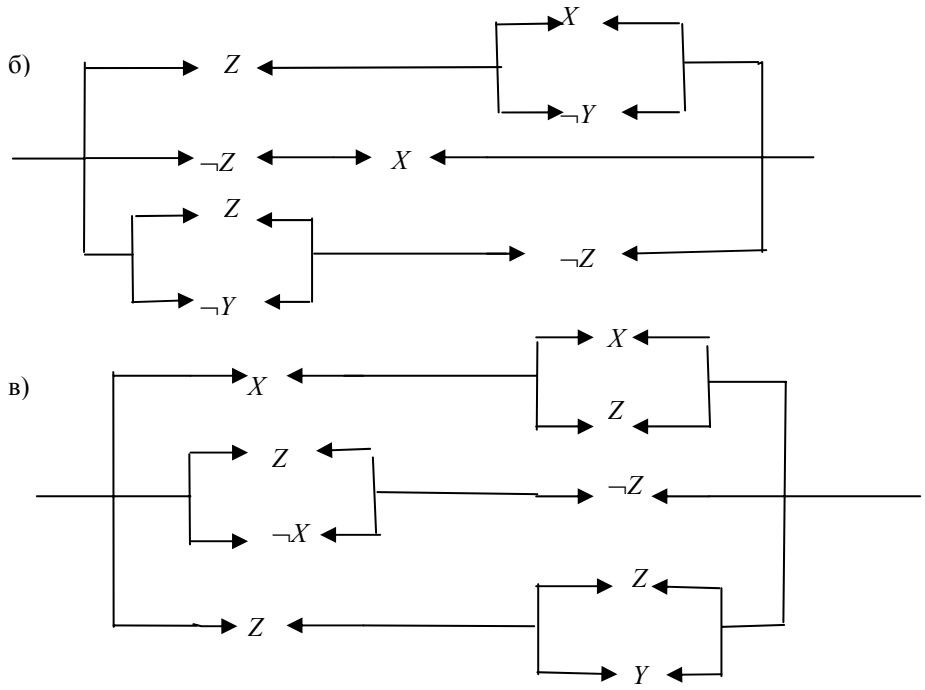
а) $(X\to Y)\ \&\ (Y\to X)$; б) $\neg[(X\to Y)\ \&\ (Y\to X)]$;

в) $X\vee Y\to X\&Y$; г) $\neg(X\&Y\to X\vee Z)$.

14. Для следующих схем построить эквивалентные им более простые схемы:

а)





15. Требуется, чтобы включение света в комнате осуществлялось с помощью трех различных выключателей таким образом, чтобы нажатие на любой из них приводило к включению света, если он был выключен, и выключению, если он был включен. Построить по возможности более простую цепь, удовлетворяющую этому требованию.

16. Пусть каждый из трех членов комитета голосуют "за", нажимая кнопку . Построить по возможности более простую цепь, которая была бы замкнута тогда и только тогда, когда не менее двух членов комитета голосуют "за".

Ответы, указания и решения

2. Приведем решение задачи 2.1. Рассмотрим высказывания: X = ”профсоюзы будут поддерживать губернатора”, Y = ”губернатор подпишет закон”, U = ”фермеры окажут губернатору поддержку”, V = ”губернатор наложит вето”. Тогда рассуждение 2.1 представимо формулами $F_1 = Y \to X$, $F_2 = V \to U$, $F_3 = (\neg Y \& V) \vee (Y \& \neg V)$, $G = \neg X \vee \neg U$.

3. Тождественно истинны формулы а), в), ж) и з). Остальные формулы тождественно истинными не являются.

4. В случаях а) и б) ответ положительный, в качестве F можно соответственно взять $X \& Y$ и Y. В случае в) ответ отрицательный. Действительно, если интерпретация φ такова, что $\varphi(Y) = 1$, $\varphi(Z) = 0$, то $\varphi(G) = 0$ независимо от $\varphi(F)$.

5. В случаях а), б) и г) формулы равносильны, в остальных нет.

6. Приведем решение задачи в случае г). Обозначим формулу $\neg[(X\&Y) \vee \neg Z]$ буквой F, а формулу $\neg(Z\to X) \vee \neg(Z\to Y)$ буквой G. Для доказательства равносильности надо из одной формулы с помощью законов логики высказываний получить другую. Применим к формуле F последовательно законы 18 и 17, получим формулу

$$F_1 = (\neg X\vee\neg Y) \ \& \ Z.$$

Далее, используя дистрибутивность (закон 11), получим формулу

$$F_2 = (\neg X\&Z) \vee (\neg Y\&Z).$$

Осталось отметить, что формула $\neg X\&Z$ равносильна $\neg(Z\to X)$ (законы 19,18 и 20), а формула $\neg Y\&Z$ равносильна $\neg(Z\to Y)$. Следовательно, F равносильна G.

7. Приведем решение задачи 7а). Отметим вначале, что логичность следствия можно доказать, построив совместную таблицу истинности формул F_1, F_2, F_3 и G, и убедиться в том, как только все формулы F_1, F_2, F_3 принимают значение *1*, то формула G принимает то же значение *1*. Однако таблица будет довольно громоздкой, у нее будет 16 строк. Применим другой способ решения задачи. Предположим противное, пусть существует интерпретация φ такая, что $\varphi(F_1) = \varphi(F_2) = \varphi(F_3) = 1$, и $\varphi(G) = 0$. Тогда $\varphi(X) = 1$ и $\varphi(Y) = 0$, поскольку $\varphi(G) = 0$, и $\varphi(W) = 0$, поскольку $\varphi(F_3) = 1$. Далее из равенств $\varphi(F_2) = \varphi(Z\to W) = 1$ и $\varphi(W) = 0$ следует, что $\varphi(Z) = 0$. Но тогда $\varphi(F_1) = \varphi(X \to Y\vee Z) = 0$, что противоречит условию $\varphi(F_1) = 1$. Противоречие показывает что G есть логическое следствие F_1, F_2, и F_3.

8. Для решения задачи необходимо найти интерпретацию, при которой формулы F_1, …, F_n, истинны, а G ложны. Такими интерпретациями являются

а) $\varphi(X) = \varphi(Z) = 1$, $\varphi(Y) = \varphi(W) = 0$;

б) $\varphi(X) = \varphi(Y) = \varphi(Z) = \varphi(Z_2) = 1$, $\varphi(Z_1) = 0$;

в) $\varphi(X) = \varphi(X_1) = \varphi(Y) = \varphi(Y_1) = 0$, $\varphi(Z) = 1$.

9. Рассуждения из задач 2.2 и 2.3 логично, а из задачи 2.1 – нелогично.

10. Приведем решение задачи 10.1. Рассмотрим высказывания: X = ”Джонс не встречал Смита”, Y = ”Смит был убийцей”, Z = ”Джонс лжет”, U = ”убийство произошло после полуночи”. Тогда рассуждения можно представить последовательностью формул: $F_1 = X \to Y\vee Z$, $F_2 = \neg Y \to X\&U$,

$F_3 = U \to Y \vee Z$, $F_4 = \neg U$, $G = X$. Существует интерпретация φ такая, что $\varphi(F_1) = \varphi(F_2) = \varphi(F_3) = \varphi(F_4) = 1$, $\varphi(G) = 0$, пример, $\varphi(X) = \varphi(U) = 0$, $\varphi(Y) = \varphi(Z) = 1$. Это означает, что G не является логическим следствием формул $F_1, \ldots, F_4$ и, следовательно, рассуждение нелогично.

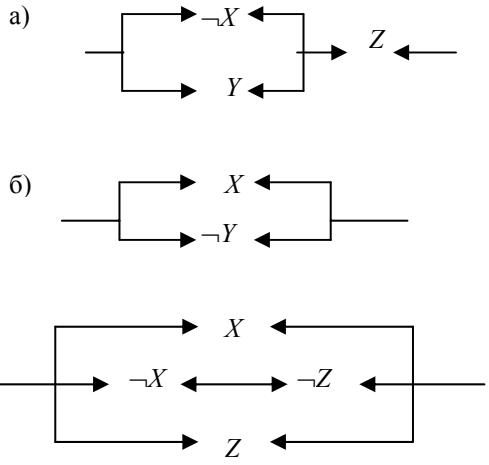
Рассуждения из задач 10.2 – 10.4 также нелогичны.

11. Указание. Применить алгоритм приведения к ДНФ.

12. Указание. Применить алгоритм приведения к СДНФ.

13. Указание. Применить алгоритм приведения к КНФ.

14. Эквивалентные более простые схемы приведены на следующих рисунках:



15. Приведем решение задачи. Рассмотрим формулу $F(X, Y, Z)$, имеющую следующую таблицу истинности (таблицу 1.6). Атомарные формулы X, Y и Z обозначают выключатели. Заметим, что в таблице истинности значения атомарных формул X, Y и Z в каждой следующей строке отличаются от предыдущей только для одной из атомарных формул, а значение формулы F меняется на противоположное. Это отражает требование о том, чтобы при нажатии на любой из выключателей свет выключался, если он был включен, и включался, если он был выключен.

X	*Y*	*Z*	*F*
1	*1*	*1*	*1*
1	*1*	*0*	*0*
1	*0*	*0*	*1*
1	*0*	*1*	*0*
0	*0*	*1*	*1*
0	*1*	*1*	*0*
0	*1*	*0*	*1*

0	0	0	0

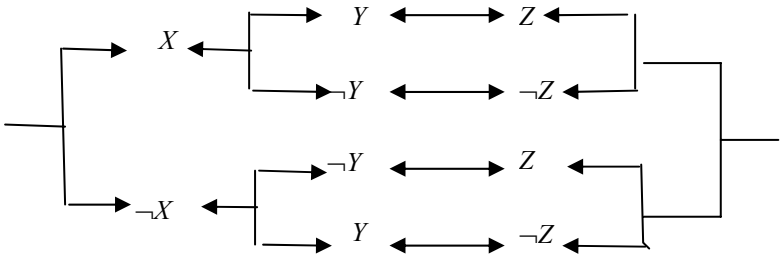
По таблице истинности выпишем формулу F:

$$F = (X\&Y\&Z) \vee (X\&\neg Y\&\neg Z) \vee (\neg X\&\neg Y\&Z) \vee (\neg X\&Y\&\neg Z).$$

Найдем наиболее простую формулу G равносильную F, в запись которой входят лишь связки $\&,\vee$ и $\neg$:

$$G = [X \,\&\, ((Y\&Z) \vee (\neg Y\&\neg Z))] \vee [\neg X \,\&\, ((\neg Y\&Z) \vee \neg(Y\&\neg Z))].$$

По формуле G построим искомую схему.



Глава 2. Логика предикатов первого порядка

Логика высказываний обладает довольно слабыми выразительными возможностями. В ней нельзя выразить даже очень простые с математической точки зрения рассуждения. Рассмотрим, например, следующее умозаключение. “Всякое целое число является рациональным. Число 2 – целое. Следовательно, 2 – рациональное число”. Все эти утверждения с точки зрения логики высказываний являются атомарными. Средствами логики высказываний нельзя вскрыть внутреннюю структуру и поэтому нельзя доказать логичность этого рассуждения в рамках логики высказываний. Мы рассмотрим расширение логики высказываний, которое называется логикой предикатов первого порядка (или логикой первого порядка, или логикой предикатов).

§1. Предикаты и операции над ними

Введем основное понятие данной главы.

Определение. Пусть M – непустое множество. Тогда *n-местным предикатом, заданным на M,* называется выражение, содержащее n переменных и обращающееся в высказывание при замене этих переменных элементами множества M.

Рассмотрим примеры. Пусть M есть множество натуральных чисел $\boldsymbol{N}$. Тогда выражения “x – простое число”, “x – четное число”, “x больше 10” являются одноместными предикатами. При подстановке вместо x натуральных чисел получаются высказывания: “2 – простое число”, “6 – простое число”, “3 – четное число”, “5 больше 10” и т.д. Выражения “x больше y”, “x делит y нацело”, “x плюс y равно 10” являются двухместными предикатами. (Конечно, последнее выражение можно было записать и так: “$x+y=10$”.) Примеры трехместных предикатов, заданных на множестве натуральных чисел: “число z лежит между x и y”, “x плюс y равно z”, “$|x-y| \le z$”.

Будем считать, что *высказывание есть нульместный предикат,* то есть предикат, в котором нет переменных для замены.

Надо отметить, что местность предикатов не всегда равна числу *всех* переменных, содержащихся в выражении. Например, выражение “существует число x такое, что $y = 2x$” на множестве натуральных чисел определяет одноместный предикат. По смыслу этого выражения в нем можно заменять только переменную y. Например, замена y на 6 дает истинное высказывание: “существует число x такое, что $6 = 2x$”, а замена y на 7 дает ложное (на множестве $\boldsymbol{N}$) высказывание: “существует число x такое, что $7 = 2x$”.

Предикат с заменяемыми переменными $x_1, \ldots, x_n$ будет обычно обозначаться заглавной латинской буквой, после которой в скобках указаны эти переменные. Например, $P(x_1, x_2)$, $Q(x_2, x_3)$, $R(x_1)$. Среди переменных в скобках могут быть и фиктивные.

На совокупности всех предикатов, заданных на множестве M, вводятся знакомые операции *конъюнкция, дизъюнкция, отрицание, импликация* и *эквиваленция*. Эти операции вводятся довольно очевидным образом. Приведем в качестве примера определение конъюнкции предикатов.

Определение. Предикат $W(x_1, \ldots, x_n)$ называется *конъюнкцией* предикатов $U(x_1, \ldots, x_n)$ и $V(x_1, \ldots, x_n)$, заданных на множестве M, если для любых $a_1, \ldots, a_n$ из M высказывание $W(a_1, \ldots, a_n)$ есть конъюнкция высказываний $U(a_1, \ldots, a_n)$ и $V(a_1, \ldots, a_n)$.

Легко по аналогии привести определения и других упомянутых выше операций.

В логике предикатов первого порядка вводятся и две новые операции. Называются они навешиванием квантора общности и навешиванием квантора существования. Эти операции рассмотрим вначале на примерах. Пусть дано выражение “существует x такой, что $x+y = 10$”. На множестве натуральных чисел это предложение определяет одноместный предикат $P(y)$, так P(2) и P(9) – истинные высказывания, P(11) – ложное. Если обозначить предикат “$x+y = 10$” через $S(x, y)$ (а это предикат двухместный), то $P(y)$ можно записать так: “существует x такой, что $S(x, y)$”. В этом случае говорят, что предикат $P(y)$ получен из предиката $S(x, y)$ навешиванием квантора существования на x и пишут $P(y) =$

$(\exists x)S(x, y)$. Рассмотрим другой пример. Выражение "для всех x справедливо, что $y \geq -x^2$" определяет на множестве целых чисел одноместный предикат $Q(y)$. Если предикат "$y \geq -x^2$" обозначить через $T(x, y)$, то $Q(y)$ можно записать так: "для всех x справедливо $T(x, y)$". В таком случае говорят, что предикат $Q(y)$ получен из предиката $T(x, y)$ навешиванием квантора общности на x и пишут $Q(y) = (\forall x)T(x, y)$.

После этих примеров нетрудно дать определение в общем виде.

Определение. Пусть $P(x_1, \ldots, x_n)$ – предикат, заданный на множестве M, y – переменная. Тогда выражение "для всякого y выполняется $P(x_1, \ldots, x_n)$" – предикат, полученный из P *навешиванием квантора общности на переменную* y, а выражение "существует y такой, что выполняется $P(x_1, \ldots, x_n)$" – предикат, полученный из P *навешиванием квантора существования на переменную* y.

Обозначения этих операций были введены выше.

Заметим, что в определении не требуется, чтобы переменная y была одной из переменных $x_1, \ldots, x_n$, хотя в содержательных примерах, которые будут приведены ниже, квантор будет навешиваться на одну из переменных $x_1, \ldots, x_n$. Указанное требование не налагается в определении, чтобы избежать усложнения определения формулы логики предикатов. Если y – одна из переменных $x_1, \ldots, x_n$, то после навешивания квантора на переменную y новый предикат является $(n-1)$-местным, если же $y \notin \{x_1, \ldots, x_n\}$, то местность нового предиката равна n.

Если предикат $W(x_1, \ldots, x_n)$ получен из предикатов $U(x_1, \ldots, x_n)$ и $V(x_1, \ldots, x_n)$ с помощью связок, то истинность высказывания $W(a_1, \ldots, a_n)$ определяется таблицами истинности этих связок. Пусть $W(x_1, \ldots, x_n) = (\forall y)U(x_1, \ldots, x_n, y)$. Тогда высказывание $W(a_1, \ldots, a_n)$ истинно тогда и только тогда, когда для любого $b \in M$ истинно высказывание $U(a_1, \ldots, a_n, b)$. Если же $W(x_1, \ldots, x_n) = (\exists y)U(x_1, \ldots, x_n, y)$, то высказывание $W(a_1, \ldots, a_n)$ истинно в том и только в том случае, когда найдется $b \in M$, для которого высказывание $U(a_1, \ldots, a_n, b)$ истинно.

Понятие предиката – весьма широкое понятие. Это видно уже из приведенных выше примеров. Расширим запас таких примеров, показав, что n-местная функция может

рассматриваться как (n+1)-местный предикат. Действительно, функции $y = f(x_1, \ldots, x_n)$, заданной на множестве M, можно поставить в соответствие выражение "y равно $f(x_1, \ldots, x_n)$". Это выражение есть некоторый предикат $P(x_1, \ldots, x_n, y)$. При этом, если элемент b есть значение функции в точке $(a_1, \ldots, a_n)$, то высказывание $P(a_1, \ldots, a_n, b)$ истинно, и обратно. (Подобное "превращение" функции в предикат мы уже делали выше для сложения натуральных чисел.)

На предикаты можно смотреть и более формально, причем с двух точек зрения.

Во-первых, предикат можно представить отношением следующим образом. Пусть предикат $P(x_1, \ldots, x_n)$ задан на множестве M. Рассмотрим прямую степень этого множества $M^n = M \times M \times \ldots \times M$ и подмножество D_p множества M^n, определяемое равенством:

$D_p = \{(a_1, \ldots, a_n) \in M^n \mid$ высказывание $P(a_1, \ldots, a_n)$ истинно$\}$.

Отношение D_p можно назвать областью истинности предиката P. Во многих случаях предикат P можно отождествить с отношением D_p. При этом, правда возникают некоторые трудности при определении операций над отношениями, аналогичных операциям над предикатами.

Во-вторых, предикат $P(x_1, \ldots, x_n)$, заданный на M, можно отождествить с функцией f_p: $M^n \to \{0, 1\}$, определяемой равенством

$$f_p(a_1, \ldots, a_n) = \begin{cases} 1, \text{ если } P(a_1, \ldots, a_n) \text{ истинно,} \\ 0 \text{ в противном случае.} \end{cases}$$

Мы, в основном, будем понимать термин "предикат" в смысле исходного определения, т.е. как языковое выражение. Связано это с тем, что одной из главных целей, как уже отмечалось во введении, является изучение выразительных возможностей логики первого порядка, возможности представления средствами этой логики информации, выраженной на естественном (скажем, русском) языке.

§2. Формулы логики первого порядка

Целью параграфа является введение понятия, вынесенного в его заголовок. В принципе это делается так же, как и в логике высказываний, т.е. сначала вводится понятие атомарной формулы, а затем формулы. Только с определением атомарной формулы в случае логики первого порядка ситуация несколько сложнее.

Будем исходить из следующих трех множеств: $\boldsymbol{F}$, $\boldsymbol{R}$, $\boldsymbol{V}$. Элементы множества $\boldsymbol{F}$ – символы (или имена) функций, элементы $\boldsymbol{R}$ – символы (или имена) предикатов, элементы множества $\boldsymbol{V}$ – переменные. Будем считать, что каждому символу функции и предиката поставлено в соответствие натуральное число или ноль – местность (т.е. число аргументов) этого символа. Допускаются нульместные символы функций, которые называются *константами*, и нульместные символы предикатов (последние будут играть роль атомарных формул логики высказываний). Объединение $\boldsymbol{F}$ и $\boldsymbol{R}$ будем называть *сигнатурой*. Сигнатуру заранеее фиксировать не будем, она будет определяться содержанием решаемой задачи. Множество $\boldsymbol{V}$ предполагается бесконечным, для обозначения его элементов будут использоваться, как правило, буквы x, y, z, u, y, w с индексами и без них.

В примерах, приведенных выше, мы видели, что для записи предикатов использовались арифметические выражения: $2x$, $x+y$, $-x^2$. Аналогом арифметического выражения в логике служит понятие терма.

Определение. *Термами* называются

1) переменные и константы;

2) выражения вида $f(t_1, \ldots, t_n)$, где $t_1, \ldots, t_n$ – термы, а f – n-местный функциональный символ.

Можно сказать, что терм – выражение, полученное из переменных и констант с помощью символов функций.

Определение. *Атомарной формулой* называется выражение вида $r(t_1, \ldots, t_n)$, где $t_1, \ldots, t_n$ – термы, r – символ n-местного предиката.

Примерами атомарных формул являются выражения $x \le y^2+1$, $|x-y| < z$, x делит нацело $y-3$.

Определение. *Формулами логики первого порядка* называются

1) атомарные формулы,

2) символы истины 1 и лжи 0,

3) выражения вида $(F)\&(G)$, $(F_1)\vee(G)$, $\neg(F)$, $(F)\to(G)$, $(F)\leftrightarrow(G)$, $(\forall v)(F)$, $(\exists v)(F)$, где F и G – формулы логики предикатов, v – переменная.

Формула F в двух последних выражениях называется *областью действия квантора по переменной* v.

К соглашениям о приоритете операций, принятом в логике высказываний, добавим следующее: кванторы имеют одинаковый приоритет, который больше приоритета всех связок.

Определение. Вхождение переменной в формулу называется *связанным*, если переменная стоит непосредственно за квантором или входит в область действия квантора по этой переменной. В противном случае вхождение называется *свободным*.

Например, в формуле

$$F = t(x) \,\&\, (\forall y)[s(x, y) \to (\exists x)(r(x, y) \vee t(y))]$$

Первое и второе вхождение переменной x свободны, третье и четвертое связаны. Все вхождения переменной y связаны.

Определение. Переменная называется *свободной в формуле,* если существует хотя бы одно свободное вхождение переменной в формулу.

Формула F из предыдущего примера имеет одну свободную переменную x.

Если $x_1, \ldots, x_n$ – все свободные переменные формулы F, то эту формулу будем обозначать через $F(x_1, \ldots, x_n)$. Это обозначение будем применять и в том случае, когда не все переменные из $x_1, \ldots, x_n$ являются свободными в F.

Как и в логике высказываний, в логике первого порядка вводится понятие подформулы. Соответствующее определение получится из определения подформулы из §2 темы 1 добавлением фразы: “Подформулами формул $(\forall v)(F)$ и $(\exists v)(F)$ являются они сами и все подформулы формулы F”.

§3. Интерпретация в логике первого порядка

Необходимо соотнести формулы логики предикатов первого порядка и предикаты. Как и в логике высказываний, подобное соотнесение осуществляет функция, называемая интерпретацией.

Определение. *Интерпретацией* на непустом множестве M называется функция, заданная на сигнатуре $\boldsymbol{F} \cup \boldsymbol{R}$, которая

1) константе ставит в соответствие элемент из M;

2) символу n-местной функции ставит в соответствие некоторую n-местную функцию, определенную на множестве M;

3) символу n-местного предиката ставит в соответствие n-местный предикат, заданный на M.

В результате любая формула F получает в соответствие предикат, местность которого равна числу свободных переменных формулы F.

Пример 1. Пусть сигнатура состоит из символов одноместного предиката P и двухместного предиката D, $M = \{2, 3, 6, 9, 12, 15\}$ и

$$F = P(x) \ \& \ (\forall y)(P(y) \to D(x, y)).$$

Поставим в соответствие символу P предикат "x – простое число", символу D – предикат "x меньше или равно y". Тогда формула F получит в соответствие предикат "$x = 2$". На этом же множестве можно рассмотреть и другую интерпретацию: P ставится в соответствие "x – нечетное число", D – предикат "x делит y". В таком случае, формула F получает в соответствие предикат "$x = 3$". Если φ – интерпретация, то предикат, соответствующий формуле F, будем обозначать через φF. □

Одним из основных типов задач этой главы являются задачи, связанные с использованием выразительных возможностей языка логики предикатов. В качестве примера рассмотрим задачу перевода на язык логики предикатов следующего рассуждения. "Каждый первокурсник знаком с кем-либо из спортсменов. Никакой первокурсник не знаком ни с одном любителем подледного лова. Следовательно, никто из спортсменов не является любителем подледного лова". Для удобства ссылок это рассуждение условимся называть рассуждением о первокурсниках. Выберем следующую сигнатуру (сигнатурные элементы приведены с предполагаемой интерпретацией).

$П(x)$: "x – первокурсник",

$C(x)$: "x – спортсмен",

$Л(x)$: "x – любитель подледного лова",

$З(x, y)$: "x знаком с y".

Тогда рассуждение запишется в виде следующей последовательности формул:

$H_1 = (\forall x)[П(\text{x}) \to (\exists y)(C(y) \ \& \ З(x, y))]$,

$H_2 = (\forall x)(\forall y)[П(x) \& Л(y) \to \neg З(x, y)]$,

$H_3 = (\forall x)(C(x) \to \neg Л(x))$.

Мы установили, что выразительных средств логики предикатов достаточно, чтобы записать рассуждение о первокурсниках. Естественно далее поставить вопрос, логично ли оно? Будет ли третье предложение следствием первых двух? На этот вопрос мы ответим в §5.

§4. Равносильность, законы логики первого порядка

Общая схема изложения материала этого и двух следующих параграфов будет напоминать изложение материала в §3 – 5 главы 1.

Определение. Формулы $F(x_1, \ldots, x_n)$ и $G(x_1, \ldots, x_n)$ называются *равносильными*, если для любой интерпретации φ с областью M высказывания $(\varphi F)(a_1, \ldots, a_n)$ и $(\varphi G)(a_1, \ldots, a_n)$ одновременно истинны или одновременно ложны при любых $a_1, \ldots, a_n$ из M.

Пример 1. Пусть P – символ двухместного предиката. Докажем□, что формулы $F(x) = \neg(\forall y)P(x, y)$ и $G(x) = (\exists y)\neg P(x, y)$ равносильны. Возьмем интерпретацию φ с областью M. Пусть высказывание $(\varphi F)(a)$ истинно для $a \in M$. Истинность этого высказывания одначает, что не для всякого $y \in M$ высказывание $(\varphi P)(a, y)$ истинно. Следовательно, найдется $b \in M$, для которого высказывание $(\varphi P)(a, b)$ ложно. Если высказывание $(\varphi P)(a, b)$ ложно, то высказывание $\neg(\varphi P)(a, b)$ истинно. Отсюда следует, что найдется $y \in M$ такой, для которого высказывание $\neg(\varphi P)(a, y)$ истинно. Это означает истинность высказывания $(\varphi G)(a)$. Мы доказали, что если высказывание $(\varphi F)(a)$ истинно, то высказывание $(\varphi G)(a)$ тоже истинно. Обратная импликация доказывается аналогично. Итак, значения истинности высказываний $(\varphi F)(a)$ и $(\varphi G)(a)$ при любом $a \in M$ совпадают. Следовательно, формулы $F(x)$ и $G(x)$ равносильны. □

Определение. Формула $F(x_1, \ldots, x_n)$ называется *тождественно истинной*, если для любой интерпретации φ с областью M высказывание $(\varphi F)(a_1, \ldots, a_n)$ при любых $a_1, \ldots, a_n$ из M является истинным.

Пример 2. Рассмотрим формулу $F(x) = (\forall y)P(x, y) \to P(x, c)$, где P – символ двухместного отношения, с – константа.

Докажем, что формула $F(x)$ тождественно истинна. Возьмем интерпретацию φ с областью M и элемент a из M. Высказывание $(\varphi F)(a)$ равно $(\forall y)(\varphi P)(a, y) \to (\varphi P)(a, \varphi(c))$. Если посылка $(\forall y)(\varphi P)(a, y)$ ложна, то вся импликация $(\varphi F)(a)$ истинна. Предположим, что посылка $(\forall y)(\varphi P)(a, y)$ истинна. Это означает, что для всякого $y \in M$ высказывание $(\varphi P)(a, y)$ истинно, в том числе оно истинно и для $y = \varphi(c)$. Следовательно, истинны заключение $(\varphi P)(a, \varphi(c))$ и вся импликация $(\varphi F)(a)$. Мы доказали, что высказывание $(\varphi F)(a)$ истинно для любого $a \in M$. Это означает, что формула $F(x)$ является тождественно истинной. □

Понятия равносильности и тождественной истинности в логике первого порядка связаны точно так же, как и в логике высказываний.

Теорема 2.1. Формулы $F(x_1, \ldots, x_n)$ и $G(x_1, \ldots, x_n)$ равносильны тогда и только тогда, когда формула

$$F(x_1, \ldots, x_n) \leftrightarrow G(x_1, \ldots, x_n)$$

тождественно истинна.

Доказательство теоремы 2.1 аналогично доказательству теоремы 1.1 и предлагается читателю в качестве упражнения.

Как и в случае логики высказываний, приведем список основных равносильностей – законов логики предикатов. Прежде всего, получим законы логики предикатов из законов 1 – 21 логики высказываний, понимая под F, G, H произвольные формулы логики предикатов. Дополним полученный список законами, специфичными для логики предикатов.

22) $(\forall x)(F(x) \& G(x))$ равносильна $(\forall x)F(x)$ & $(\forall x)G(x)$,
23) $(\exists x)(F(x) \vee G(x))$ равносильна $(\exists x)F(x) \vee (\exists x)G(x)$,
24) $(\forall x)(\forall y)F(x, y)$ равносильна $(\forall y)(\forall x)F(x, y)$,
25) $(\exists x)(\exists y)F(x, y)$ равносильна $(\exists y)(\exists x)F(x, y)$,
26) $\neg(\forall x)F(x)$ равносильна $(\exists x)\neg F(x)$,
27) $\neg(\exists x)F(x)$ равносильна $(\forall x)\neg F(x)$.

Законы 22, 23 утверждают, что квантор общности можно переносить через конъюнкцию, а квантор существования через – дизъюнкцию. Естественно поставить вопрос, можно ли квантор существования переносить через конъюнкцию, а квантор общности – через дизъюнкцию. Другими словами, будут ли равносильны следующие пары формул

$(\forall x)(F(x)\vee G(x))$ и $(\forall x)F(x) \vee (\forall x)G(x)$,

$(\exists x)(F(x)\&G(x))$ и $(\exists x)F(x) \,\&\, (\exists x)G(x)$?

Оказывается, нет. Докажем это для случая, когда $F(x)$ и $G(x)$ – атомарные формулы. Пусть основное множество – множество натуральных чисел $\boldsymbol{N}$, $F(x)$ – предикат "x – четное число", $G(x)$ – предикат "x – нечетное число". Обозначим эту интерпретацию буквой φ. Тогда $\varphi[(\forall x)(F(x)\vee G(x))] = 1$, но $\varphi[(\forall x)F(x) \vee (\forall x)G(x)] = 0$. Аналогично, $\varphi[(\exists x)(F(x)\&G(x))] = 0$ и $\varphi[(\exists x)F(x) \,\&\, (\exists x)G(x)] = 1$.

Рассмотрим законы 23 и 24. Они утверждают, что одноименные кванторы можно менять местами. Можно ли переставлять местами разноименные кванторы, сохраняя равносильность? Другими словами, равносильны ли формулы

$(\forall x)(\exists y)F(x, y)$ и $(\exists y)(\forall x)F(x, y)$?

Оказывается, тоже нет. В качестве основного множества опять возьмем множество натуральных чисел, $F(x, y)$ будем считать атомарной формулой и поставим ей в соответствие предикат "x меньше y ". Тогда левая формула будет истинной, правая – ложной.

Вернемся к законам 22 и 23. Мы отмечали, что квантор общности нельзя переносить через дизъюнкцию, а квантор существования – через конъюнкцию. Тем не менее, если одна из формул F или G не содержит переменной x (на которую навешивается квантор), то это делать можно. Запишем соответствующие законы:

28) $(\forall x)(F(x)\vee G)$ равносильна $(\forall x)(F(x) \vee G$,

29) $(\exists x)(F(x)\&G)$ равносильна $(\exists x)(F(x) \,\&\, G$, где G не содержит x.

Законы 22, 23, 28, 29 можно записать в общем виде:

30) $(Q_1x)(Q_2u)(F(x)\vee G(u))$ равносильна $(Q_1x)F(x) \vee (Q_2u)G(u)$,

31) $(Q_1x)(Q_2u)(F(x)\&G(u))$ равносильна $(Q_1x)F(x) \,\&\, (Q_2u)G(u)$,

где Q_1, Q_2 – кванторы $\forall$ или $\exists$, переменная u не входит в $F(x)$, а переменная x не входит в $G(u)$.

Для доказательства равносильности двух формул могут оказаться полезными следующие законы:

32) $(\forall x)F(x)$ равносильна $(\forall z)F(z)$,

33) $(\exists x)F(x)$ равносильна $(\exists z)F(z)$.

В законах 32 и 33 переменная z не входит в $F(x)$, а переменная x не входит в $F(z)$.

В логике высказываний мы применяли два способа доказательства равносильности формул: построение совместной таблицы истинности и переход от одной формулы к другой с помощью законов. В случае логики первого порядка остается только второй способ.

Пример 3. Проиллюстрируем его на примере следующей задачи: доказать равносильность формул:

$$F = \neg(\forall x)(\exists y)[S(x) \,\&\, P(x, y) \to (\exists z)(T(z) \,\&\, P(x, z))],$$

$$G = (\exists x)(\forall y)[S(x) \,\&\, P(x, y) \,\&\, (\forall z)(T(z) \to \neg P(x, z))].$$

Применив к формуле F последовательно законы 26, 27 и 20, получим, что формула F равносильна формуле

$$F_1 = (\exists x)(\forall y)\neg[\neg(S(x) \,\&\, P(x, y)) \vee (\exists z)(T(z) \,\&\, P(x, z))].$$

Далее, используя законы 18, 19 и 27 из F_1, получаем формулу

$$F_2 = (\exists x)(\forall y)[S(x) \,\&\, P(x, y) \,\&\, (\forall z)\neg(T(z) \,\&\, P(x, z))].$$

Осталось заметить, что в силу законов 17 и 20 в формуле F_2 подформулу $\neg(T(z) \,\&\, P(x, z))$ можно заменить на $T(z) \to \neg P(x, z)$. □

Подчеркнем, что доказательство равносильности двух формул обычно проводится с помощью законов логики первого порядка. Доказательство того, что формулы неравносильны осуществляется построением интерпретации, при которой одна формула истинна, другая ложна. Например так, как это было сделано выше для доказательства неравносильности формул $(\forall x)(F(x) \vee G(x))$ и $(\forall x)F(x) \vee (\forall x)G(x)$. Разумеется, формулы до построения интерпретации можно предварительно преобразовывать с помощью законов.

§6. Логическое следствие

Определение. Формула $G(x_1, \ldots, x_n)$ называется *логическим следствием* формул $F_1(x_1, \ldots, x_n), \ldots, F_k(x_1, \ldots, x_n)$, если для любой интерпретации φ с областью M и любых $a_1, \ldots, a_n \in M$ из истинности высказываний $(\varphi F_1)(a_1, \ldots, a_n), \ldots, (\varphi F_k)(a_1, \ldots, a_n)$ следует истинность высказывания $(\varphi G)(a_1, \ldots, a_n)$.

Пример 1. Пусть $F_1 = (\forall x)(P(x) \to Q(x) \& R(x))$, $F_2 = P(c)$, $G = Q(c)$. Покажем, что формула G является логическим

следствием формул F_1 и F_2. Возьмем интерпретацию φ с областью M такую, что высказывания φF_1 и φF_2 истинны. Элемент $\varphi(c)$ обозначим буквой b. Истинность φF_2 означает, что высказывание $(\varphi P)(b)$ истинно. А истинность высказывания φF_1 означает, что для любого элемента $x \in M$ истинно высказывание $(\varphi P)(x) \to (\varphi Q)(x) \& (\varphi R)(x)$. Поскольку это высказывание истинно для любого x, оно, в частности, истинно для $x = b$. Мы видим, что истинна импликация $(\varphi P)(b) \to (\varphi Q)(b) \& (\varphi R)(b)$ и ее посылка $(\varphi P)(b)$. Из таблицы истинности импликации следует истинность заключения $(\varphi Q)(b) \& (\varphi R)(b)$. Следовательно, истинно высказывание $(\varphi Q)(b)$. А это и есть φG. Мы доказали, что если истинны высказывания φF_1 и φF_2, то истинно высказывание φG, т.е. что G –логическое следствие F_1 и F_2.□

Пример 2. В качестве второго примера докажем нелогичность рассуждения о первокурсниках, приведенное в §3. Мы записали это рассуждение в виде последовательности формул H_1, H_2, и H_3. Для доказательства нелогичности рассуждения надо найти интерпретацию φ, при которой формулы H_1 и H_2 истинны, а формула H_3 ложна. Пусть множество М состоит из трех элементов 2, 3, 4. Символы *С*, *Л* и *П* проинтерпретируем следующим образом:

$(\varphi C)(x)$ = “x – простое число”,

$(\varphi Л)(x)$ = “x – четное число”,

$(\varphi П)(x)$ = “$x>4$”,

т.е. *П* интерпретируется как тождественно ложный предикат. Символу *З* поставим в соответствие произвольный двухместный предикат. Тогда формулы H_1 и H_2 истинны, поскольку посылки импликаций этих формул ложны при любом x. А формула H_3 ложна. Чтобы убедиться в этом достаточно взять $x = 2$. Следовательно, рассуждение о первокурсниках нелогично. □

Определение. Множество формул

$$K = \{F_1(x_1, \ldots, x_l), \ldots, F_m(x_1, \ldots, x_l)\}$$

называется *выполнимым*, если существует интерпретация φ с областью M и элементы $a_1, \ldots, a_l \in M$ такие, что все высказывания $(\varphi F_1)(a_1, \ldots, a_l), \ldots, (\varphi F_m)(a_1, \ldots, a_l)$ истинны.

Пример 3. Множество формул $K = \{F_1 = (\forall x)(\exists y)(P(y) \& Q(x, y)), F_2 = (\forall y)Q(c, y), F_3 = \neg P(c)\}$ выполнимо. Возьмем в качестве области интерпретации множество нату-

ральных чисел ***N***. Символы P, Q и c проинтерпретируем следующим образом:

$(\varphi P)(u)$ = "u – простое число",

$(\varphi Q)(u, v)$ = "u меньше или равно v",

$\varphi(c) = 1$.

Тогда высказывание φF_1 означает, что для любого натурального числа x найдется простое число y, которое не меньше x, высказывание φF_2 означает, что 1 –наименьшее натуральное число, а высказывание φF_3 означает, что 1 – непростое число. Ясно, что все эти высказывания истинны, и поэтому множество формул K выполнимо. □

Понятия логического следствия и выполнимости в логике первого порядка связаны точно так же, как и в логике высказываний.

Теорема 2.2. Формула $G(x_1, \ldots, x_n)$ является логическим следствием формул $F_1(x_1, \ldots, x_n)$, ..., $F_k(x_1, \ldots, x_n)$ тогда и только тогда, когда множество формул $\{F_1(x_1, \ldots, x_n), \ldots, F_k(x_1, \ldots, x_n), \neg G(x_1, \ldots, x_n)\}$ невыполнимо.

Доказательство теоремы 2.2 аналогично доказательству теоремы 1.2 и поэтому не приводится.

§7. Нормальные формы

Как и в логике высказываний, в логике первого порядка вводятся нормальные формы. Мы рассмотрим две из них: предваренную нормальную и сколемовскую нормальную формы.

Определение. Формула G имеет *предваренную нормальную форму* (сокращенно: ПНФ), если

$$G = (Q_1 x_1)\ldots(Q_n x_n)H,$$

где $Q_1, \ldots, Q_n$ кванторы, а формула H не содержит кванторов.

Например, формула $(\forall x)(\exists y)(P(x, y) \,\&\, \neg Q(y))$ имеет предваренную нормальную форму, а формула $\neg(\forall x)(T(x) \,\&\, S(x, y))$ не имеет.

Теорема 2.3. Для всякой формулы F существует формула G, равносильная F и имеющая предваренную нормальную форму.

Доказательство теоремы легко следует из анализа алгоритма приведения к ПНФ.

Алгоритм приведения к предваренной нормальной форме

Шаг 1. Используя законы 21 и 20, исключить эквиваленцию и импликацию.

Шаг 2. Занести отрицание к атомарным формулам, пользуясь законами 17–19 и 26–27.

Шаг 3. С помощью законов 22–23, 28–31 вынести кванторы вперед, используя при необходимости переименование связанных переменных (законы 32–33).

Пример 1. Пусть

$$F = (\forall x)P(x) \to (\exists y)(\forall z)(P(y) \,\&\, Q(y, z)).$$

Выполнив шаг 1 (с помощью закона 20), получим формулу

$$F_1 = \neg(\forall x)P(x) \vee (\exists y)(\forall z)(P(y) \,\&\, Q(y, z)).$$

С помощью закона 26 перейдем к формуле

$$F_2 = (\exists x)\neg P(x) \vee (\exists y)(\forall z)(P(y) \,\&\, Q(y, z)).$$

Тем самым, шаг 2 также выполнен. Применим закон 30 $Q_1 = \exists$, $Q_2 = \exists$, $u = y$, получим формулу

$$F_3 = (\exists x)(\exists y)[\neg P(x) \vee (\forall z)(P(y) \vee Q(y, z))].$$

(Пользуемся тем, что $\neg P(x)$ не содержит y, а $(\forall z)(P(y)$ $\&$ $Q(y, z))$ не содержит x.) Так как формула $\neg P(x)$ не содержит z, применение закона 28 дает формулу

$$F_4 = (\exists x)(\exists y)(\forall z)[\neg P(x) \vee (P(y) \vee Q(y, z))].$$

Это и есть искомая формула, имеющая ПНФ и равносильная формуле F.

В рассмотренном примере выполнение шага 3 можно организовать по-другому. В формуле F_2 связанную переменную y заменим на переменную x (закон 33), получим формулу

$$F_3' = (\exists x)\neg P(x) \vee (\exists x)(\forall z)(P(x) \,\&\, Q(x, z)).$$

Используя закон 23, перейдем к формуле

$$F_4' = (\exists x)[\neg P(x) \vee (\forall z)(P(x) \,\&\, Q(x, z))].$$

Затем, как и в предыдущем абзаце, с помощью закона 28 вынесем квантор по z за квадратную скобку. Получим формулу

$$F_5' = (\exists x)(\forall z)[\neg P(x) \vee (P(x) \,\&\, Q(x, z)].$$

Формула F_5', как и формула F_4, имеет предваренную нормальную форму и равносильна формуле F. В некоторых ситуациях формула F_5' предпочтительнее формулы F_4, по-

скольку содержит меньше кванторов. (Кстати, бескванторную часть формулы F_5' можно упростить.) □

Перейдем к изучению сколемовской нормальной формы. Отметим вначале, что в логике первого порядка понятие конъюнктивной нормальной формы вводится точно так же, как и в логике высказываний. Полностью сохраняется алгоритм приведения к КНФ и утверждение теоремы 1.3.

Определение. Формула G имеет *сколемовскую нормальную форму* (сокращенно: СНФ), если

$$G = (\forall x_1)\ldots(\forall x_n)H,$$

где формула H не содержит кванторов и имеет конъюнктивную нормальную форму.

Например, формула $(\forall x)[P(x) \,\&\, (P(y) \vee Q(x, y))]$ имеет сколемовскую нормальную форму, а формулы $(\forall x)(\exists y)Q(x, y)$ и $(\forall x)[P(x) \vee (P(y) \,\&\, Q(x, y)]$ не имеют.

В отличие от предыдущего случая предваренной нормальной формы, мы здесь вначале рассмотрим алгоритм приведения к СНФ, а затем сформулируем теорему.

Алгоритм приведения к сколемовской нормальной форме

Шаги 1 – 3 – те же, что и в предыдущем алгоритме.

Шаг 4. Бескванторную часть привести к конъюнктивной нормальной форме (алгоритм описан в §5 главы 1).

Шаг 5. Исключить кванторы существования. Этот шаг изложим на примере. Пусть после выполнения четвертого шага мы получили формулу

$$F = (\exists x)(\forall y)(\exists z)(\forall u)(\exists v)H(x, y, z, u, v),$$

где H – не содержит кванторов. Предположим, что она не содержит константы c, символов одноместной функции f и двухместной функции g. Тогда в формуле H заменим x на c, z – на $f(y)$, v заменим на $g(y, u)$. Все кванторы существования вычеркнем. Получим формулу

$$G = (\forall y)(\forall u)H(c, y, f(y), u, g(y, u)).$$

Это и есть результат выполнения шага 5.

Пример 2. Приведем пример приведения к СНФ. Пусть

$$F = (\exists x)(\forall y)[P(x, y) \to (\exists z)(Q(x, z) \,\&\, R(y))].$$

Применяя законы 20 и 23, получаем формулу

$$F_1 = (\exists x)(\forall y)(\exists z)[\neg P(x, y) \vee (Q(x, z) \,\&\, R(y))].$$

Она имеет предваренную нормальную форму. Используя закон 12, приводим бескванторную часть к КНФ:

$$F_2 = (\exists x)(\forall y)(\exists z)[(\neg P(x, y) \vee Q(x, z)) \,\&\, (\neg P(x, y) \vee R(y))].$$

Сделав подстановку $x = a$, $z = f(y)$, получим искомую формулу

$$G = (\forall y)[(\neg P(a, y) \vee Q(a, f(y))) \,\&\, (\neg P(a, y) \vee R(y))].\square$$

Теорема 2.4. Для всякой формулы F существует формула G, имеющая сколемовскую нормальную форму и одновременно с F выполнимая или невыполнимая.

Доказательство. Пусть G – результат работы алгоритма приведения к СНФ. То, что результатом работы алгоритма является формула в сколемовской нормальной форме, ясно из описания алгоритма. Формула, которая получается после выполнения шагов 1–4, равносильна исходной, и, в частности, одновременно с ней выполнима или невыполнима.

Проанализируем шаг 5. Предположим вначале, что исключается квантор существования, впереди которого нет кванторов общности. Можно считать, что это первый квантор в записи формулы, т.е.

$$E(u_1, \ldots, u_n) = (\exists y)E'(y, u_1, \ldots, u_n).$$

(Формула E' может содержать кванторы.) Рассмотрим интерпретацию φ с областью M, при которой формула E выполнима. Выполнимость означает, что найдутся элементы $a_1, \ldots, a_n \in M$ такие, что высказывание $(\varphi E)(a_1, \ldots, a_n)$ или (что тоже самое) высказывание $(\exists y)(\varphi E')(y, a_1, \ldots, a_n)$ истинно. Отсюда следует, что существует элемент $b \in M$ такой, что высказывание $(\varphi E')(b, a_1, \ldots, a_n)$ также истинно. Исключение квантора существования по y на пятом шаге приводит к формуле $D(u_1, \ldots, u_n) = E'(c, u_1, \ldots, u_n)$, где c – константа, отсутствующая в E'. Рассмотрим интерпретацию ψ, которая совпадает с φ на всех символах предикатов и функций, входящих в запись формулы F, и $\psi(c) = b$. Тогда $(\psi D)(a_1, \ldots, a_n) = (\varphi E')(b, a_1, \ldots, a_n)$. Мы доказали, что если формула E выполнима, то выполнима и формула D.

Предположим теперь, что выполнима формула

$$D(u_1, \ldots, u_n) = E'(c, u_1, \ldots, u_n).$$

Это означает, что существует интерпретация ψ с областью M и элементы $a_1, \ldots, a_n \in M$ такие, что высказывание $(\psi E')(\psi(c), a_1, \ldots, a_n)$ истинно. Но отсюда следует истинность высказывания $(\exists y)(\psi E')(y, a_1, \ldots, a_n)$. Следовательно, формула $E(u_1, \ldots, u_n)$ выполнима. Мы доказали, что из выполнимости формулы D следует выполнимость формулы E.

Рассмотрим теперь случай, когда исключается квантор существования, впереди которого есть k кванторов общности, т.е.

$$E(u_1, \ldots, u_n) = (\forall x_1)\ldots(\forall x_k)(\exists y)E'(x_1, \ldots, x_k, y, u_{1, \ldots}, u_n).$$

(Формула E' может содержать кванторы.) Исключение квантора по у на шаге 5 приведет к формуле

$$D(u_1, \ldots, u_n) =$$
$$(\forall x_1)\ldots(\forall x_k)E(x_1, \ldots, x_k, f(x_1, \ldots, x_k), u_1, \ldots, u_n),$$

где f – символ k-местной функции, не содержащийся в E. Предположим, что формула E выполнима. Выполнимость означает существование интерпретации φ с областью M и элементов $a_1, \ldots, a_n \in M$ таких, что высказывание $(\varphi E)(a_1, \ldots, a_n)$ истинно. Истинность этого высказывания означает, что для любых элементов $x_1, \ldots, x_k \in M$ найдется элемент $y \in M$ такой, что высказывание $E'(x_1, \ldots, x_k, y, a_1, \ldots, a_n)$ истинно. Если для данных элементов $x_1, \ldots, x_k$ таких элементов y несколько, то зафиксируем один. Тем самым мы определили на M функцию i: $Mx\ldots xM \to M$ такую, что высказывание

$$(\varphi E')(x_1, \ldots, x_k, i(x_1, \ldots, x_k), a_1, \ldots, a_n)$$

истинно для всех $x_1, \ldots, x_k \in M$. Рассмотрим интерпретацию ψ, которая совпадает с φ на всех символах функций и предикатов, входящих в запись формулы E, и $(\psi f)(x_1, \ldots, x_n) = i(x_1, \ldots, x_n)$. Тогда

$$(\psi D)(a_1, \ldots, a_n) =$$
$$(\forall x_1)\ldots(\forall x_k)(\varphi E')(x_1, \ldots, x_k, i(x_1, \ldots, x_k), a_1, \ldots, a_n).$$

Последнее высказывание, как мы видели истинно. Следовательно, формула $D(u_1, \ldots, u_n)$ выполнима. Мы показали, что из выполнимости формулы E следует выполнимость формулы D.

Пусть выполнима формула D. Это означает, что существует интерпретация ψ с областью M и элементы $a_1, \ldots, a_n \in M$ такие, что высказывание $(\psi D)(a_1, \ldots, a_n)$ или (что то же самое) высказывание

$(\forall x_1)\ldots(\forall x_k)(\psi E')(x_1, \ldots, x_k, (\psi f)(x_1, \ldots, x_k), a_1, \ldots, a_n)$ истинно. Отсюда следует, что для любых $x_1, \ldots, x_k$ найдется y (равный $(\varphi f)(x_1, \ldots, x_k)$) такой, что высказывание

$$(\psi E')(x_1, \ldots, x_k, y, a_1, \ldots, a_n)$$

истинно. Следовательно, истинно высказывание

$$(\forall x_1)\ldots(\forall x_k)(\exists y)(\psi E')(x_1, \ldots, x_k, y, a_1, \ldots, a_n),$$

т.е. высказывание $(\psi E)(a_1, \ldots, a_n)$. Мы доказали, что из выполнимости формулы D следует выполнимость формулы E. □

§8. Невыразимость в логике первого порядка

В примерах, рассмотренных в предыдущих параграфах, мы видели, что логика первого порядка обладает значительными выразительными возможностями. Данный параграф посвящен доказательству того, что выразительные возможности этой логики в определенном смысле ограничены.

Рассмотрим предварительно понятие транзитивного замыкания двухместного предиката.

Определение. Пусть $S(x, y)$ – двухместный предикат, заданный на множестве M. Предикат $S^+(x, y)$ называется *транзитивным замыканием* предиката $S(x, y)$, если для любых $a, b \in M$ выполняется условие:

высказывание $S^+(a, b)$ истинно $\Leftrightarrow$ существует натуральное число n и элементы $c_0, \ldots, c_n$ множества M такие, что $a = c_0$, $c_n = b$ и все высказывания $S(c_0, c_1)$, $S(c_1, c_2)$, …, $S(c_{n-1}, c_n)$ истинны.

Если предикат представлять отношением, то транзитивному замыканию предиката соответствует транзитивное замыкание бинарного отношения.

Рассмотрим пример. Пусть M – множество натуральных чисел, а $S(x, y)$ – предикат "x непосредственно предшествует y", т.е. "$y = x+1$". Тогда транзитивным замыканием предиката $S(x, y)$ будет предикат "x меньше y".

Оказывается, не существует формулы логики первого порядка, которая выражала бы транзитивное замыкание двухместного предиката. Более точная формулировка содержится в следующем утверждении.

Теорема 2.5. Пусть P – символ двухместного предиката. Не существует формулы $F(x, y)$ логики первого порядка такой, что для любой интерпретации φ предикат $(\varphi F)(x, y)$ есть транзитивное замыкание предиката $(\varphi P)(x, y)$.

Доказательство теоремы 2.5 опирается на одно известное в логике первого порядка утверждение, которое называется теоремой компактности. В формулировке теоремы

компактности используется понятие логического следствия бесконечного множества формул. Определение этого понятия легко получается из определения логического следствия множества формул $F_1, \ldots, F_k$ из §6, и мы его приводить не будем.

Теорема компактности. Если формула G является логическим следствием бесконечного множества формул K, то G является логическим следствием некоторого конечного подмножества K' множества K.

Доказательство теоремы компактности будет приведено в следующей главе.

Приведем *доказательство* теоремы 2.5. Предположим противное: существует формула $F(x, y)$ логики первого порядка такая, что для любой интерпретации φ с областью M и любых $a, b \in M$ выполняется эквиваленция

$$(\varphi F)(a, b) \Leftrightarrow (\varphi P)^{+}(a, b).$$

Рассмотрим следующее множество формул

$$K = \{E_0(x, y), E_1(x, y), \ldots, E_n(x, y), \ldots\}:$$

$$E_0(x, y) = \neg P(x, y),$$

$$E_1(x, y) = \neg(\exists z_1)[P(x, z_1) \,\&\, P(z_1, y)],$$

$$E_2(x, y) = \neg(\exists z_1)(\exists z_2)[P(x, z_1) \,\&\, P(z_1, z_2) \,\&\, P(z_2, y)],$$

$$\cdots$$

$$E_n(x, y) = \neg(\exists z_1)\ldots(\exists z_n)[P(x, z_1) \,\&\, P(z_1, z_2) \,\&\ldots\&\, P(z_n, y)],$$

$$\cdots$$

Используя определение транзитивного замыкания и предположение о том, что формула $F(x, y)$ определяет транзитивное замыкание, получаем, что формула $\neg F(x, y)$ есть логическое следствие множества формул K. По теореме компактности $\neg F(x, y)$ есть логическое следствие некоторого конечного подмножества K' множества K. Можно считать, что

$$K' = \{E_0(x, y), E_1(x, y), \ldots, E_d(x, y)\}$$

для некоторого d.

Пусть $M = \{0, 1, \ldots, d+1, d+2\}$. На множестве M определим двухместный предикат S следующим образом:

$$S(u, v) \Leftrightarrow u+1 \text{ равно } v$$

Например, высказывания $S(0, 1)$, $S(1, 2)$ истинны, а высказывание $S(0, 2)$ ложно. Отметим, что высказывание $S^{+}(0, d+2)$ истинно. Рассмотрим интерпретацию φ, для которой $(\varphi P)(u, v) = S(u, v)$. Все высказывания $(\varphi E_0)(0, d+2)$,

$(\varphi E_1)(0,\ d+2)$, …,$(\varphi E_d)(0,\ d+2)$ истинны. Так как формула $\neg F(x,\ y)$ есть логическое следствие множества формул K', истинно высказывание $\neg(\varphi F)(0,\ d+2)$. С другой стороны, поскольку $F(x,\ y)$ определяет транзитивное замыкание и истинно высказывание $S^+(0,\ d+2)$ (другими словами, высказывание $(\varphi P)^+(0,\ d+2)$), то истинно высказывание $(\varphi F)(0,\ d+2)$. Мы доказали, что истинно и последнее высказывание, и его отрицание. Полученное противоречие показывает, что не существует формулы логики первого порядка, определяющей транзитивное замыкание предиката. □

§9. Многосортная логика первого порядка

Расширим понятие формулы, введя так называемые ограниченные кванторы. Допустим, что нам надо записать на языке логики предикатов следующее утверждение: “для всякого $x>5$ существует $y>0$ такое, что $xy = 1$”. Отметим, что здесь написано не “для всякого x” и “существует y”, а “для всякого $x>5$ ” и “существует $y>0$ ”. Если на это не обратить внимание, то получится формула $(\forall x)(\exists y)(xy = 1)$, имеющая другой смысл, нежели исходное утверждение. Для правильного перевода надо немного изменить исходное предложение по форме (не меняя, разумеется, смысла): “для всякого x справедливо, если $x>5$, то существует y такой, что $y > 0$ и $xy = 1$”. Правильный перевод имеет вид $(\forall x)[x>5 \rightarrow (\exists y)(y > 0\ \&\ xy = 1)]$.

Если рассматривать более длинные исходные предложения, то соответствующие им формулы логики предикатов будут, вообще говоря, довольно громоздкими. Для того, чтобы частично избавиться от усложнения при переводе на язык логики предикатов, вводятся так называемые ограниченные кванторы. Пусть $B(x)$ – формула с одной свободной переменной x. Тогда выражение $\forall B(x)$ называется *ограниченным квантором общности* $\exists B(x)$ – *ограниченным квантором существования.* С помощью ограниченных кванторов исходное предложение предыдущего абзаца можно записать довольно просто: $(\forall x>5)(\exists y>0)(xy = 1)$.

Более формально, ограниченные кванторы вводятся следующим образом: формула $(\forall B(x))F(x)$ есть сокращение формулы $(\forall x)(B(x)\rightarrow F(x))$, формула $(\exists B(x))F(x)$ – сокраще-

ние формулы $(\exists x)(B(x)\&F(x))$. Для ограниченных кванторов справедливы аналоги законов 22–33.

Ограниченные кванторы часто вводятся неявно. Для переменных, пробегающих множество истинности формулы $B(x)$, вводят специальное обозначение. Например, в геометрии довольно часто применяется следующее соглашение: буквами A, B, C, D, ... обозначаются точки, буквами a, b, c, d, ... – прямые, а буквами α, β, γ, ... – плоскости, т.е. с нашей точки зрения первые пробегают множество истинности формулы $T(x)$, вторые – $Пр(x)$, третьи – $Пл(x)$.

Последовательное оформление этой идеи приводит к понятию многосортной (многоосновной) логики предикатов. Строгих определений давать не будем, укажем только отличие от обсуждавшейся ранее (одноосновной или односортной) логики предикатов. Исходными для построения формул также являются множества $\boldsymbol{F}$, $\boldsymbol{R}$, $\boldsymbol{V}$. Только в этом случае переменные разбиты по сортам. В примере с геометрией таких сортов три: переменные, принимающие в качестве значений точки, прямые и плоскости. Далее, для каждого символа из $\boldsymbol{F}$ указано, какой сорт имеет первый аргумент, какой – второй и т.д., какой сорт имеет значение функции. Аналогичная информация имеется и для каждого символа предиката. Для интепретации берется не одно множество, а столько, сколько сортов переменных (эти множества называются основами). Для геометрии таких основ три: множество точек, множество прямых, множество плоскостей.

Приведем пример применения многоосновной логики предикатов. В этом примере будем использовать терминологию теории баз данных без пояснений. (По поводу этой терминологии см. книгу Дж. Ульмана из списка литературы.)

Рассмотрим информационную систему под условным названием “Сделки”. Система содержит сведения о сделках купли-продажи, произведенных некоторой фирмой. Предметом сделок служат партии товаров, определяемые номером партии, наименованием товара, единицей измерения и количеством. Используются следующие атрибуты: *НОМ* – номер партии товара, НАИМ – наименование товара, ЕД – единица измерения, КОЛ – количество единиц товара в партии, ДАТА – дата сделки, АГЕНТ – покупатель или

продавец, СЕК – номер секции склада, СРОК – срок годности. Информация хранится в виде отношений: ПАР (НОМ, НАИМ, ЕД, КОЛ), ПОК (НОМ, ДАТА, АГЕНТ), ПРОД (НОМ, ДАТА, АГЕНТ), СКЛАД (СЕК, НОМ, СРОК). Первое отношение содержит сведения о партиях товара, которые были предметом сделок, второе – сведения о покупках, третье – о продажах партий товара. В четвертом указывается, в какой секции склада хранится купленная (но еще не проданная) партия товара и срок годности товара в партии. Система может вычислять отношение РАН(x, y) = "x раньше y", определенное на доменах атрибутов ДАТА и СРОК.

Формализуем эту информационную систему в многоосновной логике предикатов следующим образом. Введем восемь сортов переменных (по количеству атрибутов), для каждого атрибута – свой сорт. Переменные будут принимать значения в доменах соответствующих атрибутов. Другими словами, области интерпретации (основы) будут состоять из доменов атрибутов. Переменные по сортам графически различать не будем. А для того, чтобы указать, что переменная x, например, изменяется по домену атрибута НАИМ, а y – по домену атрибута ЕД, будем писать: $x \in$НАИМ, $y \in$ЕД. Каждому отношению поставим в соответствие предикат той же местности, что и отношение, с соответствующими типами переменных. Предикат и отношение будем обозначать одинаково. Например, отношению ПАР(НОМ, НАИМ, ЕД, КОЛ) будет соответствовать предикат ПАР(x, y, u, v), где $x \in$НОМ, $y \in$НАИМ, $u \in$ЕД, $v \in$КОЛ. Сигнатура Σ, таким образом, будет содержать 5 символов предикатов:

Σ = {ПАР(x, y, u, v), ПОК(x, y, z), ПРОД(x, y, z), СКЛАД(x, y, z), РАН(x, y)}.

В сигнатуру Σ можно добавлять константы, интепретируемые как элементы доменов атрибутов.

Эта формализация позволяет запросы к информационной системе представлять формулами логики предикатов указанной сигнатуры. Рассмотрим следующий запрос:

Q_1. "Каковы номера партий товаров, купленных у фирмы β, и каково наименование товара в этих партиях?"

Запрашиваемая информация содержится в двух отношениях ПАР(НОМ, НАИМ, ЕД, КОЛ) и ПОК(НОМ, ДАТА, АГЕНТ), которые связаны номером партии товара, АГЕНТ есть фирма β. Если взять конъюнкцию предикатов

$$\text{ПАР}(x, y, u, v) \ \&\ \text{ПОК}(x, z, \beta),$$

то эта формула будет задавать пятиместный предикат, в котором, кроме запрашиваемой, будет содердаться информация о единицах, количестве единиц и дате сделок. Судя по запросу, эта дополнительная информация пользователя не интересует, поэтому на соответствующие переменные навесим кванторы существования. Получим формулу:

$$F_1(x, y) = x \in \text{НОМ} \ \&\ y \in \text{НАИМ} \ \&$$
$$(\exists u \in \text{ЕД})(\exists v \in \text{КОЛ})(\exists z \in \text{ДАТА})[\text{ПАР}(x, y, u, v) \ \&$$
$$\text{ПОК}(x, z, \beta)].$$

Формула $F_1(x, y)$ представляет собой перевод запроса Q_1 на язык многоосновной логики предикатов.

Рассмотрим еще ряд примеров перевода запросов на язык логики предикатов.

Q_2. “Каковы наименование товара, единицы измерения и количество единиц в партии товара, срок годности которого истекает 20.03.01?”

Q_3. “Для каких фирм срок годности товара, купленного у этих фирм, истекает 20.03.01?”

Q_4. “Какой товар хранится на складе более, чем в двух партиях?”

Q_5. “Какие из закупленных партий товаров в последствии проданы?”

Эти запросы на язык логики предикатов будут переведены формулами $F_2 - F_5$:

$$F_2(x, y, z) = x \in \text{НАИМ} \ \&\ y \in \text{ЕД} \ \&\ z \in \text{КОЛ} \ \&$$
$$(\exists u \in \text{СЕК})(\exists v \in \text{НОМ})(\exists w \in \text{СРОК})$$
$$[\text{ПАР}(v, x, y, z) \ \&\ \text{СКЛАД}(u, v, w) \ \&\ \text{РАН}(w, 20.03.01)],$$

$$F_3(x) = x \in \text{АГЕНТ} \ \&\ (\forall y \in \text{НОМ})(\forall z \in \text{ДАТА})(\forall u \in \text{СЕК})$$
$$(\forall v \in \text{СРОК})[\text{ПОК}(y, z, x) \ \&\ \text{СКЛАД}(u, y, v) \rightarrow$$
$$\text{РАН}(v, 20.03.01)],$$

$$F_4(x) = x \in \text{НАИМ} \ \&\ (\exists y_1, y_2 \in \text{НОМ})(\exists z_1, z_2 \in \text{ЕД})$$
$$(\exists u_1, u_2 \in \text{КОЛ})(\exists v_1, v_2 \in \text{СЕК})(\exists w_1, w_2 \in \text{СРОК})[y_1 \neq y_2 \ \&$$
$$\text{ПАР}(y_1, x, z_1, u_1) \ \&\ \text{СКЛАД}(v_1, y_1, w_1) \ \&$$
$$\text{ПАР}(y_2, x, z_2, u_2) \ \&\ \text{СКЛАД}(v_2, y_2, w_2),$$

$$F_5(x) = x \in \text{НОМ} \ \&\ (\exists y \in \text{НАИМ})(\exists z \in \text{ЕД})(\exists u \in \text{КОЛ})$$
$$(\exists v_1, v_2 \in \text{ДАТА})(\exists w_1, w_2 \in \text{АГЕНТ})$$
$$[\text{ПАР}(x, y, z, u) \ \&\ \text{ПОК}(x, v_1, w_1) \ \&\ \text{ПРОД}(x, v_2, w_2) \ \&$$
$$\text{РАН}(v_1, v_2)].$$

Рассмотрим второй вариант выбора сигнатуры для формализации запросов $Q_1 - Q_5$. Для этого вначале к имеющимся восьми основам (доменам восьми исходных атрибутов) добавим девятую основу: множество сделок СДЕЛ. Далее, вместо предикатов ПАР(x, y, u, v), ПОК(x, y, z), ПРОД(x, y, z), СКЛАД(x, y, z) введем функции:

наим: НОМ → НАИМ, *ед*: НОМ → ЕД,
кол: НОМ → КОЛ, *сдел*: СДЕЛ → НОМ,
тип: СДЕЛ → {пок, прод}, *дата:* СДЕЛ → ДАТА,
агент: СДЕЛ → АГЕНТ, *сек*: НОМ → СЕК,
срок: НОМ → СРОК,

а предикат РАН(x, y) оставим. Функции имеют естественный смысл. Например, функция *наим* номеру партиии товара ставит в соответствие наименование товара в этой партии, функция *сдел* ставит в соответствие сделке номер партии товара, относительно которого эта сделка была заключена. (Считаем, что предметом одной сделки является одна партия товара.) Новую сигнатуру будем обозначать буквой Δ. К сигнатуре Δ, как и к ∑, можно добавлять константы. Тогда запросы $Q_1 - Q_5$ будут переведены следующим образом:

$$G_1\ (x, y) = x \in \text{НОМ}\ \&\ y \in \text{НАИМ}\ \&\ y = наим(x)\ \&\ (\exists z \in \text{СДЕЛ})x = сдел(z)\ \&\ тип(z) = \text{пок}\ \&\ агент(z) = \beta],$$

$$G_2(x, y, z) = x \in \text{НАИМ}\ \&\ y \in \text{ЕД}\ \&\ z \in \text{КОЛ}\ \&\ (\exists u \in \text{НОМ})[x = наим(u)\ \&\ y = ед(u)\ \&\ z = кол(u)\ \&\ \text{РАН}(срок(u), 20.03.01)],$$

$$G_3(x) = x \in \text{АГЕНТ}\ \&\ (\forall y \in \text{НОМ})(\forall z \in \text{СДЕЛ})[сдел(z) = y\ \&\ агент(z) = x \rightarrow \text{РАН}(срок(y), 20.03.01)],$$

$$G_4(x) = x \in \text{НАИМ}\ \&\ (\exists u_1, u_2 \in \text{НОМ})(\exists v_1, v_2 \in \text{СЕК})[наим(u_1) = наим(u_2) = x\ \&\ u_1 \neq u_2\ \&\ сек(u_1) = v_1\ \&\ сек(u_2) = v_2],$$

$$G_5(x) = x \in \text{НОМ}\ \&\ (\exists v_1, u_2 \in \text{СДЕЛ})[x = сдел(u_1)\ \&\ x = сдел(u_2)\ \&\ тип(u_1) = \text{пок}\ \&\ тип(u_2) = \text{прод}\ \&\ \text{РАН}(дата(u_1), дата(u_2))].$$

Сигнатуру ∑ можно назвать “реляционной” (или ”предикатной”), а Δ – “функциональной”. Разумеется, возможны и другие варианты выбора сигнатуры.

Задачи

1. Установить, какой из кванторов определяется следующими выражениями: “для всякого x истинно $F(x)$”, “$F(x)$ при произвольном x”, “найдется x, такой что $F(x)$”, “для подходящего x верно $F(x)$”, “всегда имеет место $F(x)$”, “каждый элемент обладает свойством F”, “найдется, по крайней мере, один x такой, что $F(x)$”, “существует не менее одного x, такого что $F(x)$ ”, “свойство F присуще всем”, “каким бы ни был x $F(x)$ истинно”, “хотя бы для одного x верно $F(x)$ ”.

2. Дана алгебраическая структура $\langle \boldsymbol{N};\ x\le y\rangle$. Показать, что следующие предикаты определяются формулами сигнатуры $\sigma = (\le)$:

а) “x меньше y”, б) “y равно $x+1$”,
в) “x равно 1”, г) “x равно 2”,
д) “y лежит между x и z”.

3. Дана алгебраическая структура $\langle \boldsymbol{N};\ x|y\rangle$. Показать, что следующие предикаты определяются формулами сигнатуры $\sigma = (\ |\ )$ ($x|y$ означает, что x делит y нацело):

а) “x равно 1”, б) “z есть $HOD(x, y)$”,
в) “z есть $HOK(x, y)$”, г) “x – простое число”.

Можно ли определить предикаты “x – четное число”, “x меньше y” формулой этой же сигнатуры?

4. Рассмотрим алгебраическую структуру $\langle \boldsymbol{N};\ x+y,\ xy,\ x\le y\rangle$. Для каждой из формул:

а) $(\forall y)(y\le x \to x\le y)$,
б) $(\exists y)(x=y+y)$,
в) $(\forall u)(\forall v)(x=uy \to x=u \vee x=v)$,
г) $(\exists y)(\forall z)(z\le x \to x\le z \vee z=y)$,
д) $y\le z\ \&\ x\le z\ \&\ (\forall u)(y\le u\ \&\ x\le u \to z\le u)$

найти предикат из следующего списка, который эта формула определяет:

а) “x – простое число или x равно 1”,
б) “x – четное число”,
в) “x равно 1”,
г) “z есть наибольшее из чисел x и y”,
д) “x принадлежит $\{1, 2\}$”.

5. На множестве M задан одноместный предикат $P(x)$. Выразить следующие утверждения формулами сигнатуры $\sigma = (P)$:

а) “существует не менее одного элемента x, удовлетворяющего предикату $P(x)$”,

б) “существует не более одного элемента x, удовлетворяющего предикату $P(x)$”.

в) “существует точно один элемент x, удовлетворяющий предикату $P(x)$”,

г), д), е) – утверждения а), б), в) с заменой “один” на “два”.

6. Пусть M – множество всех точек, прямых и плоскостей трехмерного пространства. Рассмотрим алгебраическую систему $< M; x \in y, p(x), l(x), pl(x) >$, где $\in$ – отношение принадлежности, $p(x)$ означает, что x есть точка, $l(x)$ – x есть прямая, $pl(x)$ – x есть плоскость.

Выразить следующие предикаты формулами указанной сигнатуры:

а) “плоскости x и y имеют общую точку”,

б) “если плоскости x и y имеют общую точку, то они имеют общую прямую”,

в) “прямые x и y имеют общую точку”,

г) “прямые x и y параллельны”,

д) “прямые x, y и z образуют треугольник”.

В формулах можно использовать ограниченные кванторы.

7. Подберите сигнатуру и представьте следующие рассуждения в виде последовательности формул логики предикатов.

7.1. Некоторые из первокурсников знакомы со всеми второкурсниками, а некоторые из второкурсников – спортсмены. Следовательно, какие-то первокурсники знакомы с некоторыми спортсменами.

7.2. Членом правления клуба может быть каждый совершеннолетний член клуба. Игорь и Андрей – члены клуба. Игорь – совершеннолетний, а Андрей старше Игоря. Следовательно, Андрей может быть членом правления клуба.

7.3. Таможенные чиновники обыскивают всякого, кто въезжает в страну, кроме высокопоставленных лиц. Некоторые люди, способствующие провозу наркотиков, въезжали в страну и были обысканы исключительно людьми, также способствовавшими провозу наркотиков. Никто из высокопоставленных лиц не способствовал провозу наркоти-

ков. Следовательно, некоторые из таможенников способствовали провозу наркотиков.

8. Пусть $F(x, y) = P(x, y)\ \&\ (\exists z)(P(x, z)\&P(z, y))$ и $M = \{1, 2, 3, 4, 5\}$. Найти предикаты, которые соответствуют формуле $F(x, y)$ при следующих интерпретациях:

а) $(\varphi P)(x, y)$ = "x меньше y ",

б) $(\varphi P)(x, y)$ = "x меньше или равно y ",

в) $(\varphi P)(x, y)$ = "x делит y нацело и $x \neq y$",

г) $(\varphi P)(x, y)$ = "y равно $x+1$".

9. Пусть $F(x) = x \neq a\ \&\ (\forall y)(D(y, x) \to y=a \vee y=x)$ и $M = \{1, \ldots, 9\}$. Найти предикаты, которые соответствуют формуле $F(x)$ при следующих интерпретациях:

а) $(\varphi D)(x)$ = "x делит y нацело", $\varphi(a) = 1$;

б) $(\varphi D)(x)$ = "x меньше или равно y ", $\varphi(a) = 1$;

в) $(\varphi D)(x)$ = "x делит y нацело", $\varphi(a) = 2$.

10. Пусть $F(x) = P(x) \to Q(a, g(x))$, $M = \{0, 1\}$. Найти предикаты, которые соответствуют формуле $F(x)$ при следующих интепретациях:

а) $P(x)$ = "x не равно 0", $Q(x, y)$ = "x меньше y", $a = 0$, $g(x) = x+1$;

б) $P(x)$ = "x не равно 1", $Q(x, y)$ = "x меньше y", $a = 0$, $g(x) = 0$;

в) $P(x)$ = "x не равно 1", $Q(x, y)$ = "x меньше y", $a = 1$, $g(x) = x+1$, где + – сложение по модулю 2.

11. Пусть $F(x) = P(x)\ \&\ (\forall y)(P(y) \to D(x, y))$, $M = \{2, 3, 4, 6, 9\}$. Найти предикаты, которые соответствуют $F(x)$ при следующих интерпретациях:

а) $P(x)$ = "x – простое число", $D(x, y)$ = "x меньше или равно y";

б) $P(x)$ = "x – нечетное число", $D(x, y)$ = "x делит y";

в) $P(x)$ = "x не равно 4", $D(x, y)$ = "x меньше или равно y".

Существуют ли интерпретации, при которой формуле $F(x)$ соответствуют предикаты: а) "$x = 4$", б) "x – четное число?"

12. Выяснить, будут ли равносильны следующие пары формул:

а) $(\forall x)(F(x) \vee G(x))$ и $(\forall x)F(x) \vee (\forall x)G(x)$;

б) $(\exists x)(F(x)\ \&\ G(x))$ и $(\exists x)F(x)\ \&\ (\exists x)G(x)$;

в) $(\forall x)(F(x) \to G(x))$ и $(\forall x)F(x) \to (\forall x)G(x)$;

г) $(\forall x)F(x) \to (\forall x)G(x)$ и $(\exists x)(\forall y)(F(x) \to G(y))$;
д) $(\exists x)(F(x) \to G(x))$ и $(\exists x)F(x) \to (\exists x)G(x)$;
е) $(\exists x)F(x) \to (\exists x)G(x)$ и $(\forall x)(\exists y)(F(x) \to G(y))$;
ж) $(\exists x)(F(x) \leftrightarrow G(x))$ и $(\exists x)F(x) \leftrightarrow (\exists x)G(x)$;
з) $(\forall x)(F(x) \leftrightarrow G(x))$ и $(\forall x)F(x) \leftrightarrow (\forall x)G(x)$.

13. Доказать равносильность формул:
а) $F = \neg(\exists x)[(\forall y)P(x, y) \to (\forall z)(P(z, z) \vee Q(z))]$ и
$G = (\forall x)(\forall y)(\exists z)[P(x, y) \,\&\, \neg P(z, z) \,\&\, \neg Q(z)]$;
б) $F = \neg(\forall x)[T(x) \to (\exists y)(\forall z)(R(y, z) \,\&\, T(z) \to R(z, z))]$ и
$G = (\exists x)(\forall y)(\exists z)[T(x) \,\&\, \neg R(z, z) \,\&\, \neg(R(y, z) \to \neg T(z))]$;
в) $F = (\forall x)[(\forall y)P(x, y) \to (\exists z)(P(x, z) \,\&\, Q(z))]$ и
$G = (\forall x)(\exists u)[P(x, u) \to Q(u)]$;
г) $F = \neg(\exists x)[(\exists y)T(x, y) \to (\forall z)(S(x, z) \vee Q(z))]$ и
$G = (\forall x)(\exists y)(\exists z)[T(x, y) \,\&\, \neg(\neg S(x, z) \to Q(z))]$;
д) $F = \neg(\forall x)[(\forall y)T(x, y) \to (\exists z)(T(x, z) \,\&\, Q(z))]$ и
$G = (\exists x)(\forall u)(T(x, u) \,\&\, \neg Q(u))$.

14. Привести к предваренной нормальной форме:
а) $(\forall x)F(x) \to (\forall y)G(y)$;
б) $(\exists x)F(x) \to (\exists x)G(x)$;
в) $(\forall x)F(x) \to (\exists y)G(y)$;
г) $(\exists x)F(x) \to (\forall y)G(y)$;
д) $(\forall x)P(x, y) \to (\exists z)[P(y, z) \vee (\forall u)(Q(u) \to P(z, z))]$.

15. Привести к сколемовской нормальной форме:
а) $(\exists x)[P(x) \,\&\, (\forall y)(S(y) \to T(x, y))]$,
б) $(\forall x)[Q(x) \to (\exists y)(\forall u)(R(x, y) \,\&\, S(y, u))]$,
в) $(\forall x)(\forall y)(\exists z)(\forall u)(\exists v)[L(x, y, z) \,\&\, M(z, u, v)]$,
г) $(\forall x)[(\forall y)P(x, y) \to (\exists z)(Q(x, z) \,\&\, R(z))]$,
д) $(\forall x)[(\exists y)P(x, y) \to (\forall z)(Q(x, z) \vee P(z))]$,
е) $(\forall x)[(\exists y)P(x, y) \leftrightarrow (\exists z)Q(x, z)]$.

16. Показать, что в следующих случаях формула G не является логическим следствием множества формул K:
а) $G = (\forall x)\neg R(x)$, $K = \{(\exists x)R(x) \to (\exists x)Q(x), \neg Q(a)\}$;
б) $G = (\forall x)R(x, x)$, $K = \{(\forall x)(\forall y)(R(x, y) \to R(y, x)), (\forall x)(\forall y)(\forall z)(R(x, y) \,\&\, R(y, z) \to R(x, z))\}$;
в) $G = (\exists x)(P(x) \,\&\, \neg R(x))$, $K = \{(\forall x)[P(x) \to (\exists y)(Q(y) \,\&\, S(x, y))], (\exists x)[R(x) \,\&\, (\forall y)(Q(y) \to \neg S(x, y)], (\exists x)P(x)\}$.

17. Дано утверждение: “Некоторые из первокурсников знакомы с кем-либо из спортсменов. Но ни один из первокурсников не знаком ни с одним любителем подледного ло-

ва”. Какие из следующих утверждений будут следствием этого и почему:

а) “ни один спортсмен не является любителями подледного лова”,

б) ”некоторые из спортсменов не являются любителями подледного лова”,

в) “найдется спортсмен, который любит подледный лов”?

18. Докажите нелогичность следующих рассуждений, построив интерпретацию, при которой посылки истинны, а заключение ложно.

18.1. Все студенты нашей группы – члены клуба “Спартак”. А некоторые члены клуба “Спартак” занимаются спортом. Следовательно, некоторые студенты нашей группы занимаются спортом.

18.2. Некоторые студенты нашей группы – болельщики “Спартака”. А некоторые болельщики “Спартака” занимаются спортом. Следовательно, некоторые студенты нашей группы занимаются спортом.

18.3. Каждый первокурсник знаком с кем-либо из студентов второго курса. А некоторые второкурсники – спортсмены. Следовательно, каждый первокурсник знаком с кем-либо из спортсменов.

19. Рассмотрим информационную систему под условным названием “Кадры”, которая содержит сведения о сотрудниках некоторой организации. Для представления информации используются атрибуты: ФАМ – фамилия сотрудника, ПОЛ – пол сотрудника, ВОЗР – возраст, ДОЛЖ – должность, НОМ – номер отдела (подразделения) этой организации. Сведения хранятся в виде двух отношений СОТР(ФАМ, НОМ, ДОЛЖ), АНК(ФАМ, ПОЛ, ВОЗР). Первое отношение содержит фамилии сотрудников, их должности и номера отделов, где работают эти сотрудники. Второе отношение хранит анкетные данные: фамилию, пол и возраст сотрудников. Кроме того, система может вычислять отношения МЕН(x, y) = ”x меньше y”, определенное на множестве натуральных чисел, точнее, на домене атрибута ВОЗР.

Формализуем систему “Кадры” в многоосновной логике предикатов. Введем пять сортов переменных (по количеству атрибутов), для каждого атрибута – свой сорт. Переменные будут принимать значения в доменах соответствующих

атрибутов. Эти домены и будут являться основами. Переменные графически можно не различать, а область изменения переменной указывать с помощью принадлежности домену соответствующего атрибута. Каждому из отношений поставим в соответствие предикат той же местности, что и отношение, с соответствующими типами переменных. Сигнатура Σ будет содержать три символа предиката:

$$\Sigma = \{\text{СОТР}(x, y, z), \text{АНК}(x, y, z), \text{МЕН}(x, y)\}.$$

К Σ можно добавлять константы, интепретируемые как элементы доменов.

Перевести следующие запросы на язык логики первого порядка сигнатуры Σ.

1.Кто из сотрудников–мужчин старше 40 лет?

2.Кто из сотрудников старше 40 лет и в каком отделе работает?

3.Кто из программистов старше 40 лет и в каком отделе работает?

4.В каких отделах все программисты моложе 40 лет?

5.В каких отделах работают пенсионеры7

6.В каких отделах все программисты – пенсионеры?

20. Рассмотрим предметную область, которую условно назовем “Механическая обработка деталей”. Эта область состоит из следующих множеств: деталей, станков, операций, типов деталей, типов станков, типов операций, которые мы будем обозначать соответственно как ДЕТ, СТ, ОПЕР, ТИПДЕТ, ТИПСТ, ТИПОПЕР. Введем еще и множество моментов времени – ВРЕМЯ. Будем предпо-лагать, что время измеряется в некоторых единицах, например, в минутах, и отождествим множество ВРЕМЯ с множеством натуральных чисел. Зафиксируем сигнатуру, состоящую из предиката $\leq$ и функции + на множестве ВРЕМЯ и следующих одноместных функций:

дет: ОПЕР $\to$ ДЕТ, *ст*: ОПЕР $\to$ СТ,

нач: ОПЕР $\to$ ВРЕМЯ, *кон*: ОПЕР $\to$ ВРЕМЯ,

тип дет: ДЕТ $\to$ ТИПДЕТ, *типст:* СТ $\to$ ТИПСТ,

типопер: ОПЕР $\to$ ТИПОПЕР.

В сигнатуру можно добавлять константы, интепретируемые как элементы указанных множеств. Например, *ствал* – ступенчатый вал (элемент множества ТИПДЕТ),- *фрст* – фрезерный станок (элемент множества ТИПСТ), *токобр* – токарная обработка (элемент множества

ТИПОПЕР), 6 – шесть моментов времени (шесть минут), *дет1* – деталь 1, *ст2* – станок 2, *опер3* – операция 3.

Записать в виде формул многосортной логики предикатов следующие предложения (обозначения сортов переменных не зафиксированы, поэтому для того, чтобы ограничить область изменения переменной x, скажем, множеством деталей, можно употреблять запись $x \in$ ДЕТ).

1. Деталь *дет1* обрабатывается на станке *ст2*.
2. Операция *опер 1* совершается над деталью *дет 2* в течение 10 моментов времени.
3. Фрезерование шлицев длится 6, а фрезерование резьбы 9 моментов времени.
4. Токарная обработка ступенчатого вала длится 10 моментов времени.
5. Операция *опер 3* совершается на фрезерном станке в течение 6 моментов времени.
6. Каждая деталь должна сначала пройти фрезерование торцов, а затем – фрезерование шлицев.
7. Каждый вал со шлицами должен сначала пройти токарную обработку, а затем фрезерование шлицев.
8. До операции долбление зубьев вал-шестерня должен пройти токарную обработку.
9. Первая оперция для всех деталей – фрезерование торцов.
10. Последняя операция для всех типов деталей – фрезерование резьбы или закалка.
11. Между операциями фрезерования торцов и фрезерования резьбы проводится токарная обработка.

21. Операции *дет, ст, нач, кон, типопер* из предыдущей задачи заменим на предикат ОПЕР(ДЕТ, СТ, ВРНАЧ, ВРКОН, ТИПОПЕР), где ВРНАЧ и ВРКОН – время начала и окончания операции. Предикат $\leq$ и функции +, *типдет, типст* оставим. Записать формулами измененной сигнатуры утверждения 1, 3–4, 6–11.

22. Расмотрим предметную область, которую можно назвать “Учеба на факультете”. Для представления информации об этой предметной области введем два языка многосортной логики предикатов. Первый содержит следующие имена основных множеств: СТУД, ПРЕП, ПРЕД, ГР, КУРС, АУД, ДЕНЬ, НАЧ, которые интерпретируются соответственно как множества студентов, преподавателей, изучае-

мых предметов, групп, курсов, аудиторий, рабочих дней недели, времени начала занятий. На этих множествах заданы предикаты ГР(СТУД, ГР), КУРС(ГР, КУРС), РАСП(НАЧ, ДЕНЬ, ГР, ПРЕД, ПРЕП, АУД), РАНЬШЕ(НАЧ, НАЧ). Первый определяет принадлежность студента группе, второй – группы курсу, третий представляет собой факультетское расписание на неделю (предполагается, что нет поточных лекций, лабораторных занятий с частью группы и что все занятия проводятся каждую неделю). Последний предикат определяет, когда одно занятие проводится раньше другого по времени (в течение одного дня). В сигнатуру можно добавлять константы, которые интепретируются как элементы указанных множеств. Например, *иванов* – студент, *петров* – преподаватель, *мт–101* – группа, *9.00* – начало занятий. *физкуль* – физкультура.

Второй язык, кроме указанных выше имен основных множеств, содержит множество ЗАН, которое интепретируется как множество занятий на факультете, т.е. как объединение множеств занятий, проведенных в группах факультета за неделю. Сигнатура второго языка содержит прежний предикат РАНЬШЕ (НАЧ, НАЧ) и следующие одноместные функции:

нач: ЗАН → НАЧ,	*день*: ЗАН → ДЕНЬ,
гр: ЗАН → ГР,	*пред*: ЗАН → ПРЕД,
преп: ЗАН → ПРЕП,	*ауд:* ЗАН → АУД,
номгр: СТУД → ГР,	*номк*: ГР → КУРС,

с естественной интепретацией. Например, функция *нач* ставит в соответствие занятию время его начала, а функция *пред* – предмет, который изучается на этом занятии.

Перевести на каждый из языков следующие утверждения.

1.Один и тот же преподаватель не может в одно и то же время проводить занятия в разных аудиториях.

2.Два занятия по одному предмету в один и тот же день не проводятся.

3.Занятия физкультурой проводятся сразу во всех группах.

4.В течение недели проводятся два занятия физкультурой.

5.Занятия физкультурой проводятся последней парой.

6.В субботу проводится не более трех занятий.

7.У каждой группы 4 и 5 курсов есть день, свободный от аудиторных занятий.

8.В группе *мт–101* каждый день есть не менее трех занятий.

9.Если в группе в какой то день есть занятие, то есть, по крайней мере, еще одно занятие.

23. Рассмотрим предметную область, которую условно назовем “Аэропорт”. Выбрать сигнатуру многосортной логики для представления следующей информации о (недельном) расписании движения самолетов и выразить указанные утверждения формулами.

1.В Москву каждый день выполняется не менее трех рейсов.

2.В Ростов есть, по крайней мере, три рейса в неделю.

3.Нет двух рейсов до Минеральных Вод в один день.

4.В понедельник рейс до Краснодара выполняется раньше рейса до Ростова.

5.Первый рейс до москвы выполняется раньше рейса до Ростова.

6.Между первыми двумя рейсами до Москвы есть рейс до Новосибирска.

Каким образом можно расширить выбранную сигнатуру, чтобы представить следующую информацию о промежуточных посадках?

7.Рейс до Якутска имеет две посадки.

8.Ни один рейс не имеет более трех посадок.

Ответы указания и решения

2. Приведем соответствующие формулы:

а) $x \le y \ \& \ \neg y \le x$,

б) $x < y \ \& \ (\forall z)(x < z \ \& \ z \le y \to z = y)$,

в) $(\forall y)(x \le y)$,

г) $(\exists z)(\forall y)[z \le y \ \& \ (\forall u)(z < u \ \& \ u \le x \to u = x)]$,

д) $(x < y \ \& \ y < z) \vee (z < y \ \& \ y < x)$.

3. Приведем соответствующие формулы:

а) $(\forall y)(x \mid y)$,

б) $z \mid x \ \& \ z \mid y \ \& \ (\forall u)(u \mid x \ \& \ u \mid y \to u \mid z)$,

в) $x \mid z \ \& \ y \mid z \ \& \ (\forall u)(x \mid u \ \& \ y \mid u \to z \mid u)$,

г) $\neg(x = 1) \ \& \ (\forall y)(y \mid x \to y = 1 \vee y = x)$.

Предикаты “x – четное число” и “x меньше y” невыразимы формулой сигнатуры σ. Докажем это. Пусть P – мно-

жество простых чисел. Рассмотрим отображение φ: $P \to P$ такое, что $\varphi(2) = 3$, $\varphi(3) = 2$ и $\varphi(p) = p$ для всех простых чисел p, отличных от 2 и 3. Отображение φ расширим на все множество натуральных чисел $\boldsymbol{N}$ следующим образом: Пусть $n = 2^{\alpha}3^{\beta}{p_1}^{\gamma 1 \ldots}{p_k}^{\gamma k}$, где $p_1, \ldots, p_k$ – простые числа, отличные от 2 и 3. Положим:

$$\varphi(n) = 3^{\alpha}2^{\beta}{p_1}^{\gamma 1 \ldots}{p_k}^{\gamma k} \text{ и } \varphi(1) = 1.$$

Например, $\varphi(2) = 3$, $\varphi(12) = 18$, $\varphi(42) = 42$, $\varphi(60) = 90$. Отображение φ сохраняет предикат делимости. Это означает, что

$$u|v \Leftrightarrow \varphi(u)|\varphi(v)$$

для любых u, $v \in N$. Индукцией по построению формулы можно доказать, что для любой формулы $F(x_1, \ldots x_n)$ сигнатуры σ и любых натуральных чисел $a_1, \ldots, a_n$ высказывание $F(a_1, \ldots a_n)$ истинно тогда и только тогда, когда истинно $F(\varphi(a_1), \ldots, \varphi(a_n))$. Предположим теперь, что предикат "x – четное число" определяется формулой $G(x)$. Тогда высказывание $G(2)$ истинно тогда и только тогда, когда $G(3)$ истинно. Но $G(2)$ истинно, а $G(3)$ ложно. Полученное противоречие доказывает, что предикат "x – четное число" невыразим формулой сигнатуры σ. Аналогично доказывается невыразимость предиката "x меньше y".

4. Соответствие формул предикатам содержится в следующей таблице:

Формула	а	б	в	г	д
Предикат	в	б	а	д	г

5. Приведем соответствующие формулы:

а) $(\exists x)P(x)$,

б) $(\forall x)(\forall y)(P(x) \,\&\, P(y) \to x{=}y)$,

в) $(\exists x)[P(x) \,\&\, (\forall y)(P(y) \to x{=}y)]$,

г) $(\exists x)(\exists y)[x \neq y_2 \,\&\, P(x) \,\&\, P(y)]$,

д) $(\forall x)(\forall y)(\forall z)[P(x) \,\&\, P(y) \,\&\, P(z) \to x{=}y \vee x{=}z \vee y{=}z]$,

е) $(\exists x)(\exists y)[x \neq y \,\&\, P(x) \,\&\, P(y) \,\&\, (\forall z)(P(z) \to x{=}z \vee y{=}z)]$.

6. Приведем соответствующие формулы:

а) $pl(x) \,\&\, pl(y) \,\&\, (\exists z)(p(z) \,\&\, z \in x \,\&\, z \in y)$,

б) $pl(x) \,\&\, pl(y) \,\&\, [(\exists z)(p(z) \,\&\, z \in x \,\&\, z \in y) \to (\exists u)(l(u) \,\&\, u \in x \,\&\, u \in y)]$,

в) $l(x) \,\&\, l(y) \,\&\, (\exists z)(p(z) \,\&\, z \in x \,\&\, z \in y)$,

г) $l(x) \,\&\, l(y) \,\&\, (\exists u)(pl(u) \,\&\, x \in u \,\&\, y \in u) \,\&\, \neg(\exists v)(p(v) \,\&\, v \in x \,\&\, v \in y)$,

д) $l(x)$ & $l(y)$ & $l(z)$ & $(\exists u)(\exists v)(\exists w)[p(u)$ & $p(v)$ & $p(w)$ & $u \in x$ & $u \in y$ & $v \in x$ & $v \in z$ & $w \in y$ & $w \in z$ & $u \neq v$ & $u \neq w$ & $v \neq w]$.

7. Приведем решение задачи 7.3. Пусть $T(x)$ означает "x – таможенник", $B(x)$ –"x въезжает в страну", $Л(x)$ – "x высокопоставленное лицо", $H(x)$ – "x способствует провозу наркотиков", $O(x, y)$ – "x обыскивает y". Тогда рассуждение 7.3 может быть переведено на язык логики первого порядка следующим образом:

$$F_1 = (\forall x)[B(x) \ \& \ \neg Л(x) \to (\exists y)(T(y) \ \& \ O(y, x))],$$
$$F_2 = (\exists x)[B(x) \ \& \ H(x) \ \& \ (\forall y)(O(y, x) \to H(y))],$$
$$F_3 = \neg(\exists x)[Л(x) \ \& \ H(x))],$$
$$G = (\exists x)(T(x) \ \& \ H(x)).$$

8. Формуле $F(x, y)$ соответствуют следующие предикаты:

а) "x меньше y и между ними находится в точности один элемент",

б) "x меньше или равно y",

в) "x=1 и y=4",

г) "$x \neq x$", т.е. тождественно ложный предикат.

9. Формуле $F(x)$ соответствуют следующие предикаты:

а)"x – простое число", б) "x равно 2",

в) "x равно 1".

10. Формуле $F(x)$ соответствуют следующие предикаты:

а) "x равно 0", б) "x равно 1",

в) "x равно 1".

11. Формуле $F(x)$ соответствуют следующие предикаты:

а) "x равно 2", б) "x равно 3",

в) "x равно 2",

Предикат "x равно 4" соответствует формуле $F(x)$ при интерпретации $P(x)$ = "x равно 4", $D(x, y)$ = "x больше или равно y", а предикат "x – четное число" соответствует этой формуле при интепретации $P(x)$ = "x – четное число" и $D(x, y)$ = "x равно x".

12. Формулы равносильны только в случаях г) и е).

13. Приведем решение задачи 13в). Пусть

$$F = (\forall x)[(\forall y)P(x, y) \to (\exists z)(P(x, z) \ \& \ Q(z))] \text{ и}$$
$$G = (\forall x)(\exists u)[P(x, u) \to Q(u)].$$

Применим к формуле F последовательно законы 20 и 26, получим формулу

$$F_1 = (\forall x)[(\exists y)\neg P(x, y) \vee (\exists z)(P(x, z) \ \& \ Q(z))].$$

Затем, пользуясь законом 33, в формуле $(\exists y)\neg P(x, y)$ переменную y заменим на переменную u, а в формуле $(\exists z)(P(x, z) \,\&\, Q(z))$ переменную z заменим тоже на u. Формула F_1 после этих замен превратится в формулу

$$F_2 = (\forall x)[(\exists u)\neg P(x, u) \vee (\exists u)(P(x, u) \,\&\, Q(u))].$$

Применив закон 23, вынесем квантор вперед:

$$F_3 = (\forall x)(\exists u)[\neg P(x, u) \vee (P(x, u) \,\&\, Q(u))]$$

Используя последовательно законы 12, 16 и 1, получим формулу:

$$F_4 = (\forall x)(\exists u)[\neg P(x, u) \vee Q(u)].$$

Осталось заметить, что в силу закона 20 формула $\neg P(x, u) \vee Q(u)$ равносильна формуле $P(x, u) \to Q(u)$. Равносильность F и G доказана.

14. Указание. Воспользоваться алгоритмом приведения к ПНФ (см.§6).

15. Указание. Воспользоваться алгоритмом приведения к СНФ (см.§6).

16. Приведем решение задачи 16в). Надо построить интерпретацию, при которой формулы из K истинны, а формула из G ложна. Рассмотрим множество $M = \{1, 2, 3\}$ и интерпретацию φ: $(\varphi P)(x)$ = "x равно 1", $(\varphi Q)(x)$ = "x равно 2", $(\varphi R)(x)$ = "x равно1 или x равно 2", $(\varphi S)(x, y)$ = "x равно1 и y равно 2". Легко проверить, что φ – искомая интерпретация.

17. Логическим следствием будет только утверждение б).

18. Приведем решение задачи 18.1. Пусть $St(x)$ означает, что "x – студент нашей группы", $M(x)$ – "x является членом клуба "Спартак"", $Sp(x)$ – "x занимается спортом". Тогда рассуждение 18.1 переводится на язык логики первого порядка последовательностью формул:

$$F_1 = (\forall x)(St(x) \to M(x)),$$
$$F_2 = (\exists x)(M(x) \,\&\, Sp(x)),$$
$$G = (\exists x)(St(x) \,\&\, Sp(x)).$$

Рассмотрим множество $M = \{1, 2, 3\}$ и интерпретацию φ: $(\varphi St)(x)$ = "x равно 1", $(\varphi M)(x)$ = "x равно 1 или x равно 2", $(\varphi Sp)(x)$ = "x равно 2". Легко видеть, что $\varphi(F_1) = \varphi(F_2) = 1$ и $\varphi(G) = 0$.

Глава 3. Исчисление предикатов

В этой главе мы охарактеризуем понятие тождественной истинности и логического следствия некоторым формальным образом. А именно, построим так называемое исчисление предикатов (и как частный случай – исчисление высказываний). Будут введены понятия аксиомы и правила вывода. Предметом рассмотрения будут фактически только те формулы, которые получаются из аксиом и правил вывода. Примером такого подхода является аксиоматический метод в геометрии. Отличие логики предикатов от геометрии состоит в том, что здесь фиксируются не только исходные утверждения (аксиомы), но и способы получения следствий (правила вывода).

§ 1. Об определении некоторых семантических понятий

В этой главе мы несколько изменим определение формулы логики предикатов и определение некоторых семантических понятий.

Начнем с понятия формулы. Традиция аксиоматизации той или иной области математики предполагает использовать небольшое число аксиом. В то же время аксиоматика – это формальное определение базовых понятий. В случае логики предикатов – это понятие атомарной формулы и семь логических операций. Но семь основных операций – это довольно много. Поступим поэтому следующим образом:

1) исходными объектами для построения формул будут, попрежнему множества $\boldsymbol{F}$, $\boldsymbol{G}$ и $\boldsymbol{V}$, только сейчас сигнатурой будем называть объединение $\boldsymbol{F} \cup \boldsymbol{G} \cup \boldsymbol{V}$;

2) определение терма и атомарной формулы оставим в прежнем виде,

3) из логических связок оставим только $\neg$ и $\rightarrow$;

4) из кванторов оставим только квантор общности.

Остальные логические связки и квантор существования мы будем использовать, но будем понимать их как сокращение в соответствие со следующей таблицей:

Формула	Сокращение формулы
$\neg F \to G$	$F \vee G$
$\neg(F \to \neg G)$	$F \,\&\, G$
$\neg[(F \to G) \to \neg(G \to F)]$	$F \leftrightarrow G$
$\neg(\forall x)\neg F(x)$	$(\exists x)F(x)$

Сокращение числа связок и кванторов полезно и в другом отношении. Дело в том, что в дальнейшем надо будет часто проводить доказательство индукцией по построению формулы. А чем меньше логических операций, тем меньше случаев при рассмотрении шага индукции.

Итак, понятие формулы в этой главе мы воспринимаем в более узком смысле, нежели в главе 2. А понятие интерпретации, наоборот, расширим. Напомним, что интерпретация с областью M определялась как функция φ, удовлетворяющая следующим условиям:

1) если c – константа, то $\varphi(c) \in M$;

2) если f – функциональный символ с n аргументами, то $\varphi(f)$ – всюду определенная n-местная функция, заданная на M;

3) если r – предикатный символ с n аргументами, то $\varphi(r)$ – всюду определенный n-местный предикат, заданный на M.

Такое определение интерпретации преследовало цель изучить выразительные возможности языка логики предикатов. Оно позволяло ставить вопрос о том, выразим тот или иной предикат формулой данной сигнатуры. В этой главе предикат мы будем понимать как (всюду определенное) отображение r: $M^n \to \{0, 1\}$, где n – местность предиката, M^n – декартова степень множества M. Кроме того, к трем пунктам из определения интерпретации *добавим четвертый* пункт:

4) если x – переменная, то $\varphi(x) \in M$;

Основная цель подобного расширения дать *точное* определение значения истинности формулы при данной интерпретации. Ранее это было сделано только для логических связок, а в случае кванторов значение истинности воспринималось на интуитивном уровне. Введем несложное обозначение: через $I(\varphi, x)$ будем обозначать множество интерпретаций, совпадающих всюду, кроме возможно переменной x.

Из 1. 2 и 4 пунктов определения интерпретации следует, что $\varphi(t) \in M$ для любого терма t. Значение истинности формулы F при интерпретации φ индуктивно определяется следующим образом:

1) если $F = r(t_1, t_2, \ldots, t_n)$, то $\varphi(F) = \varphi(r)(\varphi(t_1), \varphi(t_2), \ldots, \varphi(t_n))$;

2) если $F = \neg G$, то $\varphi(F) = 1 - \varphi(G)$;

3) если $F = G \to H$, то $\varphi(F) = max\{1 - \varphi(G), \varphi(H)\}$;

4) если $F = (\forall x)G(x)$, то $\varphi(F) = 1$ в том и только в том случае, когда для любой $\varphi' \in I(\varphi, x)$ выполняется равенство $\varphi'(G) = 1$.

Пример. Пусть $\boldsymbol{F} = \{c, f(x)\}$, $\boldsymbol{R} = \{q(x)\}$, $\boldsymbol{V} = \{x_0, x_1, x_2, \ldots\}$. В качестве области интерпретации возьмем множество $M = \{0, 1, 2\}$. Интерпретацию определим следующей таблицей:

M	$\varphi(f)$	$\varphi(q)$
0	1	0
1	0	1
2	2	0

и равенствами $\varphi(c) = 2$ и $\varphi(x_i)$ = остаток от деления i на 3.

Рассмотрим формулу

$$F(x_6) = q(x_6) \to q[f(c)].$$

Вычислим последовательно: $\varphi(x_6) = 0$, $(\varphi q)(0) = 0$ и поэтому $\varphi[q(x_6)] = 0$. Далее считаем: $\varphi(c)= 2$, $(\varphi f)(2)= 2$, $(\varphi q)(2)= 0$. Следовательно, $\varphi[q(f(c))] = 0$. В итоге получим, что

$$\varphi[F(x_6)] = max\{1 - \varphi[q(x_6)], \varphi[q(f(c))]\} = 1.$$

Возьмем другую формулу

$$G = (\forall x_1)[\neg q(c) \to q(f(x_1))].$$

Мы знаем, что $\varphi(G) = 1$ тогда и только тогда, когда для любой интерпретации $\varphi' \in I(\varphi, x)$ выполняется равенство $\varphi'[\neg q(c) \to q(f(x_1))] = 1$. Надо рассмотреть различные значения для $\varphi'(x_1)$.

Случай 1: $\varphi'(x_1) = 0$. Тогда $\varphi'(\neg q(c)) = 1$, $\varphi'[q(f(x1))] = 1$ и поэтому $\varphi'[\neg q(c) \to q(f(x_1))] = 1$.

Случай 2: $\varphi'(x_1) = 1$. В этом случае имеем равенства $\varphi'(\neg q(c)) = 1$ и $\varphi'[q(f(x_1))] = 0$. Это означает, что $\varphi'[\neg q(c) \to q(f(x_1))] = 0$.

На основании последнего равенства заключаем, что $\varphi(G) = 0$. Третий случай: $\varphi'(x_1) = 2$ рассматривать не надо. □

Проведенное изменение понятия интерпретации позволит определить основные семантические понятия так, как это сделано в логике высказываний, а именно:

1) формула F называется *тождественно истинной*, если для любой интерпретации φ выполняется равенство $\varphi(F) = 1$;

2) формула F называется *равносильной* формуле G, если для любой интерпретации φ выполняется равенство $\varphi(F) = \varphi(G)$;

3) формула G называется *логическим следствием* формул $F_1, F_2, \ldots, F_k$ если для любой интерпретации φ из равенств $\varphi(F_1) = \varphi(F_2) = \ldots = \varphi(F_k) = 1$ следует равенство $\varphi(G) = 1$;

4) множество формул P называется *выполнимым*, если существует интерпретация φ такая, что $\varphi(P) = 1$, т. е. $\varphi(F) = 1$ для любой формулы F из P.

Тот факт, что формула G является тождественно истинной, будем обозначать через $\models G$. Для обозначения равносильности формул F и G оставим прежнее обозначение: $F \Leftrightarrow G$. Для множества формул P запись $P \models G$ будет означать, что формула G является логическим следствием множества формул P.

В этой главе нам придется довольно часто проводить доказательство индукцией по формуле, причем индукции будут разных «сортов». Наиболее частой будет проводиться индукция *по построению формулы*, т. е. по частично упорядоченному множеству подформул этой формулы. Иногда будет использоваться индукция *по сложности формулы*. В этом случае сложностью формулы считается число логических операций в записи этой формулы, т. е. суммарное число связок и кванторов.

§ 2. Теоремы о замене (о подстановке)

Параграф играет вспомогательную роль. Он содержит утверждения либо о совпадении значений интерпретации на «близких» выражениях (термах или формулах), либо о совпадении значений «близких» интерпретаций.

Условимся, что запись $t(x_1, x_2, \ldots, x_n)$ для терма или $F(x_1, x_2, \ldots, x_n)$ для формулы означает в первом случае, что $x_1, x_2, \ldots, x_n$ – все переменные терма t, а во втором случае, что $x_1, x_2, \ldots, x_n$ – все свободные переменные формулы F.

Начнем с простого утверждения.

Лемма 1. Если $t = t(x_1, x_2, \ldots, x_n)$ – терм, а φ и ψ – две интерпретации, совпадающие на функциональных символах из t и переменных $x_1, x_2, \ldots, x_n$, то $\varphi(t) = \psi(t)$.

Доказательство теоремы легко получается индукцией по построению терма и предоставляется читателю.

Теорема 3.1. Пусть $F = F(x_1, x_2, \ldots, x_n)$ – формула логики предикатов, φ и ψ – две интерпретации, совпадающие на функциональных и предикатных символах из F и переменных $x_1, x_2, \ldots, x_n$. Тогда $\varphi(F) = \psi(F)$.

Доказательство проведем индукцией по построению формулы F.

База индукции: F – атомарная формула, т. е. $F = r(t_1, t_2, \ldots, t_n)$. Тогда $\varphi(F) = \varphi(r)[\varphi(t_1), \varphi(t_2), \ldots, \varphi(t_n)]$ и $\psi(F) = \psi(r)[\psi(t_1), \psi(t_2), \ldots, \psi(t_n)]$. По условию теоремы выполняется равенство $\varphi(r) = \psi(r)$, а по лемме 1 – равенства $\varphi(t_i) = \psi(t_1)$ для $1 \le i \le n$. Следовательно, $\varphi(F) = \psi(F)$.

Шаг индукции: предположим, что для всех собственных подформул формулы F теорема доказана. Рассмотрим ряд случаев:

Случай 1: $F = \neg G$. Тогда $\varphi(F) = 1 - \varphi(G)$ и $\psi(F) = 1 - \psi(G)$. Но по предположению индукции $\varphi(G) = \psi(G)$. Следовательно, $\varphi(F) = \psi(F)$.

Случай 2: $F = G \to H$. В этом случае имеем, что $\varphi(F) = max\{1 - \varphi(G), \varphi(H)\}$ и $\psi(F) = max\{1 - \psi(G), \psi(H)\}$. В силу равенств $\varphi(G) = \psi(G)$ и $\varphi(H) = \psi(H)$ из предположения индукции, получаем заключение теоремы.

Рассмотрим наиболее сложный

Случай 3: $F = (\forall x)G(x, x_1, x_2, \ldots, x_n)$. Предположим, что $\varphi(F) = 1$. Это означает, что для любой интерпретации $\varphi' \in I(\varphi, x)$ выполняется равенство $\varphi'(G) = 1$. Возьмем произвольную интерпретацию $\psi' \in I(\psi, x)$ и выберем φ' из $I(\varphi, x)$ так, чтобы $\varphi'(x) = \psi'(x)$. Тогда интерпретации φ' и ψ' совпадают на всех функциональных и предикатных символах формулы G и ее свободных переменных $x, x_1, x_2, \ldots, x_n$. По предположению индукции имеем равенство $\varphi'(G) = \psi'(G)$. Следовательно, для любой интерпретации $\psi' \in I(\psi, x)$ выполняется равенство $\psi'(G) = 1$. Это означает, что $\psi(F) = 1$. Мы доказали, что если $\varphi(F) = 1$, то $\psi(F) = 1$. Обратное утверждение доказывается аналогичным образом. □

Теорема 1 подтверждает содержательно понятную мысль о том, что интерпретация формулы не зависит от интерпретации ее связанных переменных. В другом варианте ту же мысль выразим в виде теоремы 3.1′.

Теорема 3.1′. Пусть G – формула логики предикатов, а G' – формула, полученная из G заменой связанных переменных на новые (т. е. такие, которых нет в формуле G). Тогда для любой интерпретации φ выполняется равенство $\varphi(G) = \varphi(G')$.

Доказательство теоремы 1′ опустим, так как оно проводится аналогично доказательству теоремы 1.

Лемма 2. Пусть $t = t(x, x_1, x_2, \ldots, x_n)$ – некоторый терм, а $s = s(u, x_1, x_2, \ldots, x_n)$ – терм, полученный из t заменой переменной x

на терм u. Пусть, далее φ и ψ – две интерпретации, совпадающие на всех функциональных символах из t и переменных x_1, x_2, …, x_n (кроме, возможно, переменной x). Тогда если $\varphi(x) = \psi(u)$, то $\varphi(t) = \psi(s)$.

Доказательство леммы легко получается индукцией по построению терма t и предоставляется читателю.

Определение. Пусть F – формула со свободной переменной x (и возможно, другими свободными переменными), u - некоторый терм. Подстановка терма u вместо всех свободных вхождений переменной x называется *допустимой*, если ни одна переменная терма u в результате подстановки не окажется связанной.

Если, например,

$$F(x) = r(x) \,\&\, (\forall y)[s(x, y) \to t(x)],$$

то для допустимости подстановки терма u вместо переменной x необходимо, чтобы терм u не содержал переменной y.

Теорема 3.2. Пусть $F = F(x, x_1, x_2, \ldots, x_n)$ – формула, а u – терм, подстановка которого в формулу F вместо x допустима. Пусть, далее φ и ψ – две интерпретации, совпадающие на всех функциональных и предикатных символах из F и переменных $x_1, x_2, \ldots, x_n$. Тогда если $\varphi(x) = \psi(u)$, то $\varphi(F(x)) = \psi(F(u))$.

Доказательство проводится индукцией по построению формулы F. База индукции – это фактически лемма 2. Из трех случаев в шаге индукции рассмотрим только последний случай. Пусть $F(x) = (\forall y)G(x, y)$. (Разумеется, формулы F и G в качестве свободных переменных содержит еще и переменные $x_1, x_2, \ldots, x_n$.)

Предположим, что $\varphi(F(x)) = 1$. Это означает, что для любой интерпретации $\varphi' \in I(\varphi, y)$ выполняется равенство $\varphi'(G(x, y)) = 1$. Так как x – свободная переменная формулы F, $x \neq y$ и поэтому $\varphi'(x) = \varphi(x)$. Надо доказать, что для любой интерпретации $\psi' \in I(\psi, y)$ выполняется равенство $\psi'(G(u, y)) = 1$. Возьмем произвольную интерпретацию $\psi' \in I(\psi, y)$. Заметим, что $\psi'(u) = \psi(u)$, так как терм u не содержит переменной y в силу допустимости подстановки. Тогда интерпретации φ' и ψ' совпадают на всех функциональных и предикатных символах формулы F и свободных переменных $x_1, x_2, \ldots, x_n$ этой формулы и $\varphi'(x) = \psi'(u)$. По предположению индукции получаем равенство $\varphi'(G(x, y)) = \psi'(G(u, y))$. Но первое выражение равно 1. Следовательно, для любой интерпретации $\psi' \in I(\psi, x)$ выполняется равенство

$\psi'(G(u, y)) = 1$. Мы доказали, что если $\varphi(F(x)) = 1$, то $\psi(F(u)).= 1$. Обратное утверждение доказывается аналогичным образом. □

§ 3. Аксиоматизация логики предикатов

Данный параграф является центральным в этой главе. В нем будут даны понятия аксиомы, правила вывода, вывода и доказательства.

Начнем по порядку.

Определение. Пусть F, G и H – формулы логики предикатов. *Аксиомой логики предикатов* называется формула одного из следующих видов:

$A1$: $F \to (G \to F)$,

$A2$: $[F \to (G \to H)] \to [(F \to G) \to (F \to H)]$,

$A3$: $(\neg G \to \neg F) \to [(\neg G \to F) \to G]$,

$A4$: $(\forall x)(F \to G(x)) \to (F \to (\forall x)G(x))$, если формула F не содержит x в качестве свободной переменной,

$A5$: $(\forall x)F(x) \to F(t)$, если подстановка терма t вместо переменной x в формулу F допустима.

Подчеркнем, что аксиомой является любая формула указанных пяти видов (разумеется, при выполнении условий, указанных в $A4$ и $A5$). Например, формула $p(x) \to (q(x) \to p(x))$ – аксиома, полученная в соответствии с пунктом $A1$ определения. В этом случае мы будем говорить, что данная аксиома имеет *схему* $A1$. Множество аксиом бесконечно. Но число схем, по которым получаются аксиомы, всего пять.

Определение. Пусть F и G – формулы логики предикатов. *Правилами вывода в логике предикатов* называются следующие правила:

$$1)\quad \frac{F,\ F \to G}{G}, \qquad 2)\quad \frac{F(x)}{(\forall y)F(y)},$$

где x и y – переменные и подстановка y вместо x во втором правиле должна быть допустимой.

Первое правило называется *правилом отделения* (или правилом *modus ponens*). Мы его сокращенно будем обозначать через *MP*. Второе правило будем называть *правилом обобщения*, и для него будем использовать сокращение *GN*.

Сейчас мы сформулируем ряд определений, из которых основное внимание надо обратить на первое определение. После формулировки определений будет приведена значительная серия примеров.

Определение. *Выводом из множества формул* (или гипотез) P называется последовательность формул

$$F_1, F_2, \ldots, F_n,$$

каждая формула F_k которой удовлетворяет (хотя бы) одному из условий:

1) F_k принадлежит P,

2) F_k – аксиома,

3) F_k следует из предыдущих (в данной последовательности) формул по правилу MP,

4) F_k следует из предыдущей формулы по правилу GN, причем обобщаемая переменная не входит свободно в формулы из P, имеющиеся в этой последовательности.

Определение. Формула G называется *выводимой из множества формул* P, если существует вывод из P, последней формулой которого является G.

Обозначение: $P \vdash G$.

Определение. Вывод из пустого множества P называется *доказательством*.

Другими словами доказательство – последовательность формул, каждая формула которой удовлетворяет одному из условий 2 – 4 из определения вывода.

Определение. Формула G называется *доказуемой* (или *теоремой*), если существует доказательство, последней формулой которого является формула G.

Обозначение: $\vdash G$.

Перейдем, наконец, к примерам. *Для примеров будем использовать сквозную (внутри главы) нумерацию для последующих ссылок*. Последовательность формул в примерах нам будет удобно записывать не в строку, а в столбец. Будем, кроме того, указывать обоснования включения формулы в данный вывод.

Пример 1. $\vdash F \to F$.

1) $F \to ((F \to F) \to F)$,	$A1$
2) $[F \to ((F \to F) \to F)] \to$ $\to [(F \to (F \to F)) \to (F \to F)]$,	$A2$
3) $(F \to (F \to F)) \to (F \to F)$,	$MP(1, 2)$
4) $F \to (F \to F)$,	$A1$
5) $F \to F$.	$MP(3, 4)$

Пример 2. $\{F, \neg F\} \vdash G$.

1) $(\neg G \to \neg F) \to ((\neg G \to F) \to G)$,	$A3$
2) $\neg F \to (\neg G \to \neg F)$,	$A1$
3) $\neg F$,	гипотеза
4) $\neg G \to \neg F$	$MP(2, 3)$
5) $(\neg G \to F) \to G$,	$MP(1, 4)$

6) $F \to (\neg G \to F)$, $A1$
7) F, гипотеза
8) $\neg G \to F$, $MP(6, 7)$
9) G. $MP(5, 8)$

Пример 3. $\{F \to G,\ G \to H\} \vdash F \to H$.

1) $(G \to H) \to [F \to (G \to H)]$, $A1$
2) $G \to H$ гипотеза
3) $F \to (G \to H)$ $MP(1, 2)$
4) $[F \to (G \to H)] \to [(F \to G) \to (F \to H)]$, $A2$
5) $(F \to G) \to (F \to H)$, $F \to H$.
6) $F \to G$, гипотеза
7) $F \to H$. $MP(5, 6)$

Первые три схемы аксиом и правило *MP* аксиоматизируют логику высказываний. Так что три приведенных примера – фактически примеры из логики высказываний. Следующий – четвертый – пример будет примером из собственно логики предикатов. В нем мы используем достаточно очевидную идею о том, что последовательность формул, имеющуюся в уже приведенных примерах, повторять не надо. Кроме того, пример 4 обладает еще одной особенностью: он легко получается из теоремы о дедукции, обсуждаемой в следующем параграфе. «Беда», однако, в том, выбранный нами способ доказательства этой теоремы использует пример 4. Поэтому предлагаем читателю внимательно с ним ознакомиться.

Пример 4. $\vdash (\forall z)G(z) \to (\forall y)G(y)$. (Здесь предполагается, что подстановка y вместо z допустима.)

1) $(\forall z)G(z) \to G(u)$, $A5$
2) $(\forall y)[(\forall z)G(z) \to G(y)]$, $GN(1)$
3) $(\forall y)[(\forall z)G(z) \to G(y)] \to [(\forall z)G(z) \to (\forall y)G(y)]$, $A4$
4) $[(\forall z)G(z) \to (\forall y)G(y)]$. $MP(2, 3)$

§ 4. Теорема о дедукции

Из общих соображений ясно, что чем больше формул содержит множество гипотез, тем легче вывести из него данную формулу. В исчислении предикатов есть утверждение, которое позволяет увеличить число гипотез. Оно называется *теоремой о дедукции*.

Теорема 3.3. Пусть P – множество формул, F и G – некоторые формулы. Тогда

$$P \vdash F \to G \Leftrightarrow P \cup \{F\} \vdash G.$$

Доказательство. Необходимость очевидна.

Докажем достаточность. Пусть

$$G_1, G_2, \ldots, G_k, \ldots, G_l = G - \quad (1)$$

вывод формулы G из множества формул $P \cup \{F\}$. Если этот вывод не содержит формулы F, то он является выводом из множества P. Добавим в этом случае к (1) формулы $G \to (F \to G)$ и $F \to G$. Первая из добавленных формул является аксиомой, вторая следует из непосредственно предшествующих формул по правилу *MP*. Расширенная двумя формулами последовательность (1) является выводом из P формулы $F \to G$.

Будем считать, что (1) содержит формулу F.

Преобразуем последовательность (1) следующим образом. К каждой из формул последовательности (1) в качестве посылки импликации добавим формулу F. Получим последовательность формул

$$F \to G_1, F \to G_2, \ldots, F \to G_k, \ldots, F \to G_l = F \to G - \quad (2)$$

Последовательность (2) выводом из P, вообще говоря, не будет. Однако ее можно расширить до вывода из P. Для этого перед каждой из формул последовательности (2) запишем несколько формул. Какие из формул записывать перед формулой $F \to G_k$ зависит от «роли», которую формула G_k играет в выводе (1). Относительно этой роли рассмотрим ряд случаев:

Случай 1: G_k – аксиома или $G_k \in P$. Тогда непосредственно перед формулой $F \to G_k$ в последовательность (2) записываем формулы

1) G_k,

2) $G_k \to (F \to G_k)$.

Первая из формул добавляется на том основании, что она аксиома или гипотеза, а вторая – аксиома по схеме $A1$. Тогда формула $F \to G_k$ будет следовать из добавленных формул по правилу *MP*.

Случай 2: $G_k = F$. В этом случае перед формулой $F \to G_k = F \to F$ добавляется доказательство формулы $F \to F$ из примера 1.

Случай 3: G_k получается по правилу *MP* из формул G_i и G_j. В этом случае одна из формул G_i или G_j должна быть импликацией, посылкой которой является другая формула. Будем считать, что это формула G_j, т. е. $G_j = G_i \to G_k$. Формулы G_i и G_j в последовательности (1) расположены раньше формулы G_k, поэтому в последовательности (2) формулы $F \to G_i$ и $F \to G_j$ уже «обоснованы». Для удобства изложения часть последовательности (2) изобразим в виде следующей схемы:

. . .

1) $F \to G_i$,

. . .

2) $F \to (G_i \to G_k)$,

. . .

3) $[F \to (G_i \to G_k)] \to [(F \to G_i) \to (F \to G_k)]$,

4) $(F \to G_i) \to (F \to G_k)$,

5) $F \to G_k$,

. . . .

Здесь добавленные формулы обозначены как 3) и 4). Формула 3) является аксиомой по схеме 2. Формула 4) следует из 2) и 3) по правилу отделения. А формула 5) – следствие формул 1) и 4) по тому же правилу.

Случай 4: формула G_k получается из предыдущей формулы G_i по правилу обобщения. Это означает, что $G_i = G_i(x)$ и $G_k = (\forall y)G_i(y)$. Отметим, что переменная x не входит свободно в формулу F, так как F содержится в выводе (1).

Если формула F не содержит y в качестве свободной переменной, то ситуация довольно проста. Действительно, перед формулой $F \to G_k$ в последовательности (2) добавим две формулы и, как и в третьем случае, для удобства изложения часть последовательности (2) изобразим в виде схемы:

. . .

1) $F \to G_i(x)$,

. . .

2) $(\forall y)(F \to G_i(y))$,

3) $(\forall y)(F \to G_i(y)) \to (F \to (\forall y)G_i(y))$,

4) $F \to G_k = F \to (\forall y)G_i(y)$,

. . . .

Здесь добавлены формулы 2) и 3). Формула 2) получается по правилу обобщения из формулы 1). Формула 3) является аксиомой по схеме 4. Итоговая формула 4) следует из 2) и 3) по правилу отделения.

Предположим теперь, что формула F содержит y в качестве свободной переменной. В этом случае последовательность (2) расширяем так, как показано на следующей схеме:

. . .

1) $F \to G_i(x)$,

. . .

2) $(\forall z)(F \to G_i(z))$, (переменная z не содержится в $F \to G_i(x)$)

3) $(\forall z)(F \to G_i(z)) \to (F \to (\forall z)G_i(z))$,

4) $F \to G_k = F \to (\forall z)G_i(z)$,

. . .

8) $(\forall z)G_i(z) \to (\forall y)G_i(y)$ [формулы с 5 по 8 – формулы из примера 4],

. . .

15) $F \to (\forall y)G_i(y)$ [формулы с 9 по 15 – формулы из примера 3 с заменой G на $(\forall z)G_i(z)$ и H на $(\forall y)G_i(y)$],

. . . .

Здесь добавлены формулы с 2 по 14. Обоснования включения той или иной формулы делать не будем, так как эти обоснования фактически те же, что и для предыдущей схемы. Случай 4 полностью рассмотрен. □

В качестве примеров применения теоремы о дедукции докажем возможность введения и снятия двойного отрицания и закон контрапозиции. Закон контрапозиции утверждает, что формула $F \to G$ равносильна формуле $\neg G \to \neg F$. Для удобства дальнейшего применения продолжим нумерацию примеров из § 3.

Пример 5. $|- \neg\neg F \to F$ [или по теореме о дедукции $\neg\neg F |- F$].

1) $(\neg F \to \neg\neg\neg F) \to ((\neg F \to \neg F) \to F)$, $A3$

2) $\neg\neg F \to (\neg F \to \neg\neg F)$, $A1$

3) $\neg\neg F$, гипотеза

4) $\neg F \to \neg\neg\neg F$, $MP(2, 3)$

5) $(\neg F \to \neg F) \to F$, $MP(1, 4)$

6) $\neg F \to \neg F$, Пример 1

7) F. $MP(5, 6)$

Пример 6. $|- F \to \neg\neg F$ [или по теореме о дедукции $F |- \neg\neg F$].

1) $(\neg\neg\neg\neg F \to \neg F) \to ((\neg\neg\neg\neg F \to F) \to \neg\neg F)$, $A3$

2) $\neg\neg\neg\neg F \to \neg F$, Пример 6

3) $(\neg\neg\neg\neg F \to F) \to \neg\neg F$, $MP(1, 2)$

4) $F \to (\neg\neg\neg\neg F \to F)$, $A1$

5) F, гипотеза

6) $\neg\neg\neg\neg F \to F$, $MP(5, 4)$

7) $\neg\neg F$. $MP(6, 3)$

Пример 7. $\neg G \to \neg F |- F \to G$ [или по теореме о дедукции $\neg G \to \neg F, F |- G$].

1) $(\neg G \to \neg F) \to ((\neg G \to F) \to G)$, $A3$

2) $\neg G \to \neg F$,
3) $(\neg G \to F) \to G$, $MP(1, 2)$
4) $F \to (\neg G \to F)$, $A1$
5) F,
6) $\neg G \to F$, $MP(5, 4)$
7) G. $MP(6, 3)$

Пример 8. $F \to G \vdash \neg G \to \neg F$ [или по теореме о дедукции $F \to G, \neg G \vdash \neg F$].
1) $(\neg\neg F \to \neg G) \to ((\neg\neg F \to G) \to \neg F)$, $A3$
2) $\neg G \to (\neg\neg F \to \neg G)$, $A1$
3) $\neg G$, гипотеза
4) $\neg\neg F \to \neg G$, $MP(3, 2)$
5) $(\neg\neg F \to G) \to \neg F$, $MP(4, 1)$
6) $F \to G$, гипотеза
7) $\neg\neg F \to F$, Пример 5
8) $\neg\neg F \to G$, Пример 3
9) $\neg F$. $MP(8, 5)$

§ 5. Оправданность аксиоматизации

Основной целью данной главы является доказательство утверждения о том, что

$$P \vdash G \Leftrightarrow P \models G,$$

т. е. что выводимость формулы G из множества формул P (с помощью аксиом и правил вывода) равносильна логическому следствию формулы G из того же множества формул. Это утверждение называется *теоремой о полноте* исчисления предикатов. В этом параграфе мы докажем необходимость, т. е. утверждение о том, что если $P \vdash G$, то $P \models G$. Последнее утверждение носит свое название и называется *теоремой об оправданности аксиоматизации*. Доказательство достаточности теоремы о полноте является довольно сложным. В качестве промежуточного шага оно потребует доказательства так называемой *теоремы о непротиворечивости* (см. § 6) и будем приведено в § 7.

Теорема 3.4 (об оправданности аксиоматизации). Если $P \vdash G$, то $P \models G$.

Доказательство теоремы использует следующее вспомогательное утверждение.

Лемма. Все аксиомы исчисления предикатов тождественно истинны.

Доказательство. Тождественная истинность аксиом, полученных по схемам $A1$ – $A3$ доказывается простым построением таблиц истинности. Рассмотрим остальные схемы аксиом.

Схема A4: $(\forall x)(F \to G(x)) \to (F \to (\forall x)G(x))$. Предполагается, что формула F не содержит свободного вхождения переменной x. Рассмотрим произвольную интерпретацию φ с областью M. Предположим, что

$$\varphi[(\forall x)(F \to G(x))] = 1 \text{ и} \tag{1}$$

$$\varphi(F) = 1. \tag{2}$$

Надо доказать равенство

$$\varphi[(\forall x)G(x)] = 1, \tag{3}$$

т. е. надо доказать, что для любой интерпретации $\varphi' \in I(\varphi, x)$ выполняется равенство $\varphi'(G(x)) = 1$.

Из (1) следует, что $\varphi'(F \to G(x)) = 1$. Далее, интерпретации φ и φ' совпадают на всех сигнатурных символах и свободных переменных формулы F. По теореме 1 имеем равенство $\varphi(F) = \varphi'(F)$. Учитывая равенство (2), получаем, что $\varphi'(F) = 1$ $\varphi'(F \to G(x)) = 1$ и поэтому $\varphi'(G(x)) = 1$. Поскольку φ – произвольная интерпретация из $I(\varphi, x)$, получаем, что $\varphi[(\forall x)G(x)] = 1$. Итак, аксиомы по схеме 4 являются тождественно истинными.

Схема A5: $(\forall x)F(x) \to F(t)$. Предполагается, что подстановка терма t вместо переменной x допустима. Как и выше, возьмем произвольную интерпретацию φ с областью M, такую, что $\varphi[(\forall x)(F(x))] = 1$. Это означает, что для любой интерпретацию $\varphi' \in I(\varphi, x)$ выполняется равенство $\varphi'(F(x)) = 1$. Среди интерпретаций из $I(\varphi, x)$ выберем такую интерпретацию φ', что $\varphi'(x) = \varphi(t)$. Тогда по теореме 2 получим равенство $\varphi'(F(x)) = \varphi(F(t))$, т. е. равенство $\varphi(F(t)) = 1$. Тождественная истинность аксиом, полученных по схеме 5, доказана. □

Приступим непосредственно к *доказательству теоремы*.

Пусть $F_1, F_2, \ldots, F_k, \ldots, F_n = F$ – вывод формулы F из множества формул P. Рассмотрим произвольную интерпретацию φ такую, что $\varphi(P) = 1$. Индукцией по длине вывода докажем, что $\varphi(F_k) = 1$ для любого k.

База индукции: $k = 1$. Тогда F_k – аксиома или гипотеза. В первом случае равенство $\varphi(F_k) = 1$ следует из леммы, во втором – из условия $\varphi(P) = 1$.

Шаг индукции: предположим, что $\varphi(F_1) = \varphi(F_2) = \ldots = \varphi(F_{k-1}) = 1$. Докажем равенство $\varphi(F_k) = 1$. Если F_k – аксиома или гипотеза, то это равенство доказывается так же, как и в базе индукции. Пусть формула F_k получается из формул F_i и F_j по

правилу MP. Тогда $F_j = F_i \to F_k$ (или $F_i = F_j \to F_k$). Так как $i, j < k$, по предположению индукции имеем равенства $\varphi(F_i) = 1$ и $\varphi(F_j) = \varphi(F_i \to F_k) = 1$. Но тогда очевидно, что $\varphi(F_k) = 1$.

Осталось рассмотреть случай, когда F_k получается из F_i по правилу обобщения. В этом случае $F_i = F_i(x)$ и $F_k = (\forall y)F_i(y)$. Равенство

$$\varphi[(\forall y)F_i(y)] = 1,$$

которое надо доказать, означает, что для любой интерпретации $\varphi' \in I(\varphi, y)$ выполняется равенство $\varphi'(F_i(y)) = 1$. Возьмем φ' из $I(\varphi, y)$. Рассмотрим интерпретацию $\psi \in I(\varphi, x)$ такую, что $\psi(x) = \varphi'(y)$. Так же, как и раньше, доказываем, что $\psi(F_i(x)) = 1$. По теореме 2 получаем равенство $\psi(F_i(x)) = \varphi'(F_i(y))$. Следовательно, $\varphi'(F_i(y)) = 1$. □

§ 6. Теорема о непротиворечивости (леммы)

Начнем с основного определения.

Определение. Множество формул P называется противоречивым, если существует формула F такая, что $P \vdash F$ и $P \vdash \neg F$.

Пример 9. Убедимся в том, что множество формул

$$P = \{(\forall y)(F(y) \to G(a, b)), F(a), (\forall x)\neg G(a, x)\}$$

является противоречивым. Рассмотрим последовательности формул $D_1, \ldots, D_5$ и E_1, E_2, E_3:

$D_1 = (\forall y)(F(y) \to G(a, b)) \to (F(a) \to G(a, b))$,

$D_2 = (\forall y)(F(y) \to G(a, b))$,

$D_3 = F(a) \to G(a, b)$,

$D_4 = F(a)$,

$D_5 = G(a, b)$.

$E_1 = (\forall x)\neg G(a, x)$,

$E_2 = (\forall x)\neg G(a, x) \to \neg G(a, b)$,

$E_3 = \neg G(a, b)$.

Легко видеть, что эти последовательности являются выводами из P. Следовательно, $P \vdash G(a, b)$ и $P \vdash \neg\, G(a, b)$. Это означает, что множество P противоречиво. □

Укажем два полезных для дальнейшего свойства противоречивых (и непротиворечивы) множеств формул.

Свойство 1. Из примера 2 легко следует, что *противоречивое множество можно было определить как множество, из которого выводима любая формула*.

Свойство 2. Если из множества формул P не выводима некоторая формула G, то множество $P \cup \{\neg G\}$ непротиворечиво.

Действительно, пусть найдется формула F такая, что $P \cup \{\neg G\} \vdash F$ и $P \cup \{\neg G\} \vdash \neg F$. По теореме о дедукции получаем, что

$$P \vdash \neg G \to F \text{ и } P \vdash \neg G \to \neg F.$$

Из этих утверждений и аксиомы по схеме $A3$

$$(\neg G \to \neg F) \to ((\neg G \to F) \to G)$$

следует, что $P \vdash G$. Получили противоречие с условием. Следовательно, множество формул $P \cup \{\neg G\}$ непротиворечиво.

Целью данного и следующего параграфов является доказательство утверждения о том, что если множество формул непротиворечиво, то оно выполнимо. Это утверждение называется теоремой о непротиворечивости. При доказательстве этой теоремы нам будет удобно считать, что непротиворечивое множество формул P удовлетворяет следующим двум формальным условиям:

(α) для любой формулы F сигнатуры множества P либо $F \in P$, либо $\neg F \in P$;

(β) если множество P содержит формулу $\neg(\forall x)F(x)$, то найдется переменная w, которой нет в этой формуле, такая, что $\neg F(w) \in P$.

Хотя условия (α) и (β) были названы формальными, им можно придать некоторый содержательный смысл. Действительно, условие (α) означает, что множество P является максимальным среди всех непротиворечивых множеств данной сигнатуры. (Конечно, пока неясно, что такое множество существует.) Смысл условия (β) состоит в следующем. Формула $\neg(\forall x)F(x)$ в логике с обычным набором связок и кванторов равносильна формуле $(\exists x)\neg F(x)$. Условие (β) требует, чтобы утверждение о существовании элемента x, для которого формула $F(x)$ ложна, «реализовалось» на некоторой переменной w.

Данный параграф посвящен «обеспечению» этих условий. Доказательство самой теоремы о непротиворечивости будет изложено в следующем параграфе.

Лемма 1. Пусть P – непротиворечивое множество формул сигнатуры Σ. Тогда существует множество формул P' той же сигнатуры Σ, удовлетворяющее условиям:

1) $P \subseteq P'$,

2) P' непротиворечиво,

3) P' удовлетворяет условию (α).

Другими словами, всякое непротиворечивое множество формул содержится в максимальном непротиворечивом расширении.

Доказательство леммы 1 использует лемму Цорна. Напомним ее формулировку: если в частично упорядоченном множестве S каждая цепь имеет верхнюю грань, то множество S содержит максимальный элемент. Цепь – это непустое линейно упорядоченное подмножество множества S.

Пусть S – множество всех непротиворечивых расширений множества P, имеющих сигнатуру Σ, т. е.

$S = \{Q \mid Q$ имеет сигнатуру Σ, $P \subseteq Q$ и Q непротиворечиво$\}$.

Ясно, что множество S непусто, так как $P \in S$, что оно частично упорядочено отношением теоретико-множественного включения. Возьмем в S цепь

$$T = \{Q_i \mid i \in I\}.$$

В качестве «кандидатуры» на максимальную грань цепи рассмотрим множество

$$Q' = \cup\{Q_i \mid i \in I\}.$$

Надо убедиться в том, что $Q' \in S$. Очевидно, что Q' имеет сигнатуру Σ и что $P \subseteq Q'$. Предположим, что Q' противоречиво. Тогда существует формула F такая, что $Q' \vdash F$ и $Q' \vdash \neg F$. Рассмотрим вначале первый случай, т. е. что $Q' \vdash F$. Пусть

$$H_1, H_2, \ldots, H_n - \qquad (*)$$

вывод формулы F из множества Q'. В выводе выберем все гипотезы. Пусть это будут формулы

$$H_{i1}, H_{i2}, \ldots, H_{ik}.$$

Тогда в цепи T найдутся множества $Q_{i1}, Q_{i2}, \ldots, Q_{ik}$ такие, что $H_{i1} \in Q_{i1}$, $H_{i2} \in Q_{i2}$, …, $H_{ik} \in Q_{ik}$. Эти множества являются элементами цепи, поэтому они попарно сравнимы относительно теоретико-множественного включения. Следовательно, среди них есть наибольшее по включению множество. Пусть это будет Q_{ik}. Тогда вывод (*) будет выводом из Q_{ik}, т. е. $Q_{ik} \vdash F$. Аналогично доказывается существование элемента цепи Q_{il} такого, что $Q_{il} \vdash \neg F$. Воспользуемся еще раз тем, что Q_{ik} и Q_{il} – элементы цепи, и потому среди них есть наибольший. Пусть это будет Q_{il}. Но тогда $Q_{il} \vdash F$ и $Q_{il} \vdash \neg F$. Это противоречит тому, что $Q_{il} \in S$. Следовательно, Q' непротиворечиво, и поэтому $Q' \in S$. Условие леммы Цорна выполнено.

Заключение леммы Цорна говорит о том, что S содержит (хотя бы один) максимальный элемент. Обозначим его через P'. Тот факт, что $P \subseteq P'$ и что P' – непротиворечиво следует из принадлежности $P' \in S$. Осталось проверить, что P' удовлетво-

ряет условию (α). Предположим противное, пусть существует формула F такая, что $F \notin P'$ и $\neg F \notin P'$. Тогда по свойству 2 множества формул $P' \cup \{F\}$ и $P' \cup \{\neg F\}$ противоречивы, так как P' – максимальное непротиворечивое множество. Проанализируем противоречивость множества $P' \cup \{\neg F\}$. По определению противоречивого множества существует формула G такая, что

$$P' \vdash G \text{ и } P' \vdash \neg G.$$

Как уже неоднократно делалось, применение здесь теоремы о дедукции и аксиомы по схеме 3 дает, что $P' \vdash F$. Но тогда множество $P' \cup \{F\}$ не может быть противоречивым, так как $P' \cup \{F\} = P'$ и P' непротиворечиво. Следовательно, P' удовлетворяет условиям 1) – 3) из формулировки леммы 1. □

С «обеспечением» условия (β) дело обстоит несколько сложнее. Мы вначале в лемме 2 получим ослабленный вариант этого условия.

Лемма 2. Пусть P – непротиворечивое множество формул сигнатуры Σ. Тогда существует сигнатура Σ′ и множество формул P' сигнатуры Σ′, для которого выполняются условия:

1) $P \subseteq P'$,

2) P' непротиворечиво,

3) если формула $\neg(\forall x)F(x)$ принадлежит P, то найдется новая переменная w такая, что формула $\neg F(w)$ принадлежит P'.

Доказательство. Пусть

$$M = \{\neg(\forall x_i)F_i(x_i) \mid i \in I\} -$$

множество всех формул вида $\neg(\forall x)F(x)$, содержащихся в P. Для каждой такой формулы $\neg(\forall x_i)F_i(x_i)$ из M добавим к Σ новую (такую, которой нет в P) переменную w_i. Получим сигнатуру Σ′. В качестве P' рассмотрим множество

$$P' = P \cup \{\neg F_i(w_i) \mid i \in I\}.$$

Ясно, что P' удовлетворяет условиям 1) и 3) леммы 2. Надо доказать, что P' непротиворечиво. Предположим противное: пусть множество P' противоречиво. Тогда достаточно рассмотреть множество P', полученное из P' добавлением конечного множества формул вида $\neg(\forall x)F(x)$, а следовательно, и одной формулы такого вида.

Итак, пусть $\neg(\forall x)F(x) \in P$, $P' = P \cup \{\neg F(w)\}$ и множество P' противоречиво. По определению противоречивости существует формула G такая, что

$$P \cup \{\neg F(w)\} \vdash G \text{ и } P \cup \{\neg F(w)\} \vdash \neg G.$$

Тогда, как это часто демонстрировалось, $P \vdash F(w)$. Применение правила обобщения дает, что $P \vdash (\forall x)F(x)$. И в то же время, $\neg(\forall x)F(x) \in P$. Получили противоречие с условием о непротиворечивости множества P. Итак, множество P' удовлетворяет условиям 1) – 3) из формулировки леммы. □

Лемма 3. Для любого непротиворечивого множества P существует непротиворечивое расширение P', удовлетворяющее условиям (α) и (β).

Доказательство. Построим последовательность множеств формул $Q_0, Q_1, \ldots, Q_n, \ldots$ сигнатур $\Sigma_0, \Sigma_1, \ldots, \Sigma_n, \ldots$ соответственно следующим образом. Пусть $Q_0 = P$ и Σ_0 – сигнатура множества формул P. К множеству Q_0 применим лемму 1, получим непротиворечивое множество Q_1, удовлетворяющее условию (α), сигнатуры Σ_1, равной Σ_0. Далее к множеству Q_1 применим лемму 2, получим непротиворечивое множество Q_2 сигнатуры Σ_2. Затем снова применим лемму 1, но теперь уже к множеству Q_2 и т. д. В итоге возникнет последовательность непротиворечивых расширений множества P:

$$P = Q_0 \subseteq Q_1 \subseteq \ldots \subseteq Q_n \subseteq \ldots.$$

Каждое множество с четным номером удовлетворяет условию (α). А каждое множество с нечетным номером удовлетворяет условию 3) леммы 2, более слабому, чем условие (β). Для удобства изложения повторим третье условие леммы 2 в новых обозначениях: если $\neg(\forall x)F(x) \in Q_l$ и l – нечетно, то существует новая переменная w такая, что $\neg F(w) \in Q_{l+1}$. Условие 3 в такой «редакции» обозначим через (4).

В качестве P' рассмотрим объединение построенной последовательности множеств формул:

$$P' = Q_0 \cup Q_1 \cup \ldots \cup Q_n \cup \ldots .$$

а в качестве – объединение сигнатур:

$$\Sigma' = \Sigma_0 \cup \Sigma_1 \cup \ldots \cup \Sigma_n \cup \ldots .$$

Доказательство непротиворечивости множества P' в близкой ситуации (см. проверку условия леммы Цорна) мы уже проводили. Действительно, если найдется формула F такая, что $P' \vdash F$ и $P' \vdash \neg F$ то существует множество Q_n с тем же свойством. Это противоречит тому, что Q_n непротиворечиво.

Докажем, что множество P' удовлетворяет условию (α). Пусть F – формула сигнатуры Σ'. Так как Σ' – объединение возрастающей последовательности сигнатур существует сигнатура Σ_k такая, что F является формулой сигнатуры Σ_k. Можно считать, что k – четное число (иначе вместо n можно взять $k + 1$).

Но тогда Q_{k+1} удовлетворяет условию (α) и поэтому либо $F \in Q_{k+1}$, либо $\neg F \in Q_{k+1}$. По построению множества P' отсюда следует, что либо $F \in P'$, либо $\neg F \in P'$.

Установим, что множество P' удовлетворяет условию (β). Возьмем формулу $\neg(\forall x)F(x)$ из P'. Тогда найдется множество Q_l, содержащее эту формулу. Можно считать, что l – нечетное число (иначе вместо l можно взять $l + 1$). Тогда по условию (4) существует переменная w такая, что $\neg F(w) \in Q_{l+1}$. Так как Q_{l+1} содержится в P', получаем, что $\neg F(w) \in P'$. □

§ 7. Теорема о непротиворечивости (доказательство теоремы)

Напомним вначале формулировку теоремы о непротиворечивости.

Теорема 3.5. Если множество формул P непротиворечиво, то P выполнимо, т. е. существует интерпретация φ такая, что $\varphi(P) = 1$.

Доказательство. В силу леммы 3 можно считать, что P удовлетворяет условиям (α) и (β).

Сигнатуру множества P обозначим через Σ. В качестве области интерпретации возьмем множество термов T сигнатуры Σ. Функцию φ определим следующим образом:

1) $\varphi(x) = x$,

2) $\varphi(c) = c$,

3) $(\varphi f)(t_1, t_2, \ldots, t_n) = f(t_1, t_2, \ldots, t_n)$,

4) $(\varphi r)(t_1, t_2, \ldots, t_n) = 1 \Leftrightarrow r(t_1, t_2, \ldots, t_n) \in P$.

Здесь x – переменная, c – константа, f и r – соответственно n-местные символы функции и предиката.

Легко видеть, что из первых трех пунктов определения интерпретации следует, что $\varphi(t) = t$ для любого терма t.

Индукцией по построению интерпретации докажем, что

$$\varphi(F) = 1 \Leftrightarrow F \in P. \qquad (*)$$

Из этой эквиваленции очевидным образом следует выполнимость множества формул P.

База индукции: F – атомарная формула. Эквиваленция (*) следует из построения интерпретации (см. пункт 4), так как множество P одновременно не может содержать атомарную формулу и ее отрицание.

Шаг индукции: предположим, что для всех собственных подформул формулы F эквиваленция (*) доказана. Рассмотрим три случая.

Случай 1: $F = \neg G$. Тогда выполняются следующие эквиваленции:

$$\varphi(F) = 1 \Leftrightarrow \varphi(\neg G) = 1 \Leftrightarrow \varphi(G) = 0 \Leftrightarrow$$
$$G \notin P \Leftrightarrow \neg G \in P \Leftrightarrow F \in P.$$

В дополнительных комментариях нуждаются только две эквиваленции: $\varphi(G) = 0 \Leftrightarrow G \notin P$ – по предположению индукции, $G \notin P \Leftrightarrow \neg G \in P$ – по условию (α).

Случай 2: $F = G \to H$. Этот случай сложнее первого. Доказательство эквиваленции (*) разобьем на две части: «туда» и «обратно». А каждую из этих частей разобьем еще на две возможности.

($\Rightarrow$) Пусть $\varphi(G \to H) = 1$.

Возможность 1: $\varphi(H) = 1$. Тогда по предположению индукции $H \in P$, и $P \vdash G \to H$ (аксиома $A1$). Но в таком случае, $G \to H \in P$, так как P непротиворечиво и удовлетворяет условию (α).

Возможность 2: $\varphi(H) = 0$. Так как $\varphi(G \to H) = 1$, имеем равенство $\varphi(G) = 0$. По предположению индукции отсюда следует, что $G \notin P$. Следовательно, $\neg G \in P$ и множество $P \cup \{G\}$ противоречиво. В таком случае, из $P \cup \{G\}$ выводима любая формула (свойство 2 и предыдущего параграфа), в том числе формула H. По теореме о дедукции получаем, что $P \vdash G \to H$. Следовательно, $G \to H \in P$.

($\Leftarrow$) Пусть $G \to H \in P$.

Возможность 1: $H \in P$. Тогда по предположению индукции $\varphi(H) = 1$, и следовательно, $\varphi(G \to H) = 1$.

Возможность 2: $H \notin P$. Тогда $G \notin P$, поскольку в противном случае из условий $G \in P$ и $G \to H \in P$ следует, что $H \in P$. По предположению индукции получаем равенство $\varphi(G) = 0$, и поэтому $\varphi(G \to H) = 1$.

Случай 2 рассмотрен полностью.

Случай 3: $F = (\forall x)G(x)$. Этот случай, как и предыдущий, разобьем на две части: «туда» и «обратно».

($\Rightarrow$) Пусть $\varphi[(\forall x)G(x)] = 1$. Надо доказать, что $(\forall x)G(x) \in P$. Предположим противное: пусть $(\forall x)G(x) \notin P$. Тогда по свойствам (α) и (β) и найдется переменная w такая, что $\neg G(w) \in P$. Еще раз воспользуемся свойством (α), получим, что $G(w) \notin P$. По предположению индукции имеем равенство $\varphi(G(w)) = 0$. С

другой стороны, известно, что аксиома $(\forall x)G(x) \to G(w)$ тождественно истинна. Посылка этой аксиомы истинна при интерпретации φ. Следовательно, $\varphi(G(w)) = 1$. Полученное противоречие показывает, что формула $(\forall x)G(x)$ принадлежит P.

($\Leftarrow$)Пусть $(\forall x)G(x) \in P$. Надо доказать, что $\varphi[(\forall x)G(x)] = 1$. Как а в части «туда» здесь удобно рассуждать от противного. Предположим, что $\varphi[(\forall x)G(x)] = 0$. Тогда существует интерпретация $\varphi' \in I(\varphi, x)$ такая, что $\varphi'(G(x)) = 0$. Вспомним, наконец, что областью определения интерпретации φ является множество термов T. Следовательно, $\varphi'(x)$ – это некоторый терм. Обозначим его через t.

Если подстановка t вместо x в формуле $G(x)$ допустима, то все заканчивается довольно просто. Действительно, тогда по теореме 2 получаем равенство $\varphi'(G(x)) = \varphi[G(t)] = 0$. По предположению индукции имеем, что $G(t) \notin P$. Получаем противоречие с предположением $(\forall x)G(x) \in P$ и аксиомой $(\forall x)G(x) \to G(t)$ по схеме $A5$.

Предположим теперь, что подстановка t вместо x в формуле $G(x)$ недопустима. Заменим в формуле $G(x)$ все связанные переменные так, чтобы эта подстановка была допустима. Полученную формулу обозначим через $\underline{G}(x)$. Если $(\forall x)\underline{G}(x) \in P$, то рассуждаем так же, как и в предыдущем абзаце. Пусть $(\forall x)\underline{G}(x) \notin P$. Тогда по условиям (α) и (β) существует переменная w такая, что $\neg\underline{G}(w) \in P$. По предположению индукции имеем равенство $\varphi(\underline{G}(w)) = 0$. Однако $\varphi(\underline{G}(w)) = \varphi(G(w))$ по теореме 1 о замене. Следовательно, $\varphi(G(w)) = 0$ и $G(w) \notin P$. Как и выше, получаем противоречие с предположением $(\forall x)G(x) \in P$ и аксиомой $(\forall x)G(x) \to G(w)$ по схеме $A5$. Случай 3 полностью рассмотрен. □

§ 8. Теоремы о полноте и о компактности

Цель этого параграфа доказать совпадение семантического понятия логического следования с синтаксическим понятием выводимости. Соответствующее утверждение, как уже отмечалось, называется теоремой о полноте. Кроме того, в этом параграфе будет доказана «обещанная» в предыдущей главе теорема о компактности.

Теорема 3.6 (о полноте). Формула выводима из множества формул тогда и только тогда, когда она является логическим следствием этого множества.

Доказательство. Необходимость уже доказана (см. теорему об оправданности аксиоматизации). Достаточность докажем методом от противного. Предположим, что формула G является логическим следствием множества формул P и G не выводима из P. Тогда по свойству 2 непротиворечивых множеств (см. § 6) множество формул $P \cup \{\neg G\}$ непротиворечиво. Из теоремы о непротиворечивости следует, что оно выполнимо. Это означает, что существует интерпретация φ такая, $\varphi(P) = 1$ и $\varphi(\neg G) = 1$, т. е. $\varphi(P) = 1$ и $\varphi(G) = 0$. Получили противоречие с тем, что формула G является логическим следствием множества P. □

Теорема 3.7 (о компактности). Если формула F является логическим следствием бесконечного множества формул P, то она является логическим следствием некоторого конечного подмножества P_0 множества P.

Доказательство. Пусть формула F является логическим следствием бесконечного множества формул P. Тогда по теореме о полноте формула F выводима из P. Это означает, что существует вывод из P, последней формулой которого является формула F:

$$F_1, F_2, \ldots, F_n = F.$$

Обозначим через P_0 множество тех формул из P, которые встречаются в выводе. Ясно, что P_0 – конечное множество и что этот вывод является выводом из множества P_0. Применим теорему о полноте еще раз, получим, что формула F является логическим следствием множества P_0. □

§ 9. Независимость аксиом

Как мы уже отмечали, есть некоторая математическая традиция, идущая еще от древних греков, согласно которой аксиом в той или иной теории должно быть как можно меньше. В идеальном варианте – наименее возможное число. Для этого обычно ставится следующий вопрос: будет ли данная аксиома следствием других аксиом? В случае положительно ответа аксиому можно удалить из списка аксиом. В случае отрицательного ответа ее удалить нельзя и аксиома называется *независимой* (от остальных аксиом). Здесь можно сослаться на известную ситуацию в геометрии, а именно на вопрос о том следует ли пятый постулат Евклида из первых четырех. Как показали Лобачевский и Бойяи, пятый постулат независим от остальных аксиом.

В этом параграфе вопрос о независимости мы поставим для схем аксиом в следующей формулировке: можно ли данную схему аксиом получить из остальных схем с помощью правил

вывода? Мы решим этот вопрос только для логики высказываний. Так что схемами аксиом будут первые три, а правило вывода будет одно – модус поненс. Как мы увидим, во всех трех случаях имеет место независимость.

Теорема 3.7. Схема $A1$ независима от схем $A2$ и $A3$.

Доказательство. Возьмем трехэлементное множество $M = \{0, 1, 2\}$. Элементы этого множества будем обозначать заглавными буквами латиницы. На множестве M введем на нем две операции: унарную $Y = \neg X$ (отрицание) и бинарную $Z = X \to Y$ (импликацию). Отрицание определяется таблицей 3.1, импликация – таблицей 3.2.

Таблица 3.1

X	$\neg X$
0	1
1	1
2	0

Таблица 3. 2

X	Y	$X \to Y$
0	0	0
0	1	2
0	2	0
1	0	2
1	1	2
1	2	0
2	0	0
2	1	0
2	2	0

Определение. *Интерпретацией* называется функция φ: $\boldsymbol{A} \to \{0, 1, 2\}$, где $\boldsymbol{A}$ – множество атомарных формул логики высказываний.

С помощью таблиц 1 и 2 интерпретацию можно расширить на все множество формул логики высказываний.

Определение. Формула F логики высказываний называется *выделенной*, если для любой интерпретации φ выполняется равенство $\varphi(F) = 0$.

Нетрудно проверить, что любая аксиома схемы $A2$ или $A3$ является выделенной. Для аксиомы схемы $A3$ это сделано в таблице 3.3. Для схемы $A2$ соответствующую проверку предлагается сделать читателю.

Таблица 3.3

F	G	$\neg G \to \neg F$	$\neg G \to F$	$(\neg G \to F) \to G$	$(\neg G \to \neg F) \to [(\neg G \to F) \to G]$
0	0	2	2	0	0
0	1	2	2	0	0
0	2	2	0	0	0
1	0	2	2	0	0
1	1	2	2	0	0
1	2	2	2	0	0
2	0	2	0	0	0
2	1	2	0	2	0

2	2	0	0	2	0

Убедимся в том, что правило *MP* сохраняет выделенность. Действительно, если $\varphi(F) = 0$ и $\varphi(F \to G) = 0$, то, как следует из таблицы 0.2, выполняется равенство $\varphi(G) = 0$. Следовательно, любая формула, которая получается из аксиом схем $A2$ и $A3$ с помощью правила *MP* является выделенной.

Рассмотрим теперь аксиому $F = X \to (Y \to X)$ схемы $A1$ и интерпретацию $\varphi(X) = 1$, $\varphi(Y) = 2$. Легко видеть, что $\varphi(F) = 2$. Аксиома F выделенной не является, а поэтому не может быть получена из аксиом схем $A2$ и $A3$ с помощью правила *MP*. □

Теорема 3.8. Схема $A2$ независима от схем $A1$ и $A3$.

Доказательство проводится аналогично доказательству предыдущей леммы. Только здесь функции $Y = \neg X$ (отрицание) и $Z = X \to Y$(импликацию) определим по-другому (см. таблицы 3.4 и 3.5)

Таблица 3.4

X	$\neg X$
0	1
1	0
2	1

Таблица 3. 5

X	Y	$X \to Y$
0	0	0
0	1	2
0	2	1
1	0	0
1	1	2
1	2	0
2	0	0
2	1	0
2	2	0

Нетрудно проверить, что любая аксиома схем $A1$ и $A3$ является выделенной, и что правило *MP* сохраняет выделенность. С другой стороны, аксиома $F = [X \to (Y \to Z)] \to [(X \to Y) \to (X \to Z)]$ схемы $A2$ при интерпретации $\varphi(X) = 0$, $\varphi(Y) = 0$, $\varphi(Z) = 1$ получает значение 2. Следовательно, F не может быть получена из аксиом схем $A1$ и $A3$ с помощью *MP*. □

Теорема 3.9. Схема $A3$ независима от схем $A1$ и $A2$.

Доказательство будет проведено методом, отличным от метода доказательства лемм 1 и 2. Для формулы F логики высказываний через $h(F)$ обозначим формулу, полученную из F удалением всех знаков отрицания. Формулу F будем называть *выделенной*, если $h(F)$ – тождественно истинная формула логики высказываний.

Поскольку $h(F \to G) = h(F) \to h(G)$, все аксиомы схем $A1$ и $A2$ являются выделенными. Нетрудно видеть также, что правило *MP* сохраняет выделенность. С другой стороны, аксиома $F =$

$(\neg Y \to \neg X) \to ((\neg Y \to X) \to X)$ схемы $A3$ не является выделенной, так как формула $h(F) = (Y \to X) \to ((Y \to X) \to X)$ не является тождественно истинной. Действительно, если $\varphi(X) = \varphi(Y) = 0$, то $\varphi(h(F)) = 0$. Это означает, что формула F не может быть получена из аксиом схем $A1$ и $A3$ с помощью MP. □

§ 9. Другие аксиоматизации

Рассмотренная в предыдущих параграфах система аксиом и правил вывода взята из книги [Мен]. Будем называть эту систему *системой Мендельсона*. Кроме этой аксиоматизации в литературе изучались и другие аксиоматизации.

В этом параграфе мы приведем некоторые их них, правда, только в случае логики высказываний.

1. *Система Мендельсона с правилом подстановки* [см. Мен]. В этой системе вместо схем аксиом берутся конкретные формулы-аксиомы и к правилу модус поненс добавляется правило подстановки, позволяющее в аксиому подставлять вместо атомарной формулы любую формулу.

2. *Система Гильберта и Аккермана* [см. ГБ или Мен]. Основные связки: $\neg$ и $\vee$. Формула $F \to G$ понимается как сокращение формулы $\neg F \vee G$. Система содержит четыре схемы аксиом:

1) $F \vee F \to F$,
2) $F \to F \vee G$,
3) $F \vee G \to G \vee F$,
4) $(G \to H) \to (F \vee G \to F \vee H)$.

Единственное правило вывода – модус поненс.

3. *Система Клини* [см. Кл, стр. 48]. Основные связки: $\neg$, $\&$, $\vee$ и $\to$. Система содержит 10 схем аксиом:

1) $F \to (G \to F)$,
2) $(F \to (G \to H)) \to [(F \to G) \to (F \to H)]$,
3) $F \,\&\, G \to F$,
4) $F \,\&\, G \to G$,
5) $F \to (G \to F \,\&\, G)$,
6) $F \to F \vee G$,
7) $G \to F \vee G$,
8) $(F \to H) \to [(G \to H) \to (F \vee G \to H)]$,
9) $(F \to G) \to [(F \to \neg G) \to \neg F]$,
10) $\neg\neg F \to F$.

Единственное правило вывода – модус поненс.

4. *Система Черча* [см. Ч, стр. 112 – 113]. Основные связки: $\neg$ и $\to$. Система содержит три схемы аксиом:

1) $F \to (G \to F)$,

2) $(F \to (G \to H)) \to [(F \to G) \to (F \to H)]$,

3) $(\neg G \to \neg F) \to (F \to G)$.

Единственное правило вывода – модус поненс.

Все рассмотренные выше варианты аксиоматизаций логики высказываний (т. е. системы аксиом и правил вывода) относятся к так называемым *системам гильбертовского типа*. В этих системах определяется некоторое множество схем аксиом (или конкретных аксиом) и множество правил вывода. Множество схем аксиом и правил вывода должно быть конечным, а множество аксиом – рекурсивным. Вывод из множества формул P определяется как последовательность формул, каждая формула которой принадлежит P или является аксиомой, или следует из предыдущих формул по одному из правил вывода. Формула G называется выводимой из P, если существует вывод из P, последней формулой которого является G. С примером аксиоматизации, которая не относится к гильбертовскому типу, можно познакомиться по книгам [ЕП] или [СО].

Приведем примеры на выводимость в разных системах.

Пример 10. Доказать, что $\{F \to G,\ G \to H\} \vdash F \to H$ в системе Гильберта и Аккермана.

1) $G \to H$, гипотеза

2) $(G \to H) \to (\neg F \vee G \to \neg F \vee H)$, $A4$

3) $(\neg F \vee G \to \neg F \vee H)$ есть $(F \to G) \to (F \to H)$, $MP(1, 2)$

4) $F \to G$, гипотеза

5) $F \to H$. $MP(3, 4)$

Пример 11. Доказать, что $\vdash F \to F$ в системе Гильберта и Аккермана.

1) $F \to F \vee F$, $A2$

2) $F \vee F \to F$, $A1$

3) $F \to F$. Пример 10

Пример 12. Доказать, что $\vdash F \to F$ в системе Черча. Доказательство то же самое, что и в системе Мендельсона (см. пример 1).

Пример 13. Доказать, что $\neg F \vdash F \to G$ в системе Черча.

1) $\neg F \to (\neg G \to \neg F)$, $A1$

2) $\neg F$,

3) $\neg G \to \neg F$, $MP(1, 2)$
4) $(\neg G \to \neg F) \to (F \to G)$, $A3$
5) $F \to G$. $MP(3, 4)$

Пример 14. Доказать, что $\{F \to (G \to H), F \,\&\, G\} \vdash H$ в системе Клини.

1) $F \,\&\, G \to F$, $A3$
2) $F \,\&\, G$,
3) F, $MP(1, 2)$
4) $F \to (G \to H)$
5) $G \to H$ $MP(3, 4)$
6) $F \,\&\, G \to G$, $A4$
7) G, $MP(2, 6)$
8) H. $MP(5, 7)$

Для всех рассмотренных выше систем справедлива теорема о полноте: формула G выводима (в данной системе) из множества формул P тогда и только тогда, когда G является логическим следствием этого множества формул.

Задачи

Если в формулировке задач не указана система аксиом и правил вывода, то предполагается, что это система Мендельсона, т. е. та, которая введена в § 3.

1. Доказать, что $\neg F \to F \vdash F$.

2. Доказать, что $F \to G$, $\neg F \to G \vdash G$ (разбор случаев).

3. Напомним, что формула $F \vee G$ понимается как сокращение формулы $\neg F \to G$. Доказать, что $F \vee G \vdash G \vee F$ (коммутативность дизъюнкции).

4. Напомним, что формула $F \,\&\, G$ понимается как сокращение формулы $\neg(F \to \neg G)$. Доказать, что $F \,\&\, G \vdash G \,\&\, F$ (коммутативность дизъюнкции).

5. Формула $(\exists x)F(x)$ была введена как сокращение формулы $\neg(\forall x)\neg F(x)$. Доказать, что $\neg(\exists x)\neg F(x) \vdash (\forall x)F(x)$.

6. Доказать, что $(\forall x)(F(x) \to G(x)) \vdash (\forall x)F(x) \to (\forall x)G(x)$.

7. Доказать, что $(\forall x)(F(x) \to G(x)) \vdash (\exists x)F(x) \to (\exists x)G(x)$..

8. В системе Гильберта и Аккермана доказать, что $\vdash F \to (G \to H)$ или, используя сокращение, доказать, что $\vdash F \to (\neg G \vee H)$.

9. В системе Черча доказать, что $\neg\neg F \vdash F$.

10. В системе Клини доказать, что $\{F \to G, \neg G\} \vdash \neg F$.

11. В системе Клини доказать, что $\{F, \neg F\} \vdash G$.

12. В системе Клини доказать, что $\{F \to (G \to H), F \,\&G\} \vdash H$.

13. В системе Клини доказать, что $\{F \to G, G \to H, F \vee G\} \vdash H$.

14. В системе Клини доказать, что $\{\neg G \to \neg F, \neg G \to F\} \vdash G$.

Решения

1. Приведем соответствующую последовательность формул.

1) $(\neg F \to \neg F) \to [(\neg F \to F) \to F]$, $A3$
2) $\neg F \to \neg F$, Пример 1
3) $(\neg F \to F) \to F$, $MP(1, 2)$
4) $\neg F \to F$, гипотеза
5) F. $MP(3, 4)$

2. Приведем соответствующую последовательность формул.

1) $F \to G$, гипотеза
2) $\neg G \to \neg F$, Пример 9
3) $\neg F \to G$, гипотеза
4) $\neg G \to \neg\neg F$, Пример 9
5) $(\neg G \to \neg\neg F) \to ((\neg G \to \neg F) \to G)$, $A3$
6) $(\neg G \to \neg F) \to G$, $MP(4, 5)$
7) G. $MP(2, 6)$

3. Фактически надо доказать, что $\neg F \to G \vdash \neg G \to F$, или с учетом теоремы о дедукции, что $\neg F \to G, \neg G \vdash F$. Приведем соответствующую последовательность формул.

1) $(\neg F \to \neg G) \to ((\neg F \to G) \to F)$, $A3$
2) $\neg G$, гипотеза
3) $\neg G \to (\neg F \to \neg G)$, $A1$
4) $\neg F \to \neg G$, $MP(2, 3)$
5) $(\neg F \to G) \to F$, $MP(4, 1)$
6) $\neg F \to G$, гипотеза
7) F. $MP(6, 5)$

4. С учетом закона контрапозиции и теоремы о дедукции задачу можно сформулировать следующим образом:

$$G \to \neg F, G \vdash \neg F.$$

Тогда решение становится очевидным. Достаточно одного применения правила *MP*.

5. С учетом сокращения надо доказать, что $\neg\neg(\forall x)\neg\neg F(x) \vdash (\forall x)F(x)$. Приведем соответствующую последовательность формул.

1) $\neg\neg(\forall x)\neg\neg F(x)$, гипотеза
2) $(\forall x)\neg\neg F(x)$, Пример 5
3) $(\forall x)\neg\neg F(x) \to \neg\neg F(u)$, $A5$
4) $\neg\neg F(u)$, $MP(2, 3)$
5) $F(u)$, Пример 5
6) $(\forall x)F(x)$. $GN(5)$

6. С учетом теоремы о дедукции надо доказать, что $(\forall x)(F(x) \to G(x))$, $(\forall x)F(x) \vdash (\forall x)G(x)$. Приведем соответствующую последовательность формул:

1) $(\forall x)(F(x) \to G(x)) \to (F(u) \to G(u))$, $A5$
2) $(\forall x)(F(x) \to G(x))$, гипотеза
3) $F(u) \to G(u)$, $MP(2, 1)$
4) $(\forall x)F(x) \to F(u)$, $A5$
5) $(\forall x)F(x)$, гипотеза
6) $F(u)$, $MP(5,4)$
7) $G(u)$, $MP(6, 3)$
8) $(\forall x)G(x)$ $GN(7)$

7. С учетом способа введения квантора существования и теоремы о дедукции, задачу можно записать следующим образом: $(\forall x)(F(x) \to G(x))$, $\neg(\forall x)\neg F(x) \vdash \neg(\forall x)\neg G(x)$. Далее, используя закон контрапозиции, задачу можно переписать так: $(\forall x)(F(x) \to G(x))$, $(\forall x)\neg G(x) \vdash (\forall x)\neg F(x)$. Приведем соответствующую последовательность формул:

1) $(\forall x)(F(x) \to G(x)) \to (F(u) \to G(u))$, $A5$
2) $(\forall x)(F(x) \to G(x))$, гипотеза
3) $F(u) \to G(u)$, $MP(2, 1)$
4) $\neg G(u) \to \neg F(u)$, Пример 5
5) $(\forall x)\neg G(x) \to \neg G(u)$, $A5$
6) $(\forall x)\neg G(x)$, гипотеза
7) $\neg G(u)$, $MP(6, 5)$
8) $\neg F(u)$, $MP(4, 7)$
9) $(\forall x)\neg F(x)$. $GN(8)$

8. Приведем соответствующую последовательность формул:

1) $F \to F \vee \neg G$, $A2$
2) $F \vee \neg G \to \neg G \vee F$, $A3$

3) $F \to \neg G \vee F$. Пример 10

9. Приведем соответствующую последовательность формул:

1) $\neg\neg F$, гипотеза
2) $\neg F \to \neg\neg\neg F$, Пример 13
3) $(\neg F \to \neg\neg\neg F) \to (\neg\neg F \to F)$, $A3$
4) $\neg\neg F \to F$, $MP(2, 3)$
4) F. $MP(1, 4)$

10. Приведем соответствующую последовательность формул:

1) $(F \to G) \to ((F \to \neg G) \to \neg F)$, $A9$
2) $F \to G$, гипотеза
3) $(F \to \neg G) \to \neg F$, $MP(1, 2)$
4) $\neg G \to (F \to \neg G)$, $A1$
5) $\neg G$, гипотеза
6) $F \to \neg G$, $MP(5, 4)$
7) $\neg F$. $MP(6, 3)$

11. Приведем соответствующую последовательность формул:

1) $(\neg G \to F) \to ((\neg G \to \neg F) \to \neg\neg G)$, $A9$
2) $F \to (\neg G \to F)$, $A1$
3) F, гипотеза
4) $\neg G \to F$, $MP(2, 3)$
5) $(\neg G \to \neg F) \to \neg\neg G$, $MP(4, 1)$
6) $\neg F \to (\neg G \to \neg F)$, $A1$
7) $\neg F$, гипотеза
8) $\neg G \to \neg F$ $MP(7, 6)$
9) $\neg\neg G$, $MP(8, 5)$
10) $\neg\neg G \to G$, $A10$
11) G. $MP(9, 10)$

Глава 4. Метод резолюций

Глава посвящена рассмотрению метода доказательства того, что формула G является логическим следствием формул F_1, F_2, …, F_k. Этот метод называется *методом резолюций*. Отметим, что задача о логическом следствии сводится к задаче о выполнимости. Действительно, формула G есть логическое следствие формул F_1, F_2, …, F_k тогда и только тогда, когда множество формул $\{F_1, F_2, \ldots, F_k, \neg G\}$ невыполнимо. Метод резолюций, если говорить более точно, устанавливает невыполнимость. Это первая особенность метода. Вторая особенность метода состоит в том, что он оперирует не с произвольными формулами, а с дизъюнктами (или элементарными дизъюнкциями).

§1. Метод резолюций в логике высказываний

Рассмотрим вначале логику высказываний. Напомним, что литералом мы назвали атомарную формулу или ее отрицание, дизъюнктом – дизъюнкцию литералов. Дизъюнкт может состоять из одного литерала. *На дизъюнкт мы иногда будем смотреть, как на множество литералов,* т.е. не будем различать дизъюнкты, которые получаются один из другого с помощью коммутативности и ассоциативности дизъюнкции, а также идемпотентности. Последнее означает, например, что дизъюнкты $X \vee \neg Y \vee X$ и $X \vee \neg Y$ равны. Нам понадобится особый дизъюнкт – *пустой*, т.е. дизъюнкт, не содержащий литералов. Его мы будем обозначать символом $\square$. Будем считать, что пустой дизъюнкт ложен при любой интерпретации. Это означает, что формула $F \& \square$ равносильна $\square$, а формула $F \vee \square$ равносильна F. Пустой дизъюнкт есть фактически то же самое, что и атомарная формула 0, но в контексте метода резолюций принято использовать $\square$.

Определение. Литералы L и $\neg L$ называются *противоположными*.

Метод резолюций в логике высказываний основан на правиле резолюций.

Определение. *Правилом резолюций в логике высказываний* называется следующее правило: из дизъюнктов $X \vee F$ и $\neg X \vee G$ выводим дизъюнкт $F \vee G$.

Например, из дизъюнктов $\neg X \vee Y \vee Z$ и $X \vee \neg Y$ выводим дизъюнкт $Y \vee Z \vee \neg Y$. Обратим внимание на то, что в первых двух дизъюнктах есть еще одна пара противоположных литералов. Условимся, что можно применять правило резолюций не обязательно к самым левым литералам (поскольку мы не различаем дизъюнкты, отличающиеся порядком записи литералов). Тогда правило резолюций, примененное к Y и $\neg Y$ в дизъюнктах $\neg X \vee Y \vee Z$ и $X \vee \neg Y$, даст $\neg X \vee Z \vee X$.

Определение. Пусть S – множество дизъюнктов. *Выводом* из S называется последовательность дизъюнктов

$$D_1, D_2, \ldots, D_n$$

такая, что каждый дизъюнкт этой последовательности принадлежит S или следует из предыдущих по правилу резолюций. Дизъюнкт D *выводим из* S, если существует вывод из S, последним дизъюнктом которого является D.

Например, если $S = \{\neg X \vee Y \vee Z,\ \neg Y \vee U,\ X\}$, то последовательность $D_1 = \neg X \vee Y \vee Z$, $D_2 = \neg Y \vee U$, $D_3 = \neg X \vee Z \vee U$, $D_4 = X$, $D_5 = Z \vee U$ – вывод из S. Дизъюнкт $Z \vee U$ выводим из S.

Применение метода резолюций основано на следующем утверждении, которое называется теоремой о полноте метода резолюций.

Теорема 4.1. Множество дизъюнктов логики высказываний S невыполнимо тогда и только тогда, когда из S выводим пустой дизъюнкт.

Доказательство. Отметим, что правило резолюций сохраняет истинность. Это означает, что если $\varphi(\neg X \vee F) = 1$ и $\varphi(X \vee G) = 1$ для некоторой интерпретации φ, то $\varphi(F \vee G) = 1$. Действительно, пусть $\varphi(\neg X \vee F) = 1$ и $\varphi(X \vee G) = 1$. Тогда если $\varphi(F) = 1$, то и $\varphi(F \vee G) = 1$. Если же $\varphi(F) = 0$, то $\varphi(\neg X) = 1$, поскольку $\varphi(\neg X \vee F) = 1$. Но в этом случае $\varphi(X) = 0$ и $\varphi(G) = 1$, так как $\varphi(X \vee G) = 1$. Если же $\varphi(G) = 1$, то и $\varphi(F \vee G) = 1$.

Докажем вначале достаточность.

Пусть из S выводим пустой дизъюнкт. Предположим противное: множество S выполнимо, т.е. существует интерпретация ψ, при которой все дизъюнкты из S истинны. Выводимость пустого дизъюнкта из S означает, что существует последовательность дизъюнктов $D_1, \ldots, D_n = \square$, каждый дизъюнкт которой

принадлежит S или получается из предыдущих по правилу резолюций. Если дизъюнкт D_j из этой последовательности принадлежит S, то по предположению $\psi(D_j) = 1$. Если же он получается из предыдущих по правилу резолюций, то также $\psi(D_j) = 1$, поскольку правило резолюций сохраняет истинность. При $i = n$ получаем, что $\psi(\square) = 1$. Противоречие показывает, что предположение о выполнимости множества S – ложное предположение. Следовательно, S невыполнимо. Достаточность доказана.

Докажем необходимость. Доказательство проведем индукцией по следующему параметру $d(S)$: это сумма числа вхождений литералов в дизъюнкты из S минус число дизъюнктов.

Пусть множество дизъюнктов S невыполнимо. Если пустой дизъюнкт принадлежит S, то он выводим из S (вывод в этом случае состоит из одного пустого дизъюнкта) и необходимость теоремы доказана. Будем считать в силу этого, что $\square \notin S$. При этом предположении каждый дизъюнкт содержит хотя бы один литерал и поэтому $d(S) \geq 1$.

База индукции: $d(S) = 1$. Если $d(S) = 1$, то все дизъюнкты состоят из одного литерала. Поскольку множество S невыполнимо, то в нем должна найтись пара противоположных литералов X и $\neg X$. В таком случае пустой дизъюнкт выводим из S, соответствующий вывод содержит три дизъюнкта: X, $\neg X$, $\square$.

Шаг индукции: $d(S) > 1$. Предположим, что для любого множества дизъюнктов T такого, что $d(\mathrm{T}) < d(S)$ необходимость теоремы доказана. Пусть

$$S = \{D_1, D_2, \ldots, D_{k-1}, D_k\}.$$

Так как $d(S) > 1$, то в S существует хотя бы один неодноэлементный дизъюнкт. Будем считать, что это дизъюнкт D_k, т.е. $D_k = L \vee D_k'$, где L – литерал и $D_k' \neq \square$. Наряду с множеством дизъюнктов S рассмотрим еще два множества дизъюнктов

$$S_1 = \{D_1, D_2, \ldots, D_{k-1}, L\},$$
$$S_2 = \{D_1, D_2, \ldots, D_{k-1}, D_k'\}.$$

Ясно, что S_1 и S_2 невыполнимы и что $d(S_1) < d(S)$ и $d(S_2) < d(S)$. По предположению индукции из S_1 и S_2 выводим пустой дизъюнкт. Пусть

$A_1, A_2, \ldots, A_i, \ldots, A_{l-1}, A_l = \square$ – (1)

вывод пустого дизъюнкта из S_1 и

$B_1, B_2, \ldots, B_j, \ldots B_{m-1}, B_m = \square$ – (2)

вывод пустого дизъюнкта из S_2. Если в первом выводе не содержится дизъюнкта L, то эта последовательность дизъюнктов будет выводом из S и необходимость теоремы доказана. Бу-

дем считать, что L содержится в первом выводе, пусть $A_i = L$. Аналогично предполагаем, что $B_j = D_k'$.

Если дизъюнкт E получается из дизъюнктов E_1 и E_2 по правилу резолюций, то будем говорить, что E *непосредственно зависит от* E_1 и от E_2. Транзитивное замыкание отношения непосредственной зависимости назовем *отношением зависимости*. (Другими словами, E зависит от E', если существуют дизъюнкты $E_1, \ldots, E_n$ такие, что $E = E_1$, $E_n = E'$ и E_1 непосредственно зависит от E_2, E_2 непосредственно зависит от $E_3, \ldots, E_{n-1}$ непосредственно зависит от E_n.) Преобразуем второй вывод следующим образом: к дизъюнкту B_j и всем дизъюнктам, которые от него зависят, добавим литерал L. Новая последовательность дизъюнктов

$$B_1, B_2, \ldots, B_j' = D_k' \vee L, \quad B'_{j+1}, \ldots, B_m' \qquad (3)$$

будет выводом из S. Если дизъюнкт B_m не зависит от B_j, то $B_m' = \square$. Это означает, что из S выводим пустой дизъюнкт, что и требовалось доказать. Предположим, что B_m зависит от B_j. Тогда $B_m' = L$. Преобразуем теперь первый вывод: на место дизъюнкта A_i (равного L) в этой последовательности подставим последовательность (3). Получим последовательность

$$A_1, \ldots, A_{i-1}, B_1, \ldots, B_j', B'_{j+1}, \ldots, B_m' = L, A_{i+1}, \ldots, A_l = \square.$$

Эта последовательность является выводом пустого дизъюнкта из множества дизъюнктов S. Следовательно, если множество S невыполнимо, то из S выводим пустой дизъюнкт. □

Для доказательства того, что формула G является логическим следствием множества формул $F_1, \ldots, F_k$, метод резолюций применяется следующим образом.

Сначала составляется множество формул $T = \{F_1, \ldots, F_k, \neg G\}$. Затем каждая из этих формул приводится к конъюнктивной нормальной форме и в полученных формулах зачеркиваются знаки конъюнкции. Получается множество дизъюнктов S. И, наконец, ищется вывод пустого дизъюнкта из S. Если пустой дизъюнкт выводим из S, то формула G является логическим следствием формул $F_1, \ldots, F_k$. Если из S нельзя вывести $\square$, то G не является логическим следствием формул $F_1, \ldots, F_k$.

Пример 1. Проиллюстрируем сказанное на примере. Покажем, что формула $G = Z$ является логическим следствием формул $F_1 = \neg X \vee Y \to X \& Z$, $F_2 = \neg Y \to Z$. Сформируем множество формул $T = \{F_1, F_2, \neg G\}$. Приведем формулы F_1 и F_2 к КНФ (формула $\neg G$ сама имеет эту форму). Мы получим, что

F_1 равносильна $X \& (\neg Y \vee Z)$,

F_2 равносильна $Y \vee Z$.

Тогда множество дизъюнктов S равно

$$\{ X, \neg Y \vee Z, Y \vee Z, \neg Z \}.$$

Из множества S легко выводится пустой дизъюнкт:

$$\neg Y \vee Z, \neg Z, \neg Y, Y \vee Z, Y, \square.$$

Следовательно, формула G является логическим следствием формул F_1 и F_2. □

§2. Подстановка и унификация

Рассмотрим метод резолюций в логике первого порядка. Относительно переменных в дизъюнктах будем предполагать, что они связаны кванторами общности, но сами кванторы писать не будем. Отсюда следует, что две одинаковые переменные в разных дизъюнктах можно считать различными.

Заметим, прежде всего, что в логике первого порядка правило резолюций в прежнем виде уже не годится. Действительно, множество дизъюнктов $S = \{P(x), \neg P(a)\}$ невыполнимо (так как предполагается, что переменная x связана квантором общности). В то же время, если использовать правило резолюций для логики высказываний, то из S пустого дизъюнкта не получить. Содержательно понятно, что именно в этом случае надо сделать. Поскольку дизъюнкт $P(x)$ можно прочитать “для любого x истинно $P(x)$”, ясно, что $P(x)$ истинно будет и для $x = a$. Сделав подстановку x = a, получим множество дизъюнктов $S' = \{P(a), \neg P(a)\}$. Множества S и S' одновременно выполнимы или невыполнимы. Но из S' пустой дизъюнкт с помощью прежнего правила резолюций выводится тривиальным образом. Этот пример подсказывает, что в логике первого порядка правило резолюций надо дополнить возможностью делать подстановку.

Дадим необходимые определения.

Определение. *Подстановкой* называется множество равенств

$$\sigma = \{x_1 = t_1, x_2 = t_2, \ldots, x_n = t_n\},$$

где $x_1, x_2, \ldots, x_n$ – различные переменные, $t_1, t_2, \ldots, t_n$ – термы, причем терм t_i не содержит переменной x_i $(1 \le i \le n)$.

Если $\sigma = (x_1 = t_1, \ldots, x_n = t_n)$ – подстановка, а F – дизъюнкт, то через $\sigma(F)$ будем обозначать дизъюнкт, полученный из F *одновременной* заменой x_1 на t_1, x_2на t_2 и т.д. x_n на t_n. Например, если $\sigma = \{ x_1 = f(x_2), x_2 = c, x_3 = g(x_4) \}$, $F = R(x_1, x_2, x_3) \vee \neg P(f(x_2))$, то $\sigma(F) = R(f(x_2), c, g(x_4)) \vee \neg P(f(c))$. Аналогично определяется действие подстановки на терм.

Для удобства введем еще и *пустую подстановку* – подстановку, не содержащую равенств. Пустую подстановку будем обозначать через ε.

Определение. Пусть $\{E_1, \ldots, E_k\}$ – множество литералов или множество термов. Подстановка σ называется *унификатором* этого множества, если $\sigma(E_1) = \sigma(E_2) = \ldots = \sigma(E_k)$. Множество *унифицируемо*, если существует унификатор этого множества.

Например, множество атомарных формул

$$\{Q(a, x, f(x)), Q(u, y, z)\}$$

унифицируемо подстановкой $\{u = a, x = y, z = f(\text{y})\}$, а множество

$$\{R(x, f(x)), R(u, u)\}$$

неунифицируемо. Действительно, если заменить x на u, то получим множество

$$\{R(u, f(u)), R(u, u)\}.$$

Проводить же замену $u = f(u)$ запрещено определением подстановки, да и бесполезно, т.к. она приводит к формулам $R(f(u), f(f(u)))$ и $R(f(u), f(u))$, которые тоже различны.

Если множество унифицируемо, то существует, как правило, не один унификатор этого множества, а несколько. Среди всех унификаторов данного множества выделяют так называемый наиболее общий унификатор.

Дадим необходимые определения.

Определение. Пусть $\xi = \{x_1 = t_1, x_2 = t_2, \ldots, x_k = t_k\}$ и $\eta = \{y_1 = s_1, y_2 = s_2, \ldots, y_l = s_l\}$ – две подстановки. Тогда *произведением* подстановок ξ и η называется подстановка, которая получается из последовательности равенств

$$\{x_1 = \eta(t_1), x_2 = \eta(t_2), \ldots, x_k = \eta(t_k), \\ y_1 = s_1, y_2 = s_2, \ldots, y_l = s_l\} \qquad (4)$$

вычеркиванием равенств вида $x_i = x_i$ для $1 \le i \le k$, $y_j = s_j$, если $y_j \in \{x_1, \ldots, x_k\}$, для $1 \le j \le l$.

Произведение подстановок ξ и η будем обозначать через $\xi \circ \eta$. Подчеркнем, что сначала действует ξ, а потом η.

Пример 2. Рассмотрим пример. Пусть $\xi = \{x = f(y), z = y, u = g(d)\}$, $\eta = \{x = c, y = z\}$. Тогда последовательность равенств (4) из определения произведения имеет вид

$$\{x = f(z), z = z, u = g(d), x = c, y = z\}.$$

В этой последовательности вычеркнем второе и четвертое равенство получим произведение

$$\xi \circ \eta = \{x = f(z),\ u = g(d),\ y = z\}.$$

Нетрудно показать, что произведение подстановок ассоциативно, т.е. для любых подстановок ξ, η, ξ выполняется равенство $\xi \circ (\eta \circ \zeta) = (\xi \circ \eta) \circ \zeta$, и что пустая подстановка является нейтральным элементом относительно умножения. Последнее означает выполнение равенств $\sigma \circ \varepsilon = \varepsilon \circ \sigma = \sigma$ для любой подстановки σ. □

Произведение подстановок $\sigma = \{x_1 = t_1\} \circ \{x_2 = t_2\} \circ \ldots \circ \{x_n = t_n\}$ мы будем иногда задавать последовательностью равенств: $\sigma = (x_1 = t_1,\ x_2 = t_2,\ \ldots,\ x_n = t_n)$. Действие подстановки σ на дизъюнкт (и на терм) в этом случае состоит в *последовательной* (а не одновременной) замене x_1 на t_1, x_2 на t_2, и т.д., x_n на t_n.

Определение. Унификатор σ множества литералов или термов называется *наиболее общим унификатором* этого множества, если для любого унификатора τ того же множества литералов существует подстановка ξ такая, что $\tau = \sigma \circ \xi$.

Например, для множества $\{P(x, f(\text{a}), g(z)), P(f(b), y, v)\}$ наиболее общим унификатором является подстановка $\sigma = \{x = f(b),\ y = f(a),\ v = g(z)\}$. Если в качестве τ взять унификатор $\{x = f(b),\ y = f(a),\ z = c,\ v = g(c)\}$, то $\xi = \{z = c\}$.

Если множество литералов унифицируемо, то наиболее общий унификатор существует. Это утверждение мы докажем в конце параграфа. А сейчас приведем алгоритм нахождения наиболее общего унификатора. Алгоритм называется *алгоритмом унификации*. Для изложения алгоритма потребуется понятие множества рассогласований.

Определение. Пусть M – множество литералов (или термов). Выделим первую слева позицию, в которой не для всех литералов (или термов) стоит один и тот же символ. Затем в каждом литерале выпишем выражение, которое начинается символом, занимающим эту позицию. (Этими выражениями могут быть сам литерал, атомарная формула или терм.) Множество полученных выражений называется *множеством рассогласований* в M.

Например, если $M = \{P(x, f(y), a),\ P(x, u, g(y)),\ P(x, c, v)\}$, то первая слева позиция, в которой не все литералы имеют один и тот же символ – пятая позиция. Множество рассогласований состоит из термов $f(y)$, u, c. Множество рассогласований $\{P(x, y),\ \neg P(a, g(z))\}$ есть само множество. Если $M = \{\neg P(x, y),$

$\neg Q(a, v)\}$, то множество рассогласований равно $\{P(x, y), Q(a, v)\}$.

Алгоритм унификации

Шаг 1. Положить $k = 0$, $M_k = M$, $\sigma_k = \varepsilon$.

Шаг 2. Если множество M_k состоит из одного литерала, то выдать σ_k в качестве наиболее общего унификатора и завершить работу. В противном случае найти множество N_k рассогласований в M_k.

Шаг 3. Если в множестве N_k существует переменная v_k и терм t_k, не содержащий v_k, то перейти к шагу 4, иначе выдать сообщение о том, что множество M неунифицируемо и завершить работу.

Шаг 4. Положить $\sigma_{k+1} = \sigma_k \circ \{v_k = t_k\}$ (подстановка σ_{k+1} получается из σ_k заменой v_k на t_k и, возможно, добавлением равенства $v_k = t_k$). В множестве M_k выполнить замену $v_k = t_k$, полученное множество литералов взять в качестве M_{k+1}.

Шаг 5. Положить $k = k+1$ и перейти к шагу 2.

Пусть $M = \{P(x, f(y)), P(a, u)\}$. Проиллюстрируем работу алгоритма унификации на множестве M. На первом проходе алгоритма будет найдена подстановка $\sigma_1 = \{x = a\}$, так как множество рассогласований N_0 равно $\{x, a\}$. Множество M_1 будет равно $\{P(a, f(y)), P(a, u)\}$. На втором проходе алгоритма подстановка будет расширена до $\sigma_2 = \{x = a, u = f(y)\}$ и $M_2 = \{P(a, f(u))\}$. Так как M_2 состоит из одного литерала, то алгоритм закончит работу и выдаст σ_2.

Рассмотрим еще один пример. Пусть $M = \{P(x, f(y)), P(a, b)\}$. На первом проходе алгоритма будет найдена подстановка $\sigma_1 = \{x = a\}$ и $M_1 = \{P(a, f(y)), P(a, b)\}$. На третьем шаге второго прохода будет выдано сообщение о том, что множество M неунифицируемо, так как множество рассогласования $N_1 = \{f(y), a\}$ не содержит переменной.

Отметим, что при выполнении шага 4 из множества M_k удаляется одна из переменных (переменная v_k), а новая переменная не возникает. Это означает, что алгоритм унификации всегда заканчивает работу, так как шаг 4 не может выполняться бесконечно. Довольно ясно, что если алгоритм заканчивает работу на шаге 3, то множество M неунифицируемо. Также понятно, что если алгоритм заканчивает работу на шаге 2, то σ_k – унификатор множества M. А вот то, что σ_k – наиболее общий унификатор, доказать не так то просто. Тем не менее, сделаем это.

Теорема 4.2. Пусть M – конечное непустое множество литералов. Если M унифицируемо, то алгоритм унификации заканчивает работу на шаге 2 и выдаваемая алгоритмом подстановка σ_k является наиболее общим унификатором множества M.

Доказательство. Пусть τ – некоторый унификатор множества M. Индукцией по k докажем существование подстановки α_k такой, что $\tau = \sigma_k \circ \alpha_k$.

База индукции: $k = 0$. Тогда $\sigma_k = \varepsilon$ и в качестве α_k можно взять τ.

Шаг индукции. Предположим, что для всех значений k, удовлетворяющих неравенству $0 \le k \le l$, существует подстановка α_k такая, что $\tau = \sigma_k \circ \alpha_k$.

Если $\sigma_l(M)$ содержит один литерал, то на следующем проходе алгоритм остановится на шаге 2. Тогда σ_l будет наиболее общим унификатором, поскольку $\tau = \sigma_l \circ \alpha_l$.

Пусть $\sigma_l(M)$ содержит более одного литерала. Тогда алгоритм унификации найдет множество рассогласований N_l. Подстановка α_l должна унифицировать множество N_l, поскольку $\tau = \sigma_l \circ \alpha_l$ – унификатор множества M. Поскольку N_l – унифицируемое множество рассогласований, оно содержит (хотя бы одну) переменную v.

Обозначим буквой t терм из N_l, отличный от v. Множество N_l унифицируется подстановкой α_l, поэтому $\alpha_l(v) = \alpha_l(t)$. Отсюда следует, что t не содержит v. Можно считать, что на шаге 4 алгоритма для получения σ_{l+1} использовано равенство $v = t$, т.е. $\sigma_{l+1} = \sigma_l \circ \{v = t\}$. Из равенства $\alpha_l(v) = \alpha_l(t)$ следует, что α_l содержит равенство $v = \alpha_l(t)$.

Пусть $\alpha_{l+1} = \alpha_l \setminus \{v = \alpha_l(t)\}$. Тогда $\alpha_{l+1}(t) = \alpha_l(t)$, так как t не содержит v. Далее, имеем равенства

$$\{v = t\} \circ \alpha_{l+1} = \alpha_{l+1} \cup \{v = \alpha_{l+1}(t)\} = \alpha_{l+1} \cup \{v = \alpha_l(t)\} = \alpha_l.$$

Это означает, что $\alpha_l = \{v = t\} \circ \alpha_{l+1}$. Следовательно,

$$\tau = \sigma_l \circ \alpha_l = \sigma_{l+1} \circ \{v = t\} \circ \alpha_l = \sigma_{l+1} \circ \alpha_{l+1}.$$

Итак, для любого k существует подстановка α_k такая, что $\tau = \sigma_k \circ \alpha_k$. Так как множество M унифицируемо, то алгоритм должен закончить работу на шаге 2. Тогда последняя подстановка σ_k будет унификатором множества M, поскольку множество $\sigma_k(M)$ состоит из одного литерала. Более того, σ_k будет наиболее общим унификатором, так как для произвольного унификатора τ существует подстановка σ_k такая, что $\tau = \sigma_k \circ \alpha_k$. □

§3. Метод резолюций в логике первого порядка

Определение. *Правилом резолюций в логике предикатов* называется следующее правило: из дизъюнктов $\neg P(t_1, \ldots, t_n) \vee F$ и $P(s_1, \ldots, s_n) \vee G$ выводим дизъюнкт $\sigma(F) \vee \sigma(G)$, где σ – наиболее общий унификатор мноожества

$$\{P(t_1, \ldots, t_n), P(s_1, \ldots, s_n)\}.$$

Дизъюнкт $\sigma(F) \vee \sigma(G)$ называется *бинарной резольвентой* первых двух дизъюнктов, а литералы $\neg P(t_1, \ldots, t_n)$ и $P(s_1, \ldots, s_n)$ *отрезаемыми* литералами.

Например, с помощью правила резолюций из дизъюнктов $\neg Q(a, f(x)) \vee R(x)$ и $Q(u, z) \vee \neg P(z)$ можно вывести дизъюнкт $R(x) \vee \neg P(f(x))$, используя подстановку $\sigma = \{u = a, z = f(x)\}$.

В отличие от логики высказываний, в логике предикатов нам понадобится еще одно правило.

Определение. *Правилом склейки в логике предикатов* называется следующее правило: из дизъюнкта $\Diamond P(t_1, \ldots, t_n) \vee \ldots \vee \Diamond P(s_1, \ldots, s_n) \vee F$ выводим дизъюнкт $\sigma(\Diamond P(t_1, \ldots, t_n)) \vee \sigma(F)$, где σ – наиболее общий унификатор множества $\{P(t_1, \ldots, t_n), \ldots, P(s_1, \ldots, s_n)\}$, $\Diamond$ – знак отрицания или его отсутствие. Дизъюнкт $\sigma(\Diamond P(t_1, \ldots, t_n)) \vee \sigma(F)$ называется *склейкой* исходного дизъюнкта. (Отметим, что если знак отрицания стоит перед одной из записанных выше атомарных формул, то он стоит и перед другими.)

Например, правило склейки, примененное к дизъюнкту

$$\neg \mathrm{P}(x, y) \vee \neg P(y, x) \vee \neg \mathrm{P}(a, a) \vee Q(x, y, v),$$

дает дизъюнкт

$$\neg \mathrm{P}(a, a) \vee Q(a, a, v).$$

Определение вывода в логике первого порядка немного отличается от аналогичного определения в логике высказываний.

Определение. Пусть S – множество дизъюнктов.*Выводом из множества дизъюнктов* S называется последовательность дизъюнктов

$$D_1, D_2, \ldots, D_n$$

такая, что каждый дизъюнкт D_i принадлежит S, выводим из предыдущих дизъюнктов по правилу резолюций или выводим из предыдущего по правилу склейки.

Как и в логике высказываний, дизъюнкт D выводим из S, если существует вывод из S, последним дизъюнктом которого является D.

Пример 1. Приведем пример. Пусть $S = \{\neg B(x) \vee \neg C(x) \vee T(f(x)), C(y) \vee T(f(z)), B(a)\}$. Тогда последовательность

$D_1 = \neg B(x) \vee \neg C(x) \vee T(f(x))$, $D_2 = C(y) \vee T(f(z))$,
$D_3 = \neg B(x) \vee T(f(x)) \vee T(f(z))$, $D_4 = \neg B(x) \vee T(f(x))$,
$D_5 = B(a)$, $D_6 = T(f(a))$

является выводом из S. Отметим, что D_1, D_2, $D_5 \in S$, дизъюнкт D_3 выводим из D_1 и D_2 по правилу резолюций, дизъюнкт D_6 выводим из D_4 и D_5 по тому же правилу, а D_4 выводим из D_3 по правилу склейки. □

Как и в логике высказываний, в логике первого порядка есть утверждение, называемое теоремой о полноте. Это утверждение фактически совпадает с формулировкой теоремы 4.1. Тем не менее, приведем его.

Теорема 4.3. Множество дизъюнктов S логики первого порядка невыполнимо тогда и только тогда, когда из S выводим пустой дизъюнкт.

Теорема имеет довольно сложное доказательство. Оно будет приведено в §6. В данном же параграфе мы ограничимся примером применения метода резолюций и рядом определений, необходимых для доказательства теоремы 4.3.

Для доказательства логичности следствия формулы G из формул $F_1, \ldots, F_k$ метод резолюций в логике предикатов применяется почти так же, как и в логике высказываний. А именно, сначала составляется множество формул $T = \{F_1, \ldots, F_k, \neg G\}$. Затем каждая из формул этого множества приводится к сколемовской нормальной форме, в полученных формах зачеркиваются кванторы общности и связки конъюнкции. Получается множество дизъюнктов S. На последнем этапе находится вывод пустого дизъюнкта из множества S. Напомним, что все переменные в дизъюнктах предполагаются связанными кванторами общности. Это означает, что *метод резолюций для доказательства логичности может применяться лишь в случае, когда формулы* $F_1, \ldots, F_k$ и G *не имеют свободных переменных*. Если все же формулы содержат свободные переменные, то их надо заменить константами (такими, которые отсутствуют в этих формулах).

Пример 2. Рассмотрим пример. Пусть

$$F_1 = (\exists x)[П(x) \,\&\, (\forall y)(C(y) \to З(x, y))],$$

$$F_2 = (\forall x)(\forall y)[П(x) \,\&\, Л(y) \to \neg З(x, y)],$$
$$G = (\forall x)(C(x) \to \neg Л(x)).$$

Докажем, что формула G является логическим следствием множества формул F_1, F_2. Для этого достаточно доказать невыполнимость множества T = $\{F_1, F_2, \neg G\}$. Каждую из формул множества T приведем к сколемовской нормальной форме; получим формулы

$$(\forall y)[П(a) \,\&\, (\neg C(y) \vee З(a, y))],$$
$$(\forall x)(\forall y)[\neg П(x) \vee \neg Л(y) \vee \neg З(x, y)],$$
$$C(b) \,\&\, Л(b).$$

Тогда множество S будет содержать дизъюнкты: $D_1 = П(a)$, $D_2 = \neg C(y) \vee З(a, y)$, $D_3 = \neg П(x) \vee \neg Л(y) \vee \neg З(x, y)$, $D_4 = Cb)$, $D_5 = Л(b)$. А последовательность дизъюнктов D_1, D_3, $\neg Л(y) \vee \neg З(a, y)$, D_5, $\neg$З(a, b), D_2, D_4, З(a, b), □ будет выводом из S. Следовательно, формула G является логическим следствием формул F_1 и F_2. □

Введем теперь ряд определений, необходимых для рассмотрений в следующих параграфах.

Напомним, что мы условились не писать в дизъюнктах повторяющиеся литералы. Это позволяет нам смотреть, если это необходимо, на дизъюнкт как на множество литералов. Если смотреть на дизъюнкт как на множество литералов, то результат применения правила резодюций к дизъюнктам D_1 и D_2 с отрезаемыми литералами L_1 и L_2 можно записать так

$$D = [\sigma(D_1) – \sigma(L_1)] \cup [\sigma(D_2) – \sigma(L_2)],$$

где σ – наиболее общий унификатор L_1 и $\neg L_2$.

Определение. *Резольвентой* дизъюнктов D_1 и D_2 называется одна из следующих бинарных резольвент:

1) бинарная резольвента дизъюнктов D_1 и D_2,
2) бинарная резольвента склейки D_1 и дизъюнкта D_2,
3) бинарная резольвента дизъюнкта D_1 и склейки D_2,
4) бинарная резольвента склейки D_1 и склейки D_2.

Пример 3. Пусть $D_1 = \neg P(y) \vee \neg P(g(x)) \vee R(f(y))$, $D_2 = P(g(a)) \vee Q(b)$. Склейка дизъюнкта D_1 есть дизъюнкт $D_1' = \neg P(g(x)) \vee R(f(g(x)))$. Бинарная резольвента D_1' и D_2 равна $R(f(g(a))) \vee Q(b)$. Следовательно, последний дизъюнкт есть резольвента дизъюнктов D_1 и D_2. □

Определение. Если D – дизъюнкт, а σ – подстановка, то дизъюнкт $\sigma(D)$ называется *примером дизъюнкта* D.

Следующее утверждение часто называют *леммой о подъеме*.

Теорема 4.4. Если D_1' – пример дизъюнкта D_1, D_2' – пример дизъюнкта D_2, а D' – резольвента D_1' и D_2', то существует резольвента D дизъюнктов D_1 и D_2 такая, что D' – пример D.

Доказательство. Если D_1 и D_2 имеют общие переменные, то заменой переменных в одном из дизъюнктов можно добиться того, что переменные дизъюнкта D_1 отличны от переменных дизъюнкта D_2. Будем поэтому считать, что D_1 и D_2 не имеют общих переменных.

Так как D_1' – пример D_1 и D_2' – пример D_2, существуют подстановки α_1 и α_2 такие, что $D_1' = \alpha_1(D_1)$ $D_2' = \alpha_2(D_2)$. Последовательность $\alpha = (\alpha_1, \alpha_2)$ также будет подстановкой, и поскольку D_1 и D_2 не имеют общих переменных, имеем $D_1' = \alpha(D_1)$ и $D_2' = \alpha(D_2)$.

Дизъюнкт D' является резольвентой дизъюнктов D_1' и D_2'. Это означает, что существуют литералы $L_1' \in D_1'$ и $L_2' \in D_2'$ и подстановка τ такие, что τ есть наиболее общий унификатор L_1' и $\neg L_2'$ и

$$D' = (\tau(D_1') - \tau(L_1')) \cup (\tau(D_2') - \tau(L_2')). \qquad (1)$$

(Если при получении резольвенты D' к дизъюнктам D_1' и D_2' применялись склейки, то будем считать, что они учтены подстановками α_1 и α_2.)

Пусть $L_1^1, \ldots, L_1^r$ – литералы дизъюнкта D_1, которые подстановкой α переводятся L_1', а $L_2^1, \ldots, L_2^s$ – литералы дизъюнкта D_2, которые подстановкой α переводятся в L_2'. Литералы $L_1^1, \ldots, L_1^r$, следовательно, унифицируемы, а поэтому существует наиболее общий унификатор β_1 для этого множества. Литерал $\beta_1(L_1^1)$ (равный $\beta_1(L_1^2), \ldots, \beta_1(L_1^r)$) обозначим через L_1. По определению наиболее общего унификатора найдется подстановка γ_1, для которой выпол–няется равенство $\alpha_1 = \beta_1 \circ \gamma_1$. По аналогичным соображениям, существуют подстановки β_2 и γ_2 такие, что β_2 – наиболее общий унификатор множества литералов $L_2^1, \ldots, L_2^s$ и $\alpha_2 = \beta_2 \circ \gamma_2$. Литерал $\beta_2(L_2^1)$ обозначим через L_2. Дегко видеть, что L_1 и L_2 не имеют общих переменных. Поскольку дизъюнкты D_1 и D_2 также не имеют общих переменных, то можно считать, что $\beta_1 = \beta_2 = \beta$, $\gamma_1 = \gamma_2 = \gamma$ и $\alpha = \beta \circ \gamma$. Сказанное в этом абзаце иллюстрируется рисунками 4.1 и 4.2.

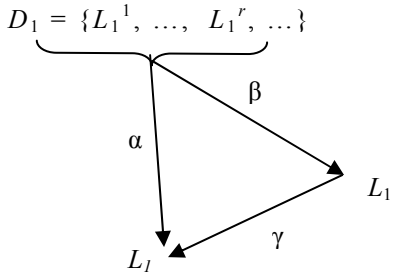


Рис. 4.1

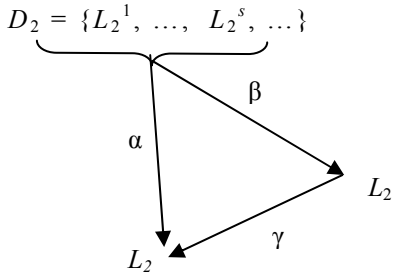


Рис. 4.2

Литералы L_1' и $\neg L_2'$, как отмечено выше, унифицируемы подстановкой τ. Следовательно, литералы L_1 и $\neg L_2$ также унифицируемы (подстановкой $\gamma\circ\tau$). Отсюда следует, что существует наиболеее общий унификатор σ множества $\{L_1, \neg L_2\}$ (см.рис.4.3). Возьмем в качестве D дизъюнкт

$$D = [\sigma(\beta(D_1)) - \sigma(L_1)] \cup [\sigma(\beta(D_2)) - \sigma(L_2)] \qquad (2)$$

Ясно, что D – резольвента дизъюнктов D_1 и D_2. Осталось показать, что D' – пример D.

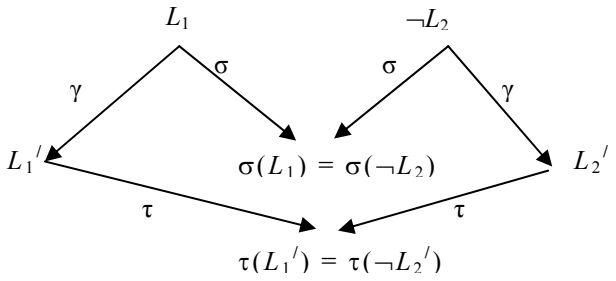


Рис. 4.3

Так как σ – наиболее общий унификатор L_1 и $\neg L_2$, то существует подстановка δ такая, что $\gamma\circ\tau = \sigma\circ\delta$. В таком случае из последнего равенства, равенств (1), (2) и $\alpha = \beta\circ\gamma$ следует, что

$$D' = (\tau(D_1') - \tau(L_1')) \cup (\tau(D_2') - \tau(L_2')) =$$
$$[\tau(\alpha(D_1)) - \tau(\alpha(L_1^1))] \cup [\tau(\alpha(D_2)) - (\tau(\alpha(L_2^1)) =$$
$$[(\alpha\circ\tau)(D_1) - (\alpha\circ\tau)(L_1^1)] \cup [(\alpha\circ\tau)(D_2) - (\alpha\circ\tau)(L_2^1)] =$$
$$[(\beta\circ\gamma\circ\tau)(D_1) - (\beta\circ\gamma\circ\tau)(L_1^1)] \cup [(\beta\circ\gamma\circ\tau)(D_2) - (\beta\circ\gamma\circ\tau)(L_2^1)] =$$
$$[(\beta\circ\sigma\circ\delta)(D_1) - (\beta\circ\sigma\circ\delta)(L_1^1)] \cup [(\beta\circ\sigma\circ\delta)(D_2) - (\beta\circ\sigma\circ\delta)(L_2^1)] =$$
$$\delta[\sigma(\beta(D_1)) - \sigma(L_1)] \cup \delta[\sigma(\beta(D_2)) - \sigma(L_2)] = \delta(D).$$

Мы доказали, что D' – пример D. □

§4. Эрбрановский универсум множества дизъюнктов

По определению интерпретации ее областью может быть любое непустое множество M. Было бы удобно иметь одно множество в качестве области интерпретации. В случае, когда решается вопрос о выполнимости множества дизъюнктов, таким множеством является так называемый эрбрановский универсум.

Пусть S – множество дизъюнктов.

Введем следующие обозначения. Через H_0 обозначим множество констант, содержащихся в S. Если S не содержит констант, то H_0 состоит из одной константы, скажем a, т.е. $H_0 = \{a\}$. Предположим, что введено множество H_i. Тогда H_{i+1} есть объединение множества H_i и термов вида $f(t_1, \ldots, t_n)$, где $t_1, \ldots, t_n \in H_i$, f – символ n-местной функции, содержащейся хотя бы в одном из дизъюнктов множества S.

Определение. Множество $H_\infty = H_0 \cup H_1 \cup \ldots \cup H_n \cup \ldots$ называется *эрбрановским универсумом* множества дизъюнктов S.

Приведем три примера, которые будем использовать в дальнейшем.

Пример 1. Пусть $S = \{P(x), \neg P(x) \vee Q(f(y)), \neg Q(f(a))\}$. Тогда
$H_0 = \{a\}$,
$H_1 = \{a, f(a)\}$,
. . .
$H_\infty = \{a, f(a), f(f(a)), \ldots\}$.

Пример 2. Пусть $S = \{P(x), Q(x) \vee \neg R(y)\}$. Множество S не содержит констант, поэтому $H_0 = \{a\}$. Так как дизъюнкты из S не содержат функциональных символов, выполняются равенства $H_0 = H_1 = H_2 = \ldots = H_\infty = \{a\}$.

Пример 3. Пусть $S = \{P(x), \neg P(b) \vee Q(y, f(y, a))\}$. Тогда $H_\infty = \{a, b, f(a, a), f(a, b), f(b, a), f(b, b), f(f(a, a), a), \ldots\}$.

Эрбрановский универсум, как мы видим, определяется не всем множеством дизъюнктов S, а только символами функций и константами, входящими в дизъюнкты из S.

Определение. Множество атомарных формул вида $P(t_1, \ldots, t_n)$, где $t_1, \ldots, t_n \in \mathrm{H}_\infty$, а P – символ n-местного предиката, входящий хотя бы в один дизъюнкт из S, называется *эрбрановским базисом* множества дизъюнктов S.

Для множества дизъюнктов S из примера 1 эрбрановским базисом будет множество атомарных формул $B_1 = \{P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), \ldots\}$. Эрбрановским базисом множества дизъюнктов S из примера 2 будет множество $B_2 = \{P(a), Q(a), R(a)\}$.

Пусть термин "выражение" означает терм, атомарную формулу, литерал или дизъюнкт. Тогда *основным выражением* будем называть выражения, не содержащие переменных.

Определение. Пусть D – дизъюнкт из множества дизъюнктов S. *Основным примером* дизъюнкта D называется дизъюнкт, полученный из D заменой переменных на элементы эрбрановского универсума H_∞.

Пусть S – множество дизъюнктов из примера 1. Тогда дизъюнкт $\neg Q(f(a))$ имеет один основной пример – сам этот дизъюнкт, а множество основных примеров дизъюнкта $\neg P(x) \vee Q(f(y))$ бесконечно: $\{\neg P(a) \vee Q(f(a)), \neg P(f(a)) \vee Q(f(a)), \neg P(a) \vee Q(f(f(a))),\ldots\}$. Если S – множество дизъюнктов из примера 2, то каждый из дизъюнктов этого множества S имеет один основной пример.

Как утверждалось в начале параграфа (и будет доказано в конце), для решения вопроса о выполнимости множества дизъюнктов в качестве области интерпретации достаточно рассматривать только эрбрановский универсум. Оказывается, можно еще ограничить и саму интерпретацию до так называемой H-интерпретации.

Определение. Пусть H_∞ – эрбрановский универсум множества дизъюнктов S. Интерпретация φ с областью H_∞ называется *H-интерпретацией множества S*, если она удовлетворяет следующим условиям:

1) для любой константы c из S выполняется равенство $\varphi(c) = c$,

2) если f – символ n-местной функции из S, то φf – функция, определенная на H_∞ равенством

$$(\varphi f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$$

для любых $t_1, \ldots, t_n \in H_\infty$.

Если $B = \{B_1, B_2, \ldots, B_n, \ldots\}$ – эрбрановский базис множества дизъюнктов S, то H-интерпретацию φ удобно представлять в виде множества литералов

$$\{L_1, L_2, \ldots, L_n, \ldots\},$$

где L_i есть B_i, если $\varphi(B_i) = 1$, и $L_i = \neg B_i$, если $\varphi(B_i) = 0$.

Например, если S – множество дизъюнктов из примера 1, то эрбрановскими интерпретациями будут

$\varphi_1 = \{ P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), \ldots\}$

$\varphi_2 = \{ P(a), Q(a), \neg P(f(a)), \neg Q(f(a)), P(f(f(a))), Q(f(f(a))), \neg P(f(f(f(a)))), \ldots\}$.

$\varphi_3 = \{ P(a), \neg Q(a), P(f(a)), \neg Q(f(a)), P(f(f(a))), \neg Q(f(f(a))), P(f(f(f(a)))), \ldots\}$.

Последнее, что нужно сделать, чтобы доказать основное утверждение этого параграфа (теорему 4.5), ввести понятие H-интерпретации φ^*, соответствующей (произвольной) интерпретации φ множества S.

Предположим, что S содержит хотя бы одну константу. Если φ – интерпретация множества S с областью M, то для любого элемента h эрбрановского универсума значение $\varphi(h)$ определено (и является элементом множества M).

Определение. Пусть φ – интерпретация множества S с областью M. Тогда *H-интерпретацией φ^*, соответствующей интерпретации* φ называется H-интерпретация, удовлетворяющая следующему условию: для любых элементов $t_1, \ldots, t_n$ эрбрановского универсума выполняется эквиваленция:

$$(\varphi^* P)(t_1, \ldots, t_n) = 1 \Leftrightarrow (\varphi P)(\varphi(t_1), \ldots, \varphi(t_n)) = 1$$

для любого символа предиката P.

Приведем пример. Пусть S – множество дизъюнктов из примера 1. Напомним, что $S = \{P(x), \neg P(x) \vee Q(f(y)), \neg Q(f(a))\}$ и эрбрановский базис множества S есть $B = \{P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), \ldots\}$. Рассмотрим интерпертацию φ с областью M = {1, 2}, определяемую равенством $\varphi(a) = 1$ и таблицей 4.1.

Таблица 4.1

	φf	φP	φQ
1	2	*0*	*1*
2	1	*1*	*1*

(В столбцах φP и φQ цифры *0* и *1* – истинностные значения.) Тогда

$$\varphi^* = \{\neg P(a), Q(a), P(f(a)), Q(f(a)), \neg P(f(f(a))), \ldots\}.$$

Рассмотрим теперь случай, когда S не содержит констант. Пусть φ – интерпретация множества S с областью M и a – константа, образующая эрбрановский универсум H_∞. В этом случае значение $\varphi(a)$ неопределено. Для получения H-интерпретации φ^* расширяем функцию φ на a, полагая $\varphi(a)$ равным произволь-

ному элементу из M. Далее поступаем так, как описано выше. Если множество M неодноэлементно, то мы можем получить не одну H-интерпретацию φ^*, соответствующую φ. Нетрудно привести пример, когда H-интерпретаций φ^* столько же, сколько элементов в множестве M.

Следующее утверждение непосредственно следует из определений.

Лемма. Пусть φ – интерпретация с областью M, при которой все дизъюнкты из S истинны. Тогда все дизъюнкты из S истинны при любой H-интерпретации φ^*, соответствующей φ.

Теорема 4.5. Множество дизъюнктов S невыполнимо тогда и только тогда, когда S ложно при всех H-интерпретациях, т.е. для любой H-интерпретации множества S в S найдется дизъюнкт, который ложен при этой H-интерпретации.

Доказательство. Необходимость очевидна. Действительно, невыполнимость множества S означает, что это множество ложно при любой интерпретации. В том числе и при любой H-интерпретации. Достаточность следует из леммы, поскольку если S выполнимо, то существует хотя бы одна интепретация φ, при которой все дизъюнкты из S истинны. Но тогда все дизъюнкты из S будут истинны и при H-интерпретации φ^*. □

§5. Семантические деревья, теорема Эрбрана

В предыдущем параграфе мы видели, что для получения ответа на вопрос о выполнимости множества дизъюнктов можно рассматривать не все интерпертации, а только H-интерпретации. В данном параграфе мы продвинемся еще дальше в этом направлении. Мы фактически покажем, что для решения упомянутого вопроса можно ограничиться конечными подмножествами эрбрановского универсума. Основным понятием этого параграфа будет понятие семантического дерева.

На дерево будем смотреть, как на корневое ориентированное дерево. Изображать дерево будем растущим вниз, ориентацию дуг указывать не будем. Дугам дерева будут поставлены в соответствие (приписаны) множества литералов. Напомним, что *листом* дерева называется вершина, из которой не выходит не одна дуга. *Путь* в дереве – это последовательность дуг

$$e_1, e_2, \ldots, e_k, \ldots$$

такая, что если дуга e_i заходит в некоторую вершину, то дуга e_{i+1} выходит из этой вершины. Путь в дереве называется *максимальным*, если к нему нельзя добавить ни одной дуги. Для каждой вершины существует единственный путь от корня к этой вершине. Если вершина является листом, то этот путь максимален. Если π – путь в дереве, то через $I(\pi)$ будем обозначать объединение всех множеств литералов, приписанных дугам пути. В случае, когда π – путь от корня до вершины v, то вместо $I(\pi)$ будем писать $I(v)$. Мы будем использовать понятие поддерева несколько в ином смысле, нежели в теории графов. А именно, *поддеревом* дерева T будем называть подграф T', удовлетворяющий следующим условиям:

1) T' – дерево,

2) T' содержит корень дерева T,

3) если из v в v' идет дуга в дереве T, v и $v' \in T'$, то T' содержит все вершины, в которые из v идет дуга.

Определение. Пусть S – множество дизъюнктов, B – эрбрановский базис для S. *Семантическим деревом для S* называется корневое дерево, каждой дуге которого приписано непустое множество формул из B или их отрицаний так, что выполнены следующие условия.

1. Из любой вершины выходит конечное число дуг e_1, …, e_k; если c_i – конъюнкция литералов, приписанных дуге e_i, то $c_1 \vee c_2 \vee \ldots \vee c_k$ – тождественно истинная формула.

2. Для любой вершины v множество $I(v)$ не содержит противоположных литералов.

Рассмотрим примеры. Пусть S – множество дизъюнктов из примера 2 предыдущего параграфа, B – его эрбрановский базис. Напомним, что $B = \{P(a), Q(a), R(a)\}$. Для простоты в примерах вместо $P(a)$, $Q(a)$ и $R(a)$ будем писать просто P, Q, R. На рисунках 4.4 и 4.5 приведены примеры семантических деревьев для S. Семантическое дерево может быть бесконечным. На рисунке 4.6 приведен пример семантического дерева для множества дизъюнктов S из примера 1 §4. Напомним, что эрбрановский базис в этом случае есть $B = \{P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))). \ldots\}$.

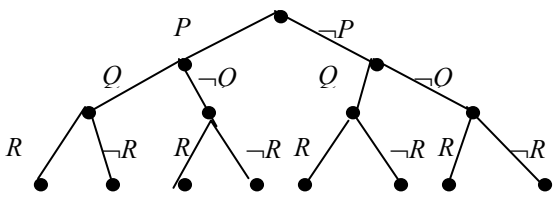


Рис. 4.4

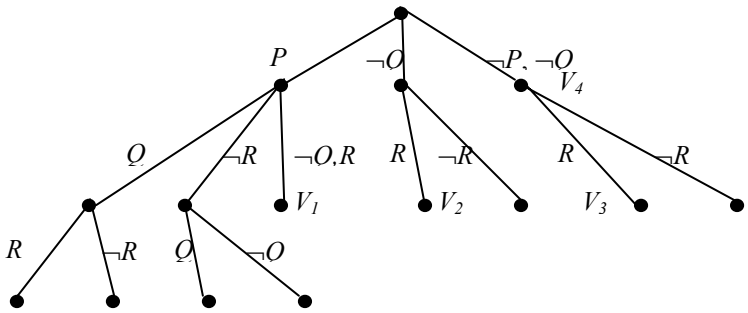


Рис. 4.5

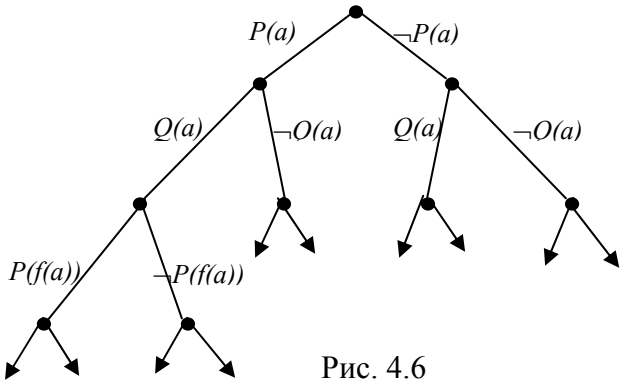


Рис. 4.6

Определение. Пусть B – эрбрановский базис множества дизъюнктов S. Семантическое дерево T называется *полным*, если для любого элемента A базиса B и любого максимального пути π множество $I(\pi)$ содержит либо A, либо $\neg A$.

Семантические деревья, изображенные на рис.4.4 и 4.6, являются полными, а семантическое дерево, изображенное на рис.4.5 – неполным.

Определение. Вершина v семантического дерева называется *опровергающей*, если $I(v)$ опровергает основной пример некоторого дизъюнкта из S. Вершина v называется *максимальной опровергающей*, если вершина v', предшествующая v, опровергающей не является.

Рассмотрим в качестве примера дерево, изображенное на рис.4.5. Напомним, что оно является семантическим деревом множества дизъюнктов $S = \{P(x), Q(x) \vee \neg R(y)\}$. Вершины v_1 и v_3 будут опровергающими вершинами, так как множество $I(v_1)$, равное $\{\neg Q(a) \& R(a), P(a)\}$, опровергает основной пример $Q(a) \vee \neg R(a)$ дизъюнкта $Q(x) \vee \neg R(y)$, а множество $I(v_3)$, равное $\{R(a), \neg P(a) \& \neg Q(a)\}$ опровергает основной пример $P(a)$ дизъюнкта $P(x)$. Вершина v_1 будет максимальной опровергающей, а вершина v_3 не будет максимальной опровергающей, потому что

опровергающей является предшествующая ей вершина v_4. Вершина v_2 опровергающей не является.

Определение. Вершина v семантического дерева называется *выводящей*, если все непосредственно следующие за ней вершины являются максимальными опровергающими.

Пусть $S = \{P(x), \neg P(x) \vee Q(f(y)), \neg Q(f(a))\}$ множество дизъюнктов примера 1 предыдущего параграфа. Дерево, изображенное на рис.4.7, является семантическим деревом для S. Вершина v этого дерева является выводящей вершиной, а никакие другие вершины выводящими не являются.

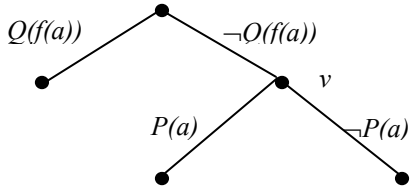


Рис.4.7

Определение. Семантическое дерево называется *замкнутым*, если каждый его лист является максимальной опровергающей вершиной.

Дерево, изображенное на рис.4.7, замкнуто, а деревья на рис.4.4 и 4.5 незамкнуты.

Следующее утверждение – знаменитая теорема математической логики, которая является основой многих алгоритмов доказательства теорем. Она называется теоремой Эрбрана.

Теорема 4.6. Множество дизъюнктов S невыполнимо тогда и только тогда, когда любое полное семантическое дерево множества S имеет конечное замкнутое поддерево.

Доказательство теоремы 4.6 использует известное в математике утверждение, которое называется леммой Кенига. Сформулируем ее.

Лемма. Если T – бесконечное дерево, из каждого узла которого выходит конечное число дуг. Тогда дерево T содержит бесконечный путь, начинающийся от корня.

Доказательство леммы Кенига приводить не будем. С ним можно познакомиться по книге К. Куратовского и А. Мостовского, указанной в списке литературы. (В этой книге доказывается несколько более общее утверждение, называемое теоремой Кенига.)

Приведем доказательство теоремы 4.6. Докажем вначале необходимость. Пусть множество дизъюнктов S невыполнимо и

T – полное семантическое дерево для S. Рассмотрим максимальный путь π в дереве T. По определению полного семантического дерева для каждой атомарной формулы A эрбрановского базиса B, либо A, либо $\neg A$ принадлежит $I(\pi)$. Это означает, что $I(\pi)$ есть H-интерпретация множества S. Поскольку S невыполнимо, то $I(\pi)$ опровергает основной пример D' некоторого дизъюнкта D из S. Дизъюнкт D' конечен, поэтому путь π должен проходить через максимальную опровергающую вершину дерева T. В каждом максимальном пути отметим такую вершину. Пусть T' – поддерево дерева T, листьями которого являются отмеченные вершины. В силу леммы Кенига, T' – конечное поддерево дерева T. Дерево T' по построению является замкнутым. Необходимость доказана.

Докажем достаточность. Пусть полное семантическое дерево T содержит конечное замкнутое поддерево T'. По определению поддерева (см. начало данного параграфа) отсюда следует, что каждый максимальный путь дерева T содержит опровергающую вершину. Множество всех максимальных путей полного семантического дерева исчерпывает все H-интерпретации множества S. Следовательно S ложно при всех H-интерпретациях. По теореме 4.5 S невыполнимо.

§6. Полнота метода резолюций в логике первого порядка

Параграф посвящен доказательству теоремы 4.3. Напомним ее формулировку.

Теорема 4.7. Множество дизъюнктов S невыполнимо тогда и только тогда, когда из S выводим пустой дизъюнкт.

Доказательство. Отметим вначале, что вывод из множества дизъюнктов S можно определить как последовательность дизъюнктов, каждый дизъюнкт которой принадлежит S или является резольвентой предыдущих дизъюнктов.

Пусть множество дизъюнктов S невыполнимо и $B = \{A_1, A_2, \ldots, A_n, \ldots\}$ – эрбрановский базис для S. Рассмотрим полное семантическое дерево T, изображенное на рис.4.8.

По теореме Эрбрана T содержит конечное замкнутое семантическое поддерево T'. Если T' состоит только из корня, то $\square \in S$, и поэтому $\square$ выводим из S. Предположим, что T' состоит не только из корня. Тогда T' имеет вершину v, потомки v_1 и v_2 которой являются максимальными опровергающими для множества S вершинами. Пусть

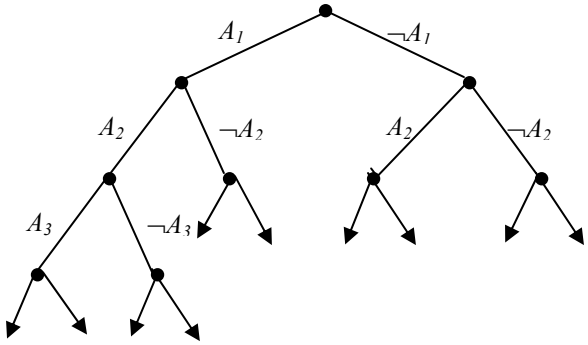


Рис. 4.8

$I(v) = \{ L_1, L_2, \ldots, L_k \}$,
$I(v_1) = \{ L_1, L_{2,} \ldots, L_k, A_{k+1} \}$,
$I(v_2) = \{ L_1, L_2, \ldots, L_k, \neg A_{k+1} \}$.

Существует дизъюнкт $D_1 \in S$ такой, что его основной пример D_1' опровергается в $I(v_1)$, и существует дизъюнкт $D_2 \in S$ такой, что его основной пример D_2' опровергается в $I(v_2)$. Так как дизъюнкты D_1' и D_2' не опровергаются в $I(v)$, то D_1' содержит $\neg A_{k+1}$, а D_2' – A_{k+1}. Применим к D_1' и D_2' правило резюлюций, отрезая литералы $\neg A_{k+1}$ и A_{k+1}, получим дизъюнкт D':

$$D' = (D_1' - \neg\neg A_{k+1}) \cup (D_2' - A_{k+1}).$$

Отметим, что дизъюнкт $D_1' - \neg\neg A_{k+1}$ ложен в $I(v)$, поскольку в противном случае, D_1' был бы истинен в $I(v_1)$. Аналогично заключаем, что $D_2' - A_{k+1}$ ложен в той же интерпретации $I(v)$.

Отсюда следует, что D' ложен при $I(v)$.

По лемме о подъеме (теорема 4.4) существует дизъюнкт D, который является резольвентой дизъюнктов D_1 и D_2. Ясно, что D опровергается в $I(v)$. Рассмотрим множество дизъюнктов $S \cup \{D\}$. Замкнутое семантическое дерево T'' для этого множества дизъюнктов можно получить вычеркиванием некоторых вершин (и идущих в них дуг) дерева T'. А именно, в дереве T' вычеркиваем все дуги и вершины, которые лежат ниже первой (на пути из корня) вершины, где дизъюнкт D' ложен. Полученное таким образом дерево T'' содержит меньше вершин, нежели дерево T', так как $v_1, v_2 \notin T''$.

Применим описанный выше процесс к T'', мы получим резольвенту дизъюнктов из $S \cup \{D\}$. Расширим множество $S \cup \{D\}$ за счет этой резольвенты, придем к конечному замкнутому дереву с меньшим числом вершин, нежели T''. В конце концов, получим замкнутое семантическое дерево, состоящее только из корня. Это возможно, лишь в случае, когда множество S, расширенное резольвентами, содержит пустой дизъюнкт, Следова-

тельно, □ выводим из S. Необходимость условия теоремы доказана.

Докажем достаточность. Пусть пустой дизъюнкт выводим из S, и $D_1, D_2, \ldots, D_n = \square$ – вывод из S. Предположим, что S выполнимо в некоторой интерпретации. Тогда, поскольку правило резолюций и правило склейки сохраняют истинность, то все дизъюнкты вывода, в том числе и пустой, являются истинными в этой интерпретации. Полученное противоречие доказывает, что S невыполнимо. □

§7. Стратегии метода резолюций

В множестве дизъюнктов существует, как правило, не одна пара дизъюнктов, к которым можно применить правило резолюций. Способ выбора дизъюнктов и литералов в них, к которым применяется правило резолюций (и правило склейки) для получения резольвенты, называется *стратегией* метода. В этом параграфе будет рассмотрено три стратегии: насыщения уровней, предпочтения более коротких дизъюнктов и вычеркивания. Достаточно полное описание известных стратегий содержится в книге Ч.Ченя и Р.Ли, приведенной в списке литературы.

Стратегия насыщения уровней. Наиболее простой с идейной точки зрения способ выбора дизъюнктов для получения резольвенты состоит в организации полного перебора возможных вариантов. Этот перебор можно организовать следующим образом. Пусть $S_0 = S$ – исходное множество дизъюнктов. Будем считать, что S_0 упорядочено. Пусть D_2 пробегает по порядку множество дизъюнктов S_0, начиная со второго. В качестве D_1 берем последовательно дизъюнкты из S_0, предшествующие D_2, начиная с первого, и формируем последовательность S_1, состоящую из всевозможных резольвент дизъюнктов D_1 и D_2. (Порядок на S_1 определяется порядком добавления дизъюнктов в S_1.) Предположим, что получены последовательности дизъюнктов $S_0, S_1, \ldots, S_{n-1}$ и $n>1$. Тогда последовательность S_n получается следующим образом. В качестве D_2 берутся по порядку дизъюнкты из S_{n-1}, а в качестве D_1 – дизъюнкты из $S_0 \cup S_1 \cup \ldots \cup S_{n-1}$, предшествующие D_2. Последовательность S_n будет состоять из всевозможных резольвент дизъюнктов D_1 и D_2. Процесс порождения резольвент прекращается, как только получается пустой дизъюнкт.

Описанная в предыдущем абзаце стратегия называется *стратегией насыщения уровней*. (Уровни – это последовательности $S_0, S_1, \ldots, S_n, \ldots$.) Проследим, как она работает на

примере множества дизъюнктов $S = \{X \vee Y, \neg X \vee \neg Y, X \vee Z, \neg X \vee Z, \neg Z\}$:

S_0: (1) $X \vee Y$,
(2) $\neg X \vee \neg Y$,
(3) $X \vee Z$,
(4) $\neg X \vee Z$,
(5) $\neg Z$,
S_1: (6) $Y \vee \neg Y_2$ (из (1) и (2)),
(7) $X \vee \neg X$ (из (1) и (2))
(8) $\neg Y \vee Z$ (из (2) и (3))
(9) $Y \vee Z$ (из (1) и (4)),
(10) Z (из (3) и (4)),
(11) X (из (3) и (5)),
(12) $\neg X$ (из (4) и (5)),
S_2: (13) $X \vee Y$ (из (1) и (6)),
(14) $\neg X \vee \neg Y$ (из (2) и (6)),
(15) $X \vee Y$ (из (1) и (7)),
(16) $\neg X \vee \neg Y$ (из (2) и (7)),
(17) $X \vee Z$ (из (3) и (7)),
(18) $\neg X \vee Z$ (из (4) и (7)),
(19) $X \vee Z$ (из (1) и (8)),
(20) $\neg Y$ (из (5) и (8)),
(21) $\neg Y \vee Z$ (из (6) и (8)),
(22) $\neg X \vee Z$ (из (2) и (9)),
(23) Y (из (5) и (9)),
(24) $Y \vee Z$ (из (6) и (9)),
(25) Z (из (8) и (9)),
(26) □ (из (5) и (10)).

Мы видим, что порождено много лишних дизъюнктов. Так, (6) и (7) – тождественно истинные дизъюнкты. Удаление или добавление тождественно истинного дизъюнкта не влияет на выполнимость множества дизъюнктов, поэтому такие дизъюнкты должны быть удалены из вывода. Далее, некоторые дизъюнкты порождаются неоднократно, например, $X \vee Y$, $\neg X \vee \neg Y$, $Y \vee Z$. Это означает, что выбором дизъюнктов для получения резольвенты надо управлять.

Стратегия предпочтения (более коротких дизъюнктов). Эта стратегия является модификацией предыдущей: сначала в качестве D_2 берется самый короткий дизъюнкт из S_{n-1} (если таких несколько, то они перебираются по порядку), затем более длинные и т.д. Аналогичные условия налагаются и на D_1. Такая стратегия в применении к тому же множеству дизъюнктов S дает следующую последовательность дизъюнктов:

S_0: (1) $X \vee Y$,
(2) $\neg X \vee \neg Y$,
(3) $X \vee Z$,
(4) $\neg X \vee Z$,
(5) $\neg Z$
S_1: (6) X (из (3) и (5)),
(7) $\neg X$ (из (4) и (5)),
(8) $Y \vee \neg Y$ (из (1) и (2)),
(9) $X \vee \neg X$ (из (1) и (2)),
(10) $\neg Y \vee Z$ (из (2) и (3)),
(11) $Y \vee Z$ (из (1) и (4))
(12) Z (из (3) и (4)),
S_2: (13) $\neg Y$ (из (2) и (6)),
(14) Z (из (2) и (6)),
(15) Y (из (1) и (7)),
(16) Z (из (3) и (7)),
(17) □ (из (6) и (7)).

Вывод оказался короче, чем в предыдущем примере, но по-прежнему содержит повторяющиеся и тождественно истинные дизъюнкты. Свободным от этих недостатков является вывод, полученный в соответствии со стратегией вычеркивания. Для ее описания введем вначале следующее понятие.

Определение. Дизъюнкт D называется *расширением* дизъюнкта C, если существует подстановка σ такая, что $\sigma(C) \subseteq D$.

Для логики высказываний это означает, что просто $D = C \vee D'$ (при некоторой перестановке литералов). В случае логики предикатов ситуация не столь проста. Например, $D = Q(a) \vee P(b,y) \vee R(u)$ есть расширение дизъюнкта $C = P(x,y) \vee Q(z) \vee Q(v)$.

Стратегия вычеркивания. Эта стратегия, как и стратегия предпочтения, является модификацией стратегии насыщения уровней. Она применяется следующим образом: после того, как получена очередная резольвента D дизъюнктов D_1 и D_2, проверяется, является ли она тождественно истинной формулой или расширением некоторого дизъюнкта C из $S_0 \cup \ldots \cup S_{n-1}$, и в случае положительного ответа D вычеркивается, т.е. не заносится в проследовательность S_n.

Применение стратегии к прежнему множеству дизъюнктов дает следующую последовательность дизъюнктов:

S_0: (1) $X \vee Y$,
(2) $\neg X \vee \neg Y$,
(3) $X \vee Z$,
(4) $\neg X \vee Z$,
(5) $\neg Z$,
S_1: (6) $\neg Y \vee Z$ (из (2) и (3)),
(7) $Y \vee Z$ (из (1) и (4)),
(8) Z (из (3) и (4)),
(9) X (из (3) и (5)),
(10) $\neg X$ (из (4) и (5)),
(11) $\neg Y$ (из (5) и (6)),
(12) Y (из (5) и (7)),
(13) $\square$ (из (5) и (8)).

Рассмотренные стратегии являются *полными* в том смысле, что если множество дизъюнктов S невыполнимо, то из S пользуясь стратегией можно вывести пустой дизъюнкт. Для первых двух стратегий это достаточно очевидно. Полнота стратегии вычеркивания следует из того, что если D и C – дизъюнкты из S и D – расширение C, то множество S невыполнимо в том и только в том случае, когда невыполнимо множество $S \setminus \{D\}$.

§8. Применение метода резолюций

Мы рассмотрим здесь применение метода резолюций в доказательстве теорем и при планировании действий.

Доказательство теорем. Применим метод резолюций в доказательстве одной простой теоремы из теории групп.

в качестве исходной возьмем следующую аксиоматику теории групп:

F_1: $(\forall x,y,z)[(xy)z = x(yz)]$,

F_2: $(\forall x,y)(\exists z)(zx = y)$,

F_3: $(\forall x,y)(\exists z)(xz = y)$.

Предположим, что нам надо доказать теорему G: $(\exists x)(\forall y)(yx = y)$, т.е. что в группе существует правая единица.

Наша задача – установить, что формула G есть логическое следствие формул F_1, F_2, F_3. Прежде, чем решать эту задачу, перейдем к другой сигнатуре. Введем символ трехместного предика P, который интерпретируется следующим образом:

$P(x, y, z)$ означает, что $xy = z$.

В новой сигнатуре формулы F_1, F_2, F_3 и G запишутся так:

$F_1' = (\forall x,y,z)[P(x, y, u)\ \&\ P(y, z, v)\ \&\ P(x, v, w) \rightarrow P(u, z, w)]$,

$F_2' = (\forall x,y)(\exists z)P(z, x, y)$,

$F_3' = (\forall x,y)(\exists z)P(x, z, y)$,

$G' = (\exists x)(\forall y)P(y, x, y)$.

Сформируем множество $T = \{F'_1, F'_2, F'_3, \neg G'\}$, каждую из формул множества T приведем к сколемовской нормальной форме и удалим кванторы общности (конъюнкция в сколемовских нормальных формах этих формул не появится). Получим множество дизъюнктов D_1, D_2, D_3, D_4:

$D_1 = \neg P(x, y, u) \vee \neg P(y, z, v) \vee \neg P(x, v, w) \vee P(u, z, w)$,

$D_2 = P(f(x,y), x, y)$,

$D_3 = P(x, g(x,y), y)$,

$D_4 = \neg P(h(x), x, h(x))$.

Построим вывод пустого дизъюнкта из множества дизъюнктов D_1, …, D_4. Пусть эти дизъюнкты – первые дизъюнкты вывода. Заменив переменные в дизъюнкте D_2, получим дизъюнкт $D_2' = P(f(x',y'), x', y')$. Литералы $P(x, y, u)$ и D_2' унифицируются подстановкой $\sigma_1 = \{x = f(x',y'), y = x', u = y'\}$. Применим правило резолюций к D_1 и D_2' (и указанным литералам), получим дизъюнкт

$D_5 = \neg P(x', z, v) \vee \neg P(f(x',y'), v, w) \vee P(y', z, w)$.

Далее, литерал $P(f(x',y'), v, w)$ и D_2 унифицируются подстановкой $\sigma_2 = \{x' = x, y' = y, v = x, w = y\}$. Правило резолюций, примененное к D_5 и D_2, дает дизъюнкт

$D_6 = \neg P(x, z, x) \vee P(y, z, y)$.

Резольвентой дизъюнктов D_3 и D_6 будет дизъюнкт

$D_7 = P(y, g(y',y'), y)$.

Для получения этой резольвенты заменим переменные в D_3, получим $D_3 = P(x', g(x',y'), y')$ и используем подстановку $\sigma_3 = \{x = y', z = g(y',y')\}$. Наконец, из дизъюнктов D_4 и D_7 с помощью подстановки $\sigma_4 = \{y = h(g(y',y')), x = g(y',y')\}$ получаем пустой дизъюнкт.

Планирование действий. Отметим вначале одно свойство метода резюлюций. Пусть сигнатура τ состоит из двух символов двухместных предикатов P и Q, которые интерпретируются следующим образом: $P(x, y)$ означает, что x –сын y, $Q(x, z)$ означает, что x – внук z.Рассмотрим формулы:

$F_1 = (\forall x,y,z)[P(x, y) \,\&\, P(y, z) \rightarrow Q(x, z)]$,

$F_2 = (\forall x)(\exists y)P(x, y)$,

$G = (\forall x)(\exists z)Q(x, z)$,

смысл которых достаточно ясен.

Используя метод резолюций, покажем, что G есть логическое следствие F_1 и F_2. Приведем формулы F_1, F_2 и $\neg G$ к сколемовской нормальной форме, получим дизъюнкты:

$D_1 = \neg P(x, y) \vee \neg P(y, z) \vee Q(x, z)$, $D_2 = P(x, f(x))$, $D_3 = \neg Q(a, z)$.

Вывод пустого дизъюнкта получается довольно просто:

$D_4 = \neg P(a, y) \vee \neg P(y, z)$ (D_1 и D_3, $\{x = a\}$),

$D_5 = \neg P(f(a), z)$ (D_2 и D_4, $\{x = a, y = f(a)\}$),

$D_6 = \square$ (D_2 и D_5, $\{x = f(a), z = f(a)\}$).

Подстановка $z = f(f(a))$ означает, что дед элемента a есть отец отца элемента a. Таким образом, метод резолюций не только устанавливает факт логического следствия формулы G из формул F_1 и F_2, но еще и “подсказывает”, как по данному x получить z такой, чтобы формула $Q(x, z)$ была истинна.

Довольно часто интересующая нас переменная участвует не в одной подстановке, как в этом примере переменная z, а в нескольких. Для того, чтобы отследить все подстановки, в которых может изменить значение переменная, к формуле $\neg G$ добавляют литерал ответа $ANS(z)$ и заканчивают вывод не пустым дизъюнктом, а литералом ответа.

В качестве примера использования метода резолюций в задачах планирования действий рассмотрим известную в теории искусственного интеллекта задачу об обезьяне и бананах. В задаче говорится об обезьяне, которая хочет съесть бананы, подвешенные к потолку комнаты. Рост обезьяны недостаточен,

чтобы достать бананы. Однако в комнате есть стул, встав на который обезьяна может достать бананы. Какие ей надо совершить действия, чтобы достать бананы?

Задачу формализуем следующим образом. Комнату с находящимися в ней обезьяной, стулом и бананами будем называть *предметной областью*. Совокупность мест, где находятся в данный момент в комнате обезьяна, стул и бананы, будем называть *состоянием предметной области*. Рассмотрим два предиката $P(x, y, z, s)$ и $R(s)$. Пусть

$P(x, y, z, s)$ означает, что в состоянии s обезьяна находится в точке x, стул – в y, бананы – в z,

$R(s)$ означает, что в состоянии s обезьяна взяла бананы.

Возможности обезьяны формализуем следующим образом. Введем три функции, которые принимают значения в множестве состояний:

ИДТИ(x, y, s) – состояние, которое получится из s, если обезьяна из точки x перешла в y,

НЕСТИ(x, y, s) – состояние, которое получится из s, если обезьяна перенесла стул из точки x в y,

БРАТЬ(s) – состояние, которое получится из s, если обезьяна взяла бананы.

Условия задачи запишутся в виде следующих формул:

$F_1 = (\forall x,y,z,s))[P(x, y, z, s) \to P(u, y, z, \text{ИДТИ}(x, u, s))]$,

$F_2 = (\forall x,z,s)[P(x, x, z, s) \to P(u, u, s, \text{НЕСТИ}(x, u, s))]$,

$F_3 = (\forall x)[P(x, x, x, s) \to R(\text{БРАТЬ}(s))]$.

Пусть в начальном состоянии s_0 обезьяна находилась в точке a, стул – в точке b, бананы – в точке c. Следовательно, к написанным формулам надо добавить формулу

$F_4 = P(a, b, c, s_0)$.

Надо показать, что формула $G = (\exists s)R(s)$ есть логическое следствие формул F_1, F_2, F_3, F_4. Из множества формул $\{F_1, F_2, F_3, F_4, \neg G\}$ получим множество дизъюнктов $D_1 - D_5$ (к дизъюнкту, полученному из $\neg G$ добавлен литерал ответа $ANS(s)$):

$D_1 = \neg P(x, y, z, s) \vee P(u, y, z, \text{ИДТИ}(x, u, s))$,

$D_2 = \neg P(x, x, z, s,) \vee P(u, u, z, \text{НЕСТИ}(x, u, s))$,

$D_3 = \neg P(x, x, x, s) \vee R(\text{БРАТЬ}(s))$,

$D_4 = P(a, b, c, s_0)$,

$D_5 = \neg R(s) \vee ANS(s)$.

Последовательность дизъюнктов $D_1 - D_5$ продолжаем до вывода литерала ответа:

$D_6 = \neg P(x, x, x, s) \vee ANS(\text{БРАТЬ}(s))$ полчается из D_3 и D_5,

$D_7 = \neg P(x, x, u, s) \vee ANS(\text{БРАТЬ}(\text{НЕСТИ}(x, u, s)))$ получается из D_2 и D_6,

$D_8 = \neg P(x, y, z, s) \vee ANS(\text{БРАТЬ}(\text{НЕСТИ}((y, z,$
$\text{ИДТИ}(x, y, s))))$ почается из D_1 и D_7,

$D_9 = ANS(\text{БРАТЬ}(\text{НЕСТИ}(b, c, \text{ИДТИ}((a, b, s_0))))$ получается из D_4 и D_8.

Итак, для того, чтобы обезьяне взять бананы, надо сначала из точки a идти в точку b, затем из точки b нести стул в точку c и в точке c, встав на стул, взять бананы.

§9. Метод резолюций и логическое программирование

Наиболее распространенная в наше время технология решения задач на компьютере состоит в том, что вначале программист должен разработать алгоритм решения задачи, а затем записать его на определенном формальном языке. Эти этапы вместе с последующей отладкой требуют от программиста значительных затрат времени и достаточно высокой квалификации. На большинстве этапов решения задачи имеется сильная зависимость от внутренних механизмов компьютера, на котором это решение будет осуществлено. Отмеченные (а также и некоторые другие) недостатки алгоритмической технологии программирования стимулировали поиск новых возможностей. Осознание того, что вычисление – частный случай логического вывода, а алгоритм – формальное задание функции, привело к идее логического программирования. Суть этой идеи состоит в том, чтобы компьютеру предлагать не алгоритмы, а описания предметной области задачи и саму задачу в виде некоторой аксиоматической системы, а решение задачи в – виде вывода в такой системе. От программиста при таком подходе требуется описать с достаточной степенью полноты предметную область и формулировку задачи на языке этой системы, а поиск вывода, приводящего к решению задачи, поручается компьютеру.

Конечно, при таком широком понимании логического программирования, любая программная система, поддерживающая ту или иную логическую модель, представляет собой фактически и систему логического программирования. Разница может возникнуть лишь тогда, когда в инструментальной системе программист определяет в существенной степени способ обработки описания предметной области и задачи. На самом деле, логическое программирование понимается, как правило, в более узком смысле.

Логическая программа представляет собой совокупность формул логики предикатов одного из следующих видов

(1) $p(t_1, \ldots, t_k).$,

(2) $q(s_1, \ldots, s_l) :- q_1(s_1, \ldots, s_l), \ldots, q_n(s_1, \ldots, s_l).$,

где $p(t_1, \ldots, t_k)$, $q(s_1, \ldots, s_l)$, $q_1(s_1, \ldots, s_l)$, …, $q_n(s_1, \ldots, s_l)$ – атомарные формулы логики первого порядка, буквы t и s с индексами – термы. Синтаксис языка логического программирования требует, чтобы в конце каждого выражения ставилась точка. Формулы первого вида называются *фактами*, а второго – *правилами*. Формула $q(s_1, \ldots, s_l)$ называется заголовком правила (2). Выполнение программы инициализируется запросом – формулой вида

(3) $r_1(u_1, \ldots, u_m), \ldots, r_{n0}(u_1, \ldots, u_m).$,

где $r_j(u_1, \ldots, u_m)$ $(1 \le j \le n_0)$ – атомарные логики первого порядка, буквы u с индексами – термы.

Мы описали синтаксис основных конструкций логического программирования. Семантика обычно представляется в двух видах – логическая семантика и процедурная семантика.

Введем сначала *логическую семантику*. Каждому факту (1) поставим в соответствие формулу вида

$(1')$ $F = (\forall x^*)p(t_1, \ldots, t_k)$,

где кванторы общности навешаны на все переменные атомарной формулы (1). (Кроме переменных, в термах могут быть, разумеется, константы.) Правилу (2) поставим в соответствие формулу вида

$(2')$ $G = (\forall x^*)[q_1(s_1, \ldots, s_l)\ \&\ldots\&\ q_n(s_1, \ldots, s_l) \to q(s_1, \ldots, s_l)]$,

где кванторы общности, как и выше, навешаны на все переменные. Запрос (3) получит в соответствие формулу

$(3')$ $H = (\exists x^*)[r_1(u_1, \ldots, u_m)\ \&\ldots\&\ r_{n0}(u_1, \ldots, u_m)]$,

где квантор существования связывает все переменные. Пусть $F_1, \ldots, F_a$ – формулы, соответствующие всем фактам, $G_1, \ldots, G_b$ – всем правилам. Тогда значение пары (программа, запрос) в логической семантике есть утверждение о том, что формула H есть логическое следствие формул $F_1, \ldots, F_a, G_1, \ldots, G_b$.

Операционная семантика – действия компьютера при ответе на запрос. Введем ее на примере следующей программы.

(1) $r(a, b).$,

(2) $q(b, g(c)).$,

(3) $p(x, f(y)) :- r(x, z),\ q(z, f(y)).$,

(4) $p(x, f(y)) :- r(x, z), q(z, g(y)).$,

(5) $r(x, z) :- q(f(x), g(z))$.

(Здесь a, b, c, d – константы, x, y, z – переменные.) Номера в скобках не являются синтаксической конструкцией логического программирования, они проставлены для удобства ссылок.

Предположим, что запрос есть

(6) $p(u, f(v))$.

При вычислении ответа на этот запрос, интерпретатор формулирует цель $p(u, f(v))$ и пытается достичь ее, унифицируя цель с фактами. В нашем случае цель $p(u, f(v))$ не унифицируется ни с одним из фактов. Тогда интерпретатор пытается ее унифицировать с заголовком одного из правил. Это можно сделать с заголовком правила (3) с помощью подстановки $\sigma = (u = x, v = y)$. Запрос (6) принимает следующий вид

(6′) $r(x, z), q(z, f(y))$

и формируется цель $r(x, z)$. Она достигается унификацией с первым фактом подстановкой $\sigma_1 = (x = a, z = b)$ и интерпретатор пытается достичь цели $q(b, f(y))$, но эта цель не унифицируется ни с одним из фактов, ни с заголовками правил. Следовательно, цель $q(b, f(y))$ недостижима и происходит возврат к запросу (6′) и цели $r(x, z)$. Делается попытка достичь этой цели при помощи правила (5), но эта попытка также неудачна. Происходит возврат к запросу (6) и цели $p(u, f(v))$. (См. рис.4.9, где цели подчеркнуты.)

Правило (3) “отработано”. Интерпретатор унифицирует цель с заголовком правила (4) той же подстановкой σ. Запрос принимает вид

(6″) $r(x, z), q(z, g(y))$,

а целью становится $r(x, z)$. Цель унифицируется с первым фактом подстановкой σ_1 и ставится новая цель $q(b, g(y))$, которая унифицируется со вторым фактом подстановкой $\sigma_2 = (y = c)$. Исходная цель оказалась достижимой при результирующей подстановке $\sigma_3 = (u = a, v = c)$. Интерпретатор при этом может закончить работу и выдать подстановку σ_3. Возможен и другой режим работы интерпретатора, при котором он пытается найти все подстановки, ведущие к достижению цели.

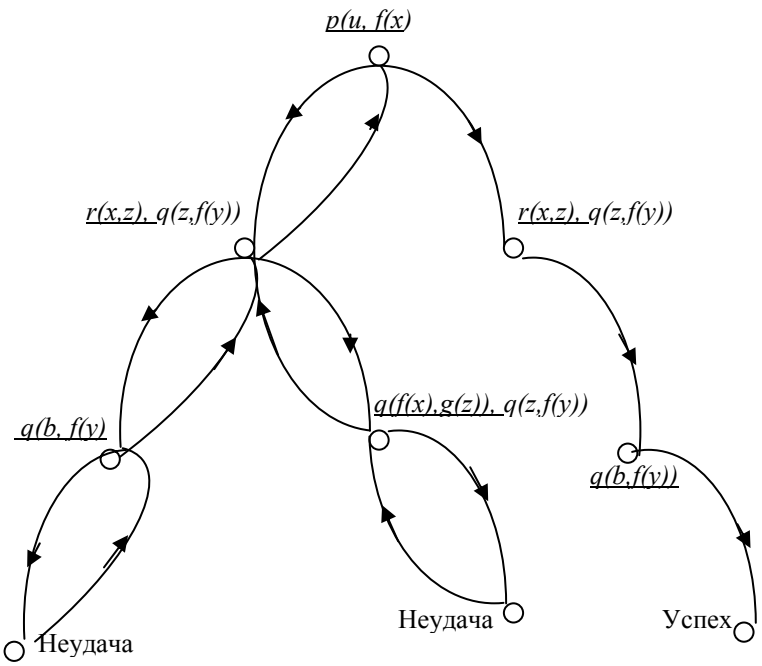


Рис. 4.9

Убедимся в том, что интерпретатор при поиске ответа на запрос фактически строит вывод с помощью правила резолюций.

Логическая семантика программе (1) – (5) ставит в соответствие следующие формулы:

$F_1 = r(a, b)$,

$F_2 = q(b, g(c))$,

$G_1 = (\forall x,y,z)[r(x, z) \ \& \ q(z, f(y)) \rightarrow p(x, f(y))]$,

$G_2 = (\forall x,y,z)[r(x, z) \ \& \ q(z, g(y)) \rightarrow p(x, f(y))]$,

$G_3 = (\forall x,z)[q(f(x), g(z)) \rightarrow r(x, z)]$,

а запросу (6) – формулу

$H = (\exists u,v)p(u, f(v))$.

Эта семантика, напомним, состоит в том, что H есть логическое следствие множества формул $\{F_1, F_2, G_1, G_2, G_3\}$.

Как будет применяться в этой ситуации метод резолюций? Вначале будет составлено множество формул $T = \{F_1, F_2, G_1, G_2, G_3, \neg H\}$. Затем каждая из формул множества T будет приведена к сколемовской нормальной форме, из которой будет получено множество дизъюнктов S. В нашем случае множество S состоит из дизъюнктов $D_1 - D_6$:

$D_1 = r(a, b)$,

$D_2 = q(b, g(c))$,

$D_3 = \neg r(x, z) \vee \neg q(z, f(g) \vee p(x, f(y)$,

$D_4 = \neg r(x, z) \vee \neg q(z, g(y)) \vee p(x, f(y))$,

$D_5 = \neg q(f(x), g(z)) \vee r(x, z)$,

$D_6 = \neg p(u, f(v))$.

Отметим, что дизъюнкт D_6 получился из формулы $\neg H$.

Вывод пустого дизъюнкта будем осуществлять в соответствии с процедурной семантикой. Правило резолюций будет применяться так, что одним из исходных (для правила) дизъюнктов будет дизъюнкт D_6 или его потомки. Попытка получить пустой дизъюнкт за один шаг, т.е. правило резолюций применить к паре $\{D_1, D_6\}$ или $\{D_2, D_6\}$, не приводит к успеху. Применим тогда правило резолюций к паре $\{D_3, D_6\}$, получим резольвенту

$D_7 = \neg r(x, z) \vee \neg q(z, f(y))$

с помощью подстановки $\sigma = (u = x, v = y)$. Попробуем теперь в D_7 "уничтожить" литерал $\neg r(x, z)$. Это можно сделать с помощью дизъюнктов D_1 и D_5. Резольвентой дизъюнктов D_1 и D_7 будет дизъюнкт

$D_8 = \neg q(b, f(y))$,

единственный литерал, которого нельзя "уничтожить" ни одним из дизъюнктов $D_1 - D_5$. То же самое можно сказать и о резольвенте дизъюнктов D_5 и D_7.

Мы фактически доказали, что из множества дизъюнктов $\{D_1, \ldots, D_5, D_7\}$ пустой дизъюнкт невыводим (см. пути в графе на рис. 4.9, приводящие к неудаче).

Возьмем теперь резольвенту дизъюнктов D_4 и D_6:

$D_9 = \neg r(x, z) \vee \neg q(z, g(y))$.

Литерал $\neg r(x, z)$ дизъюнкта D_9 "уничтожим" с помощью D_1 и подстановки $\sigma_1 = (x = a, z = b)$, получим дизъюнкт

$D_{10} = \neg q(b, g(y))$.

Резольвента дизъюнктов D_2 и D_{10} дает пустой дизъюнкт, при этом используется подстановка $\sigma_2 = (y = c)$. Мы получили вывод пустого дизъюнкта из множества дизъюнктов $\{D_1, \ldots, D_6\}$ (см. путь в графе на рис. 4.9, приводящий к успеху).

Итак, интерпретатор при поиске ответа на запрос строит резолютивный вывод.

Задачи

1. Доказать с помощью метода резолюций, что формула G есть логическое следствие формул $F_1, \ldots F_n$:

а) $F_1 = X \vee Y$, $F_2 = X \to Z$, $G = (Y \to Z) \to Z$;

б) $F_1 = X$, $F_2 = X \& Y \to Z$, $G = Y \to Z$;

в) $F_1 = X \to Y \vee Z$, $F_2 = Z \to W$, $F_3 = \neg W$, $G = X \to Y$;

г) $F_1 = X \vee Y \vee \neg Z$, $F_2 = X \to X1$, $F_3 = Y \to Y1$, $F_4 = Z$, $G = X1 \vee Y1$;

д) $F_1 = X\&Y \to \neg X\&Z$, $F_2 = \neg(X\&\neg Y) \vee Z$, $G = X \to Z$;

е) $F_1 = X \to [\neg Y \ \& \ (\neg Y \to Z)]$, $F_2 = (X \to \neg Y) \ \& \ \neg(\neg X\&\neg W)$, $G = W \vee Z$.

2. Запишите следующие рассуждения в виде последовательности формул логики высказываний. Если рассуждение логично, то докажите это методом резолюций; если нелогично, то постройте интерпретацию, при которой посылки истинны, а заключение ложно.

2.1. Если конгресс отказывается принять новые законы, то забастовка не будет окончена, кроме случая, когда она длится более месяца и президент фирмы уйдет в отставку. Допустим, что конгресс отказывается действовать и забастовка заканчивается. Следовательно, забастовка длилась более месяца.

2.2. Если подозреваемый совершил эту кражу, то она была тщательно подготовлена или он имел соучастника. Если бы кража была тщательно подготовлена, то, если бы он имел соучастника, был бы украден дорогой компьютер. Компьютер остался на месте. Следовательно, подозреваемый невиновен.

2.3. Рассуждение из задачи 2.1 главы 1.

2.4. Рассуждение из задачи 2.2 главы 1.

2.5. Рассуждение из задачи 2.3 главы 1.

3. Пусть $\sigma = \{x_1 = f(x_2), x_2 = d, x_3 = f(x_1)\}$ – подстановка, $F = P(x_1, f(x_2)) \vee \neg Q(x_3)$, $G = P(x_3, x_2) \vee R(x_1, g(x_2))$. Найти $\sigma(F)$ и $\sigma(G)$.

4. Определить, унифицируемы ли следующие множества атомарных формул:

а) $M = \{P(a, y, y), P(z, x, f(x))\}$,

б) $M = \{P(x, y, z), P(u, h(y,y), y), P(a, b, c)\}$,

в) $M = \{P(a, f(x), g(x,y)), P(u, y, g(f(a),h(y))\}$.

5. Определить, имеют ли следующие дизъюнкты склейки. Если имеют, то найти их:

а) $P(x) \vee P(a) \vee Q(f(x))$

б) $P(x) \vee Q(f(x)) \vee P(f(x))$,

в) $P(a) \vee P(b) \vee P(x)$.

6. Найти все возможные резольвенты (если они есть) следующих пар дизъюнктов:

а) $C = \neg P(x) \vee Q(x, b)$, $D = P(a) \vee Q(a, b)$,

б) $C = \neg P(x) \vee Q(x, x)$, $D = \neg Q(a, f(a))$,

в) $C = \neg P(x) \vee \neg P(a) \vee \neg Q(y, f(b))$, $D = P(u) \vee Q(c, f(v))$.

7. Найти H_0, H_1, H_2 (см.§4), если

а) $S = \{P(f(x), a), \neg P(y, g(x))\}$,
б) $S = \{P(a, g(x)), P(u, b)\}$,
в) $S = \{P(x, y), P(h(u, v), z)\}$.

8. Построить замкнутое семантическое дерево, если
а) $S_1 = \{P, \neg P \vee Q, \neg Q\}$,
б) $S_2 = \{P(x), \neg P(f(y))\}$,
в) $S_3 = \{P, Q \vee R, \neg P \vee \neg Q, \neg P \vee \neg R\}$.

9. Найти невыполнимое множество S_i' основных примеров дизъюнктов множества S_i $(1 \le i \le 3)$, если
а) $S_1 = \{P(x, a, g(x,b)), \neg P(f(y), z, g(f(a),b)\}$,
б) $S_2 = \{P(a), \neg D(y) \vee L(a,y), \neg P(x) \vee \neg Q(y) \vee \neg L(x,y), D(b), Q(b)\}$,
в) $S_3 = \{\neg K(x,y) \vee \neg L(y) \vee M(f(x)), \neg K(x,y) \vee \neg L(y) \vee M(z), \neg M(z), K(a, b), L(b)\}$.

10. Доказать с помощью метода резолюций, что формула G есть логическое следствие формул $F_1, \ldots, F_k$:
а) $F_1 = (\forall x)(P(x) \to Q(x) \,\&\, R(x))$, $F_2 = (\exists x)(P(x) \,\&\, T(x))$, $G = (\exists x)(R(x) \,\&\, T(x))$,
б) $F_1 = (\forall x)[(\exists y)(M(y) \,\&\, S(x, y)) \to (\exists z)(I(z) \,\&\, E(x,z))]$,
$G = \neg(\exists x)I(x) \to (\forall u)(\forall v)(S(u, v) \to \neg M(v))$.
в) $F_1 = (\forall x)[P(x) \to (\exists y)(Q(y) \,\&\, S(x, y))]$, $F_2 = (\exists x)[R(x) \vee (\forall y)\neg(\neg Q(y) \to S(x, y))]$, $F_3 = (\exists x)P(x)$, $G = (\exists x)(\neg P(x) \vee R(x))$.

11. Используя метод резолюций в логике предикатов, докажите логичность следующих рассуждений.

11.1. Все студенты нашей группы – члены клуба “Спартак”. А каждый член клуба “Спартак” занимается спортом. Следовательно, все студенты нашей группы занимаются спортом.

11.2. Все студенты нашей группы – болельщики “Спартака”, а некоторые занимаются спортом. Следовательно, некоторые из болельщиков “Спартака” занимаются спортом.

11.3. Некоторые пациенты уважают всех докторов. Ни один пациент не уважает знахаря. Следовательно, никакой доктор не является знахарем.

11.4. Если деталь обрабатывалась на токарном станке, то она обрабатывалась и на фрезерном. Деталь $D1$ обрабатывалась на токарном станке $C1$. Следовательно, она обрабатывалась на фрезерном станке.

12. Будут ли логичны следующие рассуждения? Если логичны, то доказать это методом резолюций. Если нет, то по-

строить интерпретацию, при которой посылки истинны, а заключение ложно.

12.1. Если кто-нибудь может решить эту задачу, то и любой математик может ее решить. Олег – математик, но не может решить эту задачу. Следовательно, задачу не сможет решить никто.

12.2. Всякий, кто может решить эту задачу – математик. Олег – математик, но не может решить эту задачу. Следовательно, задачу не может решить никто.

12.3. Если кто-нибудь может решить эту задачу, то и какой-нибудь математик может ее решить. Олег – математик, но не может решить задачу. Следовательно, задачу не может решить никто.

12.4. Некоторые из первокурсников знакомы со всеми спортсменами института. Ни один первокурсник не знаком ни с одним любителем подледного лова. Следовательно, ни один спортсмен не является любителем подледного лова.

12.5. Каждый из первокурсников знаком с кем-либо из спортсменов. Некоторые из первокурсников не знакомы ни с одним любителем подледного лова. Следовательно, ни один спортсмен не является любителем подледного лова.

Ответы, указания и решения

1. Приведем решение задачи 1д). Рассмотрим множество $T = \{F_1, F_2, \neg G\}$. Каждую из формул этого множества приведем к КНФ, получим соответственно $\neg X \vee \neg Y$, $\neg X \vee Y \vee Z$, $X \& \neg Z$. Тогда S равно $\{\neg X \vee \neg Y, \neg X \vee Y \vee Z, X, \neg Z\}$, а вывод пустого дизъюнкта есть $\neg X \vee \neg Y$, X, $\neg Y$, $\neg X \vee Y \vee Z$, $\neg X \vee Z$, Z, $\neg Z$, □.

2. Приведем решение задачи 2.1. Рассмотрим высказывания: X = ”конгресс отказывается принять законы”, Y = ”забастовка не будет закончена”, Z = ”забастовка длится более месяца”, U = ”президент фирмы уйдет в отставку”. Тогда предложения рассуждения 2.1 можно представить формулами $F_1 = X \to Y \vee (Z \& U)$, $F_2 = X \& \neg Y$, $G = Z$. Сформируем множество $T = \{F_1, F_2, \neg G\}$, каждую из формул приведем к КНФ и получим множество дизъюнктов $S = \{\neg X \vee Y \vee Z, \neg X \vee Y \vee U, X, \neg Y, \neg Z\}$. Легко выдеть, что из S выводим □. Следовательно рассуждение логично.

В задачах 2.4 и 2.5 рассуждение логично, а в задачах 2.2 и 2.3 – нелогично.

4. Указание. Воспользоваться алгоритмом унификации.

5. Дизъюнкт 5а) имеет склейку, остальные не имеют.

6. Дизъюнкты в задаче 6б) не имеют резольвент, в задаче 6а) дизъюнкты имеют одну резольвенту $Q(a, b)$, в задаче 6в) – пять $\neg P(a)\vee\neg Q(y, f(b))\vee Q(c, f(v))$, $\neg P(x)\vee\neg Q(y, f(b))\vee Q(c, f(v))$, $\neg P(x)\vee\neg P(a)\vee P(u)$, $\neg Q(y, f(b))\vee Q(c, f(v))$, $\neg P(a)\vee P(u)$.

7. В задаче 7б) $H_0 = \{a, b\}$, $H_1 = \{a, b, g(a), g(b)\}$, $H_2 = \{a, b, g(a), g(b), g(g(a)), g(g(b))\}$.

9. На рис. 4.9 изображены семантические деревья для S_1, на рис. 4.11 – для S_3. На рис. 4.10 изображено семантическое дерево для S_2. (Эти множества дизъюнктов имеют и другие семантические деревья.)

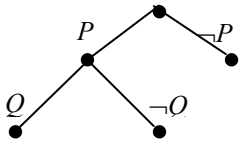


Рис. 4.9

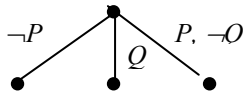
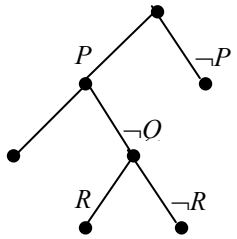


Рис. 4.10



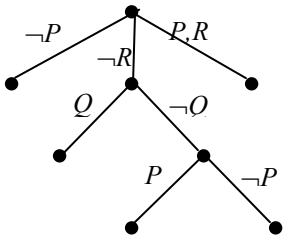


Рис. 4.11

9. Невыполнимое множество основных примеров есть

$S_1' = \{P(f(a), a, g(f(a), b)), \neg P(f(a), a, g(f(a), b))\}$,

$S_2' = \{P(a), \neg D(b)\vee L(a, b), \neg P(a)\vee\neg Q(b)\vee\neg L(a, b), D(b), Q(b)\}$,

$S_3' = \{\neg K(a, b)\vee\neg L(b)\vee M(f(a)), \neg M(f(a)), K(a, b), L(b)\}$.

10. Приведем решение задачи 10в). Составим множество $\{F_1, F_2, F_3, \neg G\}$. Каждую из формул приведем к сколемовской нормальной форме. Получим соответственно формулы

$H_1 = (\forall x)[(\neg P(x) \vee Q(f(x)) \,\&\, (\neg P(x) \vee S(x, f(x))]$,

$H_2 = (\forall y)[(R(a) \vee \neg Q(y)) \,\&\, (R(a) \vee \neg S(a, y))]$,

$H_3 = P(b)$,

$H_4 = (\forall x)(P(x) \,\&\, \neg R(x))$.

Множество S будет состоять из семи дизъюнктов: $S = \{\neg P(x) \vee Q(f(x)), \neg P(x) \vee S(x,f(x)), R(a) \vee \neg Q(y), R(a) \vee \neg S(a, y)), P(b), P(x), \neg R(x)\}$. Выводом пустого дизъюнкта

будет последовательность $D_1 = \neg P(x) \vee Q(f(x))$, $D_2 = P(x)$, $D_3 = Q(f(x))$, $D_4 = \neg R(x)$, $D_5 = R(a) \vee \neg Q(y)$, $D_6 = \neg Q(y)$, $D_7 = \square$. Отметим, что D_3 следует по правилу резолюций из D_1, D_2, D_6 – из D_4 и D_5, а D_7 – из D_3 и D_6. Дизъюнкты D_1, D_2, D_4, D_5 принадлежат S.

11. Приведем решение задачи 11.2. Пусть символы одноместных предикатов F, G и S интерпретируются следующим образом

$F(x)$: “x – болельщик ”Спартака”,

$G(x)$: ”x – студент нашей группы”,

$S(x)$: ”x - спортсмен”.

Тогда рассуждение 11.2 можно представить формулами:

$F = (\forall x)(G(x) \to F(x)) \;\&\; (\exists y)(G(y) \;\&\; S(y))$,

$G = (\exists x)(F(x) \;\&\; S(x))$.

Составим множество формул $\{F, \neg G\}$ и каждую из них приведем к сколемовской нормальной форме. Получим формулы

$H_1 = (\forall x)(\neg G(x) \vee F(x)) \;\&\; G(a) \;\&\; S(a)$

$H_2 = (\forall x)(\neg F(x) \vee \neg S(x))$.

Множество дизъюнктов S равно $\{\neg G(x) \vee F(x), \neg F(x) \vee \neg S(x), G(a), S(a)\}$. Пустой дизъюнкт из множества S выводится очевидным образом.

12. Рассуждения 12.1, 12.4 логичны, остальные рассуждения нелогичны.

Глава 5. Функции k-значной логики

Формулы логики высказываний определяют функции, заданные на множестве истинностных значений {0, 1}. Такие функции называются *булевыми*. Они изучались фактически с момента оформления символики математической логики. Классический результат здесь – теорема Поста, которая будет рассмотрена в § 6. При решении ряда теоретических и практических задач потребовалось обобщение двузначной логики в сторону увеличения числа значений истинности. В науке возникли многозначные логики. В качестве значений k-значной логики принято брать множество $\boldsymbol{B}_k = \{0, 1, \ldots, k-1\}$. Данная глава посвящена изучению функций, заданных на этом множестве. В первых параграфах главы будет рассмотрен «частный случай» – класс булевых функций. Здесь вместо $\boldsymbol{B}_2$ мы будем писать $\boldsymbol{B}$.

§ 1. Булевы функции. Способы задания

Определение. *Булевой функцией* называется (всюду определенная) функция вида

$$f\colon \boldsymbol{B}^n \to \boldsymbol{B},$$

где n – число аргументов (переменных), $\boldsymbol{B}^n$ – декартова степень множества $\boldsymbol{B}$.

Другими словами, булева функция – это отображение множества n-элементных последовательностей, составленных из 0 и 1, в множество {0, 1}. Допускается случай $n = 0$; такие булевы функции называются *константами*. Заметим, что константами часто называются также функции, принимающие одно и то же значение при любых значениях аргументов.

Будем считать, что зафиксировано бесконечное множество переменных (аргументов). Элементы этого множества будем обозначать последними строчными буквами латиницы, используя в качестве индексов натуральные числа. Условимся, что *если дана булева функция, то зафиксировано*

обозначение ее переменных. В силу этого, функции $x_1 + x_2$ и $x_3 + x_4$ будем считать различными.

Рассмотрим **способы задания булевых функций** и одновременно приведем примеры.

1. *Задание функций таблицей*. Функцию f от переменных $x_1, x_2, \ldots, x_n$ можно задать таблицей, содержащей $n+1$ столбец, следующим образом. В строках первых n столбцов запишем всевозможные значения переменных $x_1, x_2, \ldots, x_n$. В последнем столбце – соответствующие значения функции f. Например, таблица 5.1 задает так называемую *мажоритарную* функцию (или функцию большинства).

Таблица 5.1

x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Надо обратить внимание на порядок, в котором перебираются в таблице значения аргументов. Мы видим, что в таблице 5.1 этот порядок совпадает с двоичным представлением чисел 0, 1, ..., 7 в указанном порядке. В общем случае n переменных порядок перебора их значений будет совпадать с двоичным представлением чисел 0, 1, ..., $n-1$.

Легко видеть, что таблица, задающая функцию от переменных $x_1, x_2, \ldots, x_n$, содержит 2^n строк. Следовательно, последний столбец представляет собой последовательность из 0 и 1 длины 2^n. Так как первые n столбцов, задающих функции от переменных $x_1, x_2, \ldots, x_n$ совпадают, то число булевых функций совпадает с числом последовательностей из 0 и 1 длины 2^n. Это означает, что число функций $p(n)$от переменных $x_1, x_2, \ldots, x_n$ равно 2^m, где $m = 2^n$. Число $p(n)$ быстро растет с ростом n: $p(1) = 4$, $p(2) = 16$, $p(3) = 256$, $p(4) = 65536$ и т. д.

2. *Задание функций термом*. Предположим, что кроме множества переменных зафиксировано *множество имен функций* $\sigma = \{f, g, h, \ldots\}$. Каждому имени функции поставлено неотрицательное целое число – *местность* функции (или *арность*), т. е. число аргументов функции. Множество σ – называется *сигнатурой*. Напомним (см. главу 2), что *терм* сигнатуры σ – это переменная или выражение вида

$f(t_1, t_2, \ldots, t_n)$, где f – имя n-местной функции, а $t_1, t_2, \ldots, t_n$ – термы. Мы будем использовать также введенное в главе 2 понятие интерпретации, правда в более узком смысле. А именно, под *интерпретацией* здесь будем понимать отображение, определенное на сигнатуре, которое имени n-местной функции ставит в соответствие n-местную функцию, заданную на $\boldsymbol{B}$.

Ясно, что если зафиксирована интерпретация, то каждому терму t ставится в соответствие булева функция f, с тем же набором переменных, что и у терма. В этом случае мы будем говорить, что *терм t задает функцию f*.

В дальнейшем часто будет использоваться сигнатура

$$\tau = \{\theta(x), \iota(x), \varepsilon(x), \neg x,$$
$$x\cdot y, x\vee y, x\to y, x\leftrightarrow y, x+y, x|y, x\downarrow y\}.$$

Интерпретация сигнатуры τ задается таблицами 5.2 и 5.3.

Таблица 5.2

x	$\theta(x)$	$\iota(x)$	$\varepsilon(x)$	$\neg x$
0	0	1	0	1
1	0	1	1	0

Таблица 5.3

x	y	$x\cdot y$	$x\vee y$	$x\to y$	$x\leftrightarrow y$	$x+y$	$x\|y$	$x\downarrow y$
0	0	0	0	1	1	0	1	1
0	1	0	1	1	0	1	1	0
1	0	0	1	0	0	1	1	0
1	1	1	1	1	1	0	0	0

Функции $\theta(x)$ и $\iota(x)$ называются *константами*, $\varepsilon(x)$ – *тождественной функцией*. Для функций $\neg x$, $x\vee y$, $x\to y$, $x\leftrightarrow y$ сохраним названия, идущие из логики высказываний: *отрицание*, *дизъюнкция, импликация* и *эквиваленция*. Функции $x\cdot y$ и $x+y$, – естественно, *умножение* и *сложение*; умножение можно было бы назвать *конъюнкцией*. Точку, обозначающую умножения, будем, как правило, опускать. Функция $x|y$ – *штрих Шеффера*, $x\downarrow y$ – *стрелка Пирса* (или *стрелка Лукасевича*).

Функция $f(x_1,x_2,x_3)$, заданная таблицей 1, может быть задана следующим термом: $x_1x_2 \vee x_1x_3 \vee x_2x_3$ сигнатуры τ.

3. *Задание функций на n-мерном кубе*. Этот способ используется для задания функций с числом аргументов не выше четырех. На множестве $\boldsymbol{B}^n$, т. е. n-той декартовой степени множества $\boldsymbol{B}$, введем отношение порядка следующим

образом. Пусть $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$, и $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ – элементы из $\boldsymbol{B}$. Тогда

$$\boldsymbol{a} \le \boldsymbol{b} \Leftrightarrow a_1 \le b_1, a_2 \le b_2, \ldots, a_n \le b_n.$$

Например, для $n = 4$ выполняются неравенства (0, 1, 0, 0) ≤ (0, 1, 0, 1), (1, 0, 1, 0) ≤ (1, 0, 1, 1). Элементы $\boldsymbol{a}$ = (1, 0, 1, 0) и $\boldsymbol{b}$ = (0, 1, 1, 0) множества $\boldsymbol{B}^4$ несравнимы, т. е. не выполняется ни $\boldsymbol{a} \le \boldsymbol{b}$, ни $\boldsymbol{b} \le \boldsymbol{a}$. Диаграмма Хассе частично упорядоченного множества $(\boldsymbol{B}, \le)$ называется *n-мерным булевым кубом*. На рисунках 5.1 и 5.2 изображены 2-мерный и 3-мерный булевы кубы.

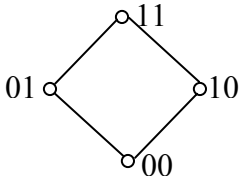


Рис. 5.1

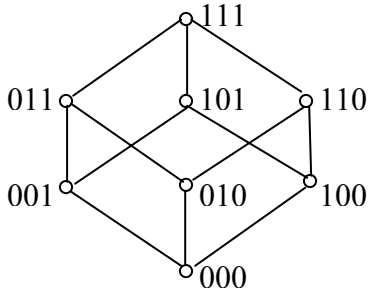


Рис. 5.2

Булева функция f на n-мерном кубе задается следующим образом. Если в некоторой точке a из $\boldsymbol{B}^n$ $f(a) = 1$, то соответствующая вершина-кружочек n-мерного куба заполняется черным цветом. Если же $f(a) = 0$, то вершина-кружочек остается полой.

Например, функция $f(x_1,x_2,x_3) = (x_1 \to x_2) + x_3$ на 3-мерном кубе будет задана так, как показано на рис. 5.3.

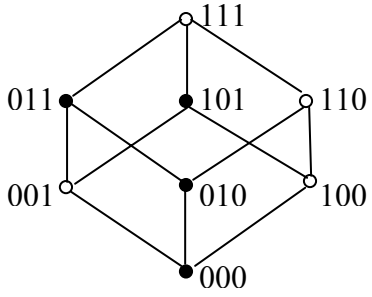


Рис. 5.3

4. *Задание функций картой Карно*. Способ используется для задания функций от трех и четырех аргументов. Карта Карно – это своеобразная форма таблицы. На рисунке 5.4 приведена «заготовка» карты Карно, для функции от переменных x_1, x_2, x_3, x_4.

x_1x_2 \ x_3x_4	00	01	11	10
00				
01				
11				
10				

Рис. 5.4

Условимся, что «шапка» таблицы, приведенной на рис 4, – нулевая строка и что нумерация столбцов также начинается с нуля. Строки карты задают значения переменных x_1 и x_2. Например, во второй строке, $x_1 = 0$, $x_2 = 1$. Столбцы задают значения переменных x_3 и x_4. Например, третий столбец задает значения $x_3 = 1$, $x_4 = 1$. Надо обратить внимание на то, что соседние наборы значений переменных в строках различаются значением ровно одной переменной. Это справедливо и для первой и последней строк. Аналогичное утверждение выполняется и для столбцов. Таблицу, ограниченную пунктирным прямоугольником, часто надо представлять в виде тора. Для этого представим себе, что «склеили» сначала верхний и нижний края этой таблицы, получили цилиндр. Затем «склеили» левую и правую окружности.

Клетка таблицы, ограниченной пунктирным прямоугольником, определяется значениями всех четырех переменных. Так, например, если взять $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$, то получим клетку, расположенную в третьей строке и втором столбце. Заполнять карту Карно для функции f надо следующим образом. Если на наборе переменных $\boldsymbol{a}$ выполняется равенство $f(\boldsymbol{a}) = 1$, то в клетке, соответствующей набору $\boldsymbol{a}$, записываем 1. Если же $f(\boldsymbol{a}) = 0$, то клетка остается пустой. На рис. 5.5 приведена карта Карно, задающая функцию $(x_1x_2) \to (x_3+x_4)$.

x_1x_2 \ x_3x_4	00	01	11	10
00	1		1	
01	1	1	1	1
11		1		1
10	1	1	1	1

Рис. 5.5

Каким образом трехместная функция задается картой Карно, в общем случае излагать не будем. Ограничимся лишь примером. На рис. 5.6 картой Карно задана функция $f(x_1,x_2,x_3) = (x_1 \to x_2) + x_3$.

x_1 \ x_2x_3	00	01	11	10
0	1			1
1		1		1

Рис. 5.6

5. *Задание функций вектором значений*. Выше мы условились, что в случае n переменных порядок перебора их значений будет совпадать с двоичным представлением чисел 0, 1, ..., $n-1$. Можно, следовательно, говорить о нулевом наборе значений переменных, первом, втором и т. д. Например, в случае трех переменных третьим набором будет 011. Поскольку множество значений аргументов упорядочено, функцию можно задать вектором значений. Например, вектор значений f = (0110 1101), задает функцию трех аргументов (так как значений 8, а это 2^3), для которой $f(0, 0, 0) = 0$, $f(0, 0, 1) = 1$, $f(0, 1, 0)= 1$ и т. д.

Закончим параграф введением понятия равенства булевых функций. Предварительно сформулируем следующее

Определение. Переменная x_i $(1 \le i \le n)$ функции $f(x_1, x_2, \ldots, x_n)$ называется *существенной*, если существуют наборы $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$ значений переменных $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$, такие что

$$f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \ne f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n).$$

В противном случае переменная x_i называется *фиктивной*.

Например, у функции $f(x_1, x_2, x_3) = x_1 \to (x_2x_3 + x_2\neg x_3)$ переменные x_1 и x_2 существенны, а переменная x_3 фиктивна. В этом можно убедиться, воспользовавшись дистрибутивностью умножения относительно сложения и тем, что функция $x_3 + \neg x_3$ тождественно равна 1.

Функцию $x_1 + x_2$ можно обозначить через $g(x_1, x_2, x_3)$. Переменная x_3 является здесь фиктивной. Мы будем говорить, что функция $g(x_1, x_2, x_3)$ получена из $x_1 + x_2$ *введением фиктивной переменной*. Введение фиктивных переменных мы будем часто использовать, если рассматривается некоторое конечное множество функций. Тогда можно считать,

что все функции *зависят от одного и того же набора переменных*.

Определение. Функции f и g называются *равными*, если существенные переменные этих функций одни и те же, и значения функций f и g совпадают на любом наборе значений существенных переменных.

Например, функции $f(x_1, x_2) = x_1 \vee x_2$ и $g(x_1, x_2, x_3, x_4) = x_1(\theta(x_3) \to x_4) \vee (x_2 + (x_3 \cdot \neg x_3))$ равны, так как функция $\theta(x_3) \to x_4$ тождественно равна 1, а функция $x_3 \cdot \neg x_3$ тождественно равна 0.

§2. Булевы функции. Замкнутость и полнота

В этом параграфе будут рассмотрены два важных понятия, вынесенные в название параграфа.

Определение. Класс булевых функций K называется *замкнутым относительно переименования аргументов,* если из того, что n-местная функция $f(x_1, \ldots, x_n)$ принадлежит K, а $y_1, \ldots y_n$ – другие переменные (среди которых могут быть и совпавшие), то функция $f(y_1, \ldots, y_n)$ также принадлежит K.

Например, если K содержит функцию $x+y$ и замкнут относительно переименования аргументов, то K содержит также и функцию $\theta(x)$.

Определение. Класс булевых функций K называется *замкнутым относительно суперпозиции*, если из того, что функции $f(x_1, \ldots, x_k)$, $g_1(x_1, \ldots, x_n)$, $\ldots$, $g_k(x_1, \ldots, x_n)$ принадлежат K следует, что функция

$$f(g_1(x_1, \ldots, x_n), \ldots g_k(x_1, \ldots, x_n))$$

также принадлежит K.

Например, если K содержит функцию $x \vee y$ и $\neg x$ и замкнут относительно суперпозиции (и переименования аргументов), то K содержит и $x \cdot y$, поскольку $x \cdot y = \neg(\neg x \vee \neg y)$.

Определение. Класс булевых функций K называется *замкнутым*, если K

1) содержит тождественную функцию;
2) замкнут относительно переименования аргументов;
3) замкнут относительно суперпозиции.

Приведем примеры. Функция $f(x_1, \ldots, x_n)$ называется *сохраняющей ноль*, если $f(0, \ldots, 0) = 0$. Функции $x \cdot y$, $x \vee y$, $x+y$

сохраняют ноль, а $x \leftrightarrow y$, $x|y$, $x \downarrow y$ не сохраняют. Через T_0 обозначим класс всех функций, сохраняющих 0, т.е.

$$T_0 = \{f(x_1, \ldots, x_n) | \, f(0, \ldots, 0) = 0\}.$$

Ясно, что T_0 – замкнутый класс.

Функция $f(x_1, \ldots, x_n)$ называется *сохраняющей единицу,* если $f(1, \ldots, 1) = 1$. Функции $x \cdot y$, $x \vee y$, $x \to y$ сохраняют единицу, а функции $x+y$, $x|y$, $x \downarrow y$ не сохраняют. Через T_1 обозначим класс всех функций, сохраняющих 1, т.е.

$$T_1 = \{f(x_1, \ldots, x_n) | \, f(1, \ldots, 1) = 1\}.$$

Класс T_1 также, как и T_0, является замкнутым.

Теорема 5.1. Пересечение непустого семейства $\{K_i \mid i \in I\}$ замкнутых классов является замкнутым классом.

Доказательство. Пусть $K = \cap \{K_i \mid i \in I\}$. Класс K содержит тождественную функцию, так как все K_i ее содержат. Если $f(x_1, \ldots, x_n)$ принадлежит K, а $y_1, \ldots, y_n$ – новый набор переменных, то для любого i функция $f(x_1, \ldots, x_n)$ принадлежит K_i, поскольку K – пересечение этих классов, и $f(y_1, \ldots, y_n) \in K_i$, поскольку K_i замкнут относительно переименования аргументов. Если функция $f(y_1, \ldots, y_n)$ принадлежит всем классам K_i, то эта функция принадлежит и их пересечению, т.е. K. Следовательно, K замкнут относительно переименования аргументов.

Предположим, что функции $f(x_1, \ldots, x_k)$, $g_1(x_1, \ldots, x_n)$, $\ldots$, $g_k(x_1, \ldots, x_n)$ принадлежат K. Тогда эти функции принадлежат всем классам K_i. Класс K_i замкнут относительно суперпозиции. Следовательно, для любого i класс K_i содержит функцию

$$f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)).$$

Отсюда следует, что эта функция принадлежит пересечению этих классов – классу K. Мы доказали, что K замкнут относительно суперпозиции. □

Определение. *Замыканием* класса K называется пересечение всех замкнутых классов, содержащих K.

Отметим, что для любого класса K существует замкнутый класс, содержащий K, – это класс всех булевых функций.

Замыкание класса K будем обозначать через $[K]$. Из теоремы 1 следует, что $[K]$ – замкнутый класс. Отметим ряд простых свойств замыкания.

Теорема 5.2. Пусть K – класс булевых функций. Тогда

1) $K \subseteq [K]$,

2) если $K \subseteq L$, то $[K] \subseteq [L]$,

3) класс K замкнут тогда и только тогда, когда $K = [K]$.

Доказательство. Пусть $P = \{K_i \mid i \in I\}$ – семейство (всех) замкнутых классов, содержащих K. Так как K содержится в K_i для любого i, то K содержится и в их пересечении, т.е. в $[K]$.

Предположим, что $Q = \{L_j \mid j \in J\}$ – семейство замкнутых классов, содержащих L, и что $K \subseteq L$.Тогда $K \subseteq L_j$ для любого j и поэтому $Q \subseteq P$. Отсюда следует, что пересечение классов, принадлежащих P содержится в пересечении классов, принадлежащих Q, т.е. что $[K] \subseteq [L]$.

Если K – замкнутый класс, то $K \in P$. Следовательно, $K = \cap P = [K]$, так как все классы из P содержат K. Если же $K = [K]$, то, как отмечалось выше, $[K]$ – замкнутый класс. □

Убедимся в том, что замыкание класса K можно определить, как класс L всех функций, которые получаются из K и тождественной функции с помощью суперпозиции и переименования аргументов. Действительно, пусть P – семейство замкнутых классов, содержащих K. Легко видеть, что класс L замкнут. Следовательно, $L \in P$, поскольку $K \subseteq L$. Пусть K' – класс из P. Тогда K' содержит K, тождественную функцию и замкнут относительно суперпозиции и переименования аргументов. Это означает, что $L \subseteq K'$. Мы убедились в том, что $L \in P$ и что все классы из P содержат L. Следовательно, пересечение всех классов из P, т. е. замыкание класса K, равно L.

Пусть $K = \{\theta(x), \iota(x), x+y, x \cdot y\}$. Тогда класс $[K]$ содержит многочлены (от любого числа переменных). Эти многочлены называются *полиномами Жегалкина*. Поскольку произведение элементов множества B удовлетворяет тождеству $u^2 = u$, то полином Жегалкина можно записать в виде $\Sigma a_{i1,\ldots,ik} x_{i1 \ldots} x_{ik,}$ где $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, $a_{i1,\ldots,ik} \in B$ и суммирование ведется по всем подмножествам множества $\{1, \ldots, n\}$. Например, $1 \cdot x_1 x_2 + 0 \cdot x_1 + 1 \cdot x_2 + 1$ – полином Жегалкина от двух переменных x_1 и x_2, $a_{12} = 1$, $a_1 = 0$, $a_2 = 1$, $a_0 = 1$. Этот полином можно, конечно, записать и так: $x_1 x_2 + x_2 + 1$.

Обозначим через $\boldsymbol{P}_2$ класс всех булевых функций.

Определение. Класс функций K называется *полным*, если $[K] = \boldsymbol{P}_2$.

Другими словами, класс K полон, если любую булеву функцию можно получить из K (и тождественной функции) с помощью суперпозиции и переименования аргументов.

Теорема 5.3. Следующие классы функций являются полными

$1) K_1 = \{x \cdot y,\ x \vee y,\ \neg x\}$,
$2) K_2 = \{x \cdot y,\ \neg x\ \}$,
$3) K_3 = \{x \vee y,\ \neg x\}$,
$4) K_4 = \{x \cdot y,\ x+y,\ \theta(x),\ \iota(x)\}$.

Доказательство. Докажем, что любая функция может быть получена из функций класса K_1 с помощью суперпозиции и переименования аргументов. Доказательство проведем на примере функции $f(x_1, x_2, x_3)$, заданной таблицей 5.4.

Таблица 5.4

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Выделим те строки таблицы, где в последнем столбце стоит 1. Это вторая, шестая, и седьмая строки. Каждой строке поставим в соответствие произведение переменных или их отрицаний следующим образом. Второй строке поставим в соответствие произведение $\neg x_1 \cdot \neg x_2 \cdot x_3$, шестой – $x_1 \cdot \neg x_2 \cdot x_3$, седьмой – $x_1 \cdot x_2 \cdot \neg x_3$, т.е. если переменная x принимает значение 1, то пишем x, если принимает значение 0, то пишем $\neg x$. Легко видеть, что функция

$g(x_1, x_2, x_3) = [\neg x_1 \cdot \neg x_2 \cdot x_3] \vee [x_1 \cdot \neg x_2 \cdot x_3] \vee [\ x_1 \cdot x_2 \cdot \neg x_3]$

задается той же таблицей 4. Это означает, что функция $f(x_1, x_2, x_3)$ равна $g(x_1, x_2, x_3)$. Но в записи функции $g(x_1, x_2, x_3)$ использованы только функции из класса K_1. Следовательно, $f(x_1, x_2, x_3)$ может быть получена из функций класса K_1 суперпозицией и переименованием аргументов, т.е. $f(x_1, x_2, x_3) \in [K]$. (Отметим, что есть прямая аналогия между построением функции $g(x_1, x_2, x_3)$ и приведением к СДНФ, из-

ложенном в конце §5 главы 1.) На основании этого заключаем, что любая булева функция принадлежит замыканию $[K_1]$. Следовательно, K_1 – полный класс.

Полнота классов K_2 и K_3 получается из полноты класса K_1 с помощью следующего утверждения. Если K – полный класс и любая функция класса K выражается (с помощью суперпозиции и переименования аргументов) через функции класса L, то L – полный класс. Действительно, если любая булева функция выражается через функции класса K, то она будет выражаться и через функции класса L. Мы уже отмечали, что $x \cdot y = \neg(\neg x \vee \neg y)$. Это равенство доказывает полноту класса K_3. Далее справедливо равенство $x \vee y = \neg(\neg x \cdot \neg y)$, из которого следует полнота класса K_2. Полнота класса K_4 следует из полноты класса K_2 и равенства $\neg x = x + \iota(x)$. Полнота класса K_4 означает, в частности, что каждая булева функция может быть задана полиномом Жегалкина. Например, $x_1 \to x_2 = x_1 \cdot x_2 + x_1 + 1$, $x_1 \vee x_2 = x_1 \cdot x_2 + x_1 + x_2$. □

Связь излагаемого материала с логикой высказываний достаточно очевидна. Тем не менее, отметим следующее. Пусть $F(X_1, \ldots, X_n)$ и $G(X_1, \ldots, X_n)$ – формулы логики высказываний, в которых нет импликации и эквиваленции. Заменим $X\&Y$ на $X \cdot Y$, большие буквы X_i на маленькие x_i $(1 \le i \le n)$. Получим функции $f(x_1, \ldots, x_n)$ и $g(x_1, \ldots, x_n)$. Тогда формулы $F(X_1, \ldots, X_n)$ и $G(X_1, \ldots, X_n)$ равносильны в том и только в том случае, когда функции $f(x_1, \ldots, x_n)$ и $g(x_1, \ldots, x_n)$ равны. Это замечание позволит нам при доказательстве равенства функций ссылаться на законы логики высказываний.

§3. Самодвойственные функции

Этот и два следующих параграфа посвящены рассмотрению трех конкретных классов функций: самодвойственных, монотонных и линейных.

Определение. Функция $g(x_1, \ldots, x_n)$ называется *двойственной к* $f(x_1, \ldots, x_n)$, если выполняется равенство

$$g(x_1, \ldots, x_n) = \neg[f(\neg x_1, \ldots, \neg x_n)].$$

Например, $x \cdot y$ двойственна $x \vee y$, и наоборот. Это следует из равенств $x \cdot y = \neg(\neg x \vee \neg y)$ и $x \vee y = \neg(\neg x \cdot \neg y)$, которые мы уже приводили. Функция $x \to y$ двойственна функции $\neg x \cdot y$, поскольку $\neg(\neg x \to \neg y) = \neg[\neg(\neg x) \vee \neg y] = \neg(x \vee \neg y) = \neg x \cdot y$. Отметим, что первое равенство выполняется на осно-

вании закона 20 логики высказываний, второе – на основании закона 19 и третье – на основании законов 18 и 19.

Определение. Функция $f(x_1, \ldots, x_n)$ называется *самодвойственной*, если выполняется равенство

$$f(x_1, \ldots, x_n) = \neg[f(\neg x_1, \ldots, \neg x_n)] \qquad (1)$$

Другими словами, функция самодвойственна, если она совпадает со своей двойственной.

Приведем примеры. Легко видеть, что самодвойственными функциями являются тождественная функция и отрицание: $\neg[\varepsilon(\neg x)] = \neg[\neg x] = x = \varepsilon(x)$, $\neg[\neg(\neg x)] = \neg x$. В то же время, произведение $x \cdot y$ самодвойственной не является, поскольку двойственно дизъюнкции $x \vee y$ и $x \cdot y \neq x \vee y$. Можно показать, что никакая функция от двух переменных, существенно зависящая от каждого аргумента, самодвойственной не является. В качестве еще одного примера самодвойственной функции приведем функцию

$$f(x_1, x_2, x_3) = (x_1 \cdot x_2) \vee (x_1 \cdot x_3) \vee (x_1 \cdot x_3).$$

Действительно, $\neg[f(\neg x_1, \neg x_2, \neg x_3] =$

$$= \neg[(\neg x_1 \cdot \neg x_2) \vee (\neg x_1 \cdot \neg x_3) \vee (\neg x_2 \cdot \neg x_3)] =$$
$$= \neg[\neg(x_1 \vee x_2) \vee \neg(x_1 \vee x_3)) \vee \neg(x_2 \vee x_3)] =$$
$$= (x_1 \vee x_2) \cdot (x_1 \vee x_3) \cdot (x_2 \vee x_3) =$$
$$= (x_1 \cdot x_1 \cdot x_2) \vee (x_1 \cdot x_1 \cdot x_3) \vee (x_1 \cdot x_3 \cdot x_2) \vee (x_1 \cdot x_3 \cdot x_3) \vee$$
$$\vee (x_2 \cdot x_1 \cdot x_2) \vee (x_2 \cdot x_1 \cdot x_3) \vee (x_2 \cdot x_3 \cdot x_2) \vee (x_2 \cdot x_3 \cdot x_3) =$$
$$= (x_1 \cdot x_2) \vee (x_1 \cdot x_3) \vee (x_1 \cdot x_2 \cdot x_3) \vee (x_1 \cdot x_3) \vee$$
$$\vee (x_1 \cdot x_2) \vee (x_1 \cdot x_2 \cdot x_3) \vee (x_2 \cdot x_3) \vee (x_2 \cdot x_3) =$$
$$= (x_1 \cdot x_2) \vee (x_1 \cdot x_3) \vee (x_2 \cdot x_3) \vee (x_1 \cdot x_2 \cdot x_3) =$$
$$= (x_1 \cdot x_2) \vee (x_1 \cdot x_3) \vee (x_2 \cdot x_3).$$

Последнее равенство выполняется на основании закона поглощения.

Отметим, что равенство (1) из определения самодвойственности равносильно равенству

$$\neg(f(x_1, \ldots, x_n)) = f(\neg x_1, \ldots, \neg x_n). \qquad (2)$$

Класс всех самодвойственных функций обозначим буквой S. Убедимся в том, что S – замкнутый класс. Как уже отмечалось, S содержит $\varepsilon(x)$. Пусть $f(x_1, \ldots, x_n) \in S$ и $y_1, \ldots, y_n$ – новый набор переменных. Тогда поскольку равенство $f(x_1, \ldots, x_n) = \neg[f(\neg x_1, \ldots, \neg x_n)]$ выполняется для всех значений переменных $x_1, \ldots, x_n$, то $\neg[f(\neg y_1, \ldots, \neg y)] = f(y_1, \ldots, y_n)$. Следовательно, $f(y_1, \ldots, y_n) \in S$ и S – замкнут относительно переименования аргументов. Возьмем теперь функ-

ции $f(x_1, \ldots, x_k)$, $g_1(x_1, \ldots, x_n)$, $\ldots$, $g_k(x_1, \ldots, x_n)$ из S. Поскольку эти функции принадлежат S, используя равенство (2), получаем, что

$$\neg f(x_1, \ldots, x_k) = f(\neg x_1, \ldots, \neg x_n),$$
$$\neg(g_1(x_1, \ldots, x_n)) = g_1(\neg x_1, \ldots, \neg x_n),$$
$$\cdot \quad \cdot \quad \cdot$$
$$\neg(g_k(x_1, \ldots, x_n)) = g_k(\neg x_1, \ldots, \neg x_n).$$

Тогда если $h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))$, то

$$h(\neg x_1, \ldots, \neg x_n) =$$
$$= f[g_1(\neg x_1, \ldots, \neg x_n), \ldots, g(\neg x_1, \ldots, \neg x_n)] =$$
$$= f[\neg(g_1(x_1, \ldots, x_n)), \ldots, \neg(g_k(x_1, \ldots, x_n))] =$$
$$= \neg[f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))] =$$
$$= \neg(h(x_1, \ldots, x_n)).$$

В силу равенства (2), $h(x_1, \ldots, x_n)$ – самодвойственная функция. Следовательно, класс S замкнут относительно суперпозиции.

Следующее утверждение называется *леммой о несамодвойственной функции.*

Лемма. Пусть $f(x_1, \ldots, x_n)$ – несамодвойственная функция. Тогда замыкание класса $K = \{f(x_1, \ldots, x_n), \neg x\}$ содержит константы $\theta(x)$ и $\iota(x)$.

Доказательство. Поскольку $f(x_1, \ldots, x_n)$ – несамодвойственная функция, то существуют $a_1, \ldots, a_n \in B$ такие, что

$$\neg[f(a_1, \ldots, a_n)] \neq f(\neg a_1, \ldots, \neg a_n).$$

Множество B содержит только два элемента. Поэтому из этого неравенства следует равенство

$$f(a_1, \ldots, a_n) = f(\neg a_1, \ldots, \neg a_n).$$

Для удобства обозначений предположим, что $a_1, \ldots, a_k = 0$, $a_{k+1}, \ldots, a_n = 1$. Тогда последнее равенство можно записать так:

$$f(0, \ldots, 0; 1, \ldots, 1) = f(1, \ldots, 1; 0, \ldots, 0),$$

где точка с запятой отделяет k-й аргумент от $(k+1)$-го.

Рассмотрим функцию

$$g(x) = f(x, \ldots, x; \neg x, \ldots, \neg x).$$

Заметим, что $g(x)$ принадлежит $[K]$. Выполняются равенства

$$g(0) = f(0, \ldots, 0; \neg 0, \ldots, \neg 0) = f(0, \ldots, 0; 1, \ldots, 1) =$$
$$= f(1, \ldots, 1; 0, \ldots, 0) = f(1, \ldots, 1; \neg 1, \ldots, \neg 1) = g(1).$$

Следовательно, $g(x)$ – одна из констант, принадлежащая $[K]$. Поскольку K содержит отрицание, то и другая константа принадлежит $[K]$. □

В заключение параграфа рассмотрим пример решения задачи на распознавание самодвойственности: определить, будет ли функция $f(x_1, x_2) = x_2 \cdot (x_2 \to x_1)$ самодвойственной? (Впрочем, мы уже знаем, что $f(x_1, x_2)$ несамодвойственна, но надо это доказать). Для получения ответа на вопрос составим таблицу, задающую функции$f(x_1, x_2)$ и $\neg(f(\neg x_1, \neg x_2)$. (см. таблицу 5.5)

Таблица 5.5

x_1	x_2	$f(x_1,x_2)$	$\neg f(\neg x_1, \neg x_2)$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Мы видим, что $f(x_1, x_2) \neq \neg f(\neg x_1, \neg x_2)$, например, значения этих функций различаются при $x_1 = 0$, $x_2 = 1$. Следовательно, функция $f(x_1, x_2)$ самодвойственной не является. (Для того, чтобы сделать заключение о несамодвойственности, можно было, конечно, прервать составление таблицы на второй строке.)

§4. Монотонные функции

Напомним, что в первом параграфе на множестве $\boldsymbol{B}^n$ был введен порядок прямого произведения:

$$(a_1, \ldots, a_n) \le (b_1, \ldots, b_n) \Leftrightarrow a_1 \le b_1, \ldots, a_n \le b_n.$$

Определение. Функция $f(x_1, \ldots, x_n)$ называется *монотонной*, если для любых двух векторов $\boldsymbol{a} = (a_1, \ldots, a_n)$ и $\boldsymbol{b} = (b_1, \ldots, b_n)$, принадлежащих $\boldsymbol{B}^n$, из неравенства $\boldsymbol{a} \le \boldsymbol{b}$ следует неравенство $f(\boldsymbol{a}) \le f(\boldsymbol{b})$.

Примерами монотонных функций являются $\theta(x)$, $\iota(x)$, $x \cdot y$, $x \vee y$. Функции $\neg x$, $x \to y$, $x + y$ немонотонны.

Рассмотрим задачу: выяснить будет ли функция

$$f(x_1, x_2, x_3) = x_1 + x_2 \cdot x_3$$

монотонной? Для решения этой задачи функцию f зададим на 3-мерном кубе $\boldsymbol{B}^3$ (см. рис. 5.7).Монотонность функции означает, что если в какой-то вершине диаграммы она принимает значение 1, то и всюду выше функция принимает то же значение 1. Функция $f(x_1, x_2, x_3) = x_1 + x_2 \cdot x_3$ монотонной не

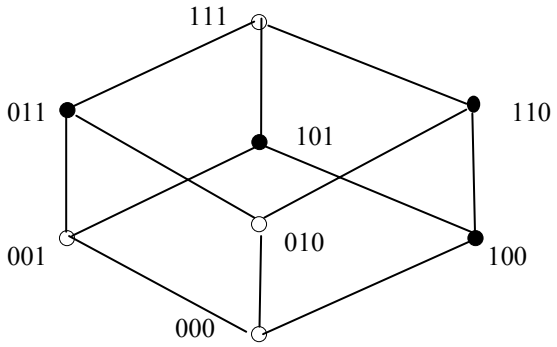


Рис. 5.7

является, так как в вершине 011 она принимает значение 1, а в вершине 111, которая выше первой, принимает значение 0.

Класс всех монотонных функций обозначим буквой M. Убедимся в том, что M – это замкнутый класс. Монотонность тождественной функции очевидна. Пусть $f(x_1, \ldots, x_n) \in M$ и $y_1, \ldots, y_n$ – новые переменные. Возьмем два набора значений переменных $y_1, \ldots, y_n$: $\boldsymbol{a} = (a_1, \ldots, a_n)$, $\boldsymbol{b} = (b_1, \ldots, b_n)$ таких, что $\boldsymbol{a} \le \boldsymbol{b}$. Но эти векторы будут и наборами значений переменных $x_1, \ldots, x_n$, и поэтому выполняется неравенство $f(\boldsymbol{a}) \le f(\boldsymbol{b})$. Это означает, что класс M замкнут относительно переименования аргументов.

Пусть $f(x_1, \ldots, x_k)$, $g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)$ – монотонные функции и $\boldsymbol{a} = (a_1, \ldots, a_n)$, $\boldsymbol{b} = (b_1, \ldots, b_n)$ – два набора значений переменных $x_1, \ldots, x_n$ таких, что $\boldsymbol{a} \le \boldsymbol{b}$. Рассмотрим суперпозицию

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)).$$

Значения функций $g_1(a_1, \ldots, a_n), \ldots, g_k(a_1, \ldots, a_n)$ обозначим соответственно через $c_1, \ldots, c_k$, а значения $g_1(b_1, \ldots, b_n)$, $\ldots, g_k(b_1, \ldots, b_n)$ – через $d_1, \ldots, d_k$. В силу монотонности функций $g_1, \ldots, g_k$ имеем $c_1 \le d_1, \ldots, c_k \le d_k$. Тогда $h(a_1, \ldots, a_n) = f(g_1(a_1, \ldots, a_n), \ldots, g_k(a_1, \ldots, a_n)) = f(c_1, \ldots, c_k) \le f(d_1, \ldots, d_k) = f(g_1(b_1, \ldots, b_n), \ldots, g_k(b_1, \ldots, b_n)) = h(b_1, \ldots, b_n)$. (Знак неравенства поставлен в силу монотонности функции f.) Мы доказали, что класс M замкнут относительно суперпозиции.

Следующее утверждение называется *леммой о немонотонной функции.*

Лемма. Пусть $f(x_1, \ldots, x_n)$ – немонотонная функция. Тогда замыкание класса $K = \{f(x_1, \ldots, x_n), \theta(x), \iota(x)\}$ содержит $\neg x$.

Доказательство. Поскольку $f(x_1, \ldots, x_n)$ немонотонна, существуют наборы значений переменных $\boldsymbol{a} = (a_1, \ldots, a_n)$, $\boldsymbol{b} = (b_1, \ldots, b_n)$ такие, что $\boldsymbol{a} < \boldsymbol{b}$, но $f(\boldsymbol{a}) > f(\boldsymbol{b})$. Неравенство $f(\boldsymbol{a}) > f(\boldsymbol{b})$ означает, что $f(\boldsymbol{a}) = 1$ и $f(\boldsymbol{b}) = 0$. Рассмотрим диаграмму частично упорядоченного множества $\boldsymbol{B}^n$. Так как $\boldsymbol{a} < \boldsymbol{b}$, точка $\boldsymbol{a}$ на диаграмме лежит ниже точки $\boldsymbol{b}$. Существует

восходящая цепь, соединяющая точку ***a*** с точкой ***b*** (см.рис. 5.8).

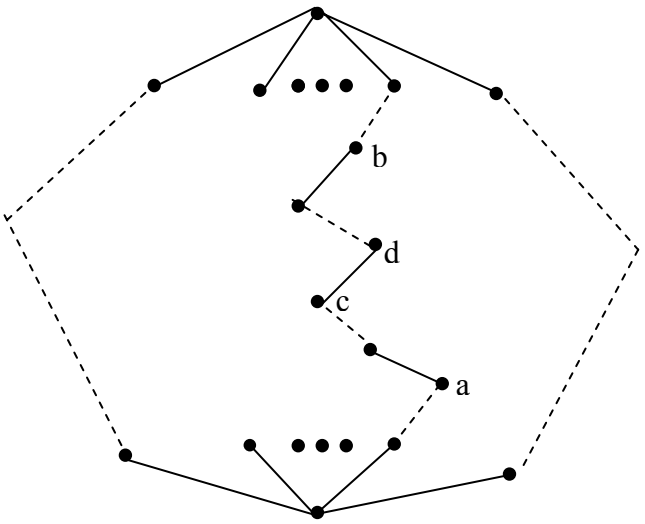


Рис. 5.8

На этой цепи найдутся соседние точки $\boldsymbol{c} = (c_1, \ldots, c_n)$ и $\boldsymbol{d} = (d_1, \ldots, d_n)$ такие, что $\boldsymbol{c} < \boldsymbol{d}$, $f(\boldsymbol{c}) = 1$ и $f(\boldsymbol{d}) = 0$. Поскольку $\boldsymbol{c}$ и $\boldsymbol{d}$ – соседние точки, то для некоторого i выполняются равенства

$$\boldsymbol{c} = (c_1, \ldots, c_{i-1}, 0, c_{i+1}, \ldots, c_n) \text{ и}$$
$$\boldsymbol{d} = (c_1, \ldots, c_{i-1}, 1, c_{i+1}, \ldots, c_n).$$

Для удобства обозначений будем считать, что $c_1 = \ldots = c_{i-1} = 0$, $c_{i+1} = \ldots c_n = 1$. Рассмотрим функцию

$$g(x) = f(\theta(x), \ldots, \theta(x), x; \iota(x), \ldots, \iota(x)),$$

где точка с запятой отделяет i-ю компоненту от $(i+1)$-ой. Ясно, что $g(x)$ принадлежит замыканию класса K. Далее,

$$g(0) = f(\theta(0), \ldots, \theta(0), 0; \iota(0), \ldots, \iota(0) =$$
$$= f(c_1, \ldots, c_{i-1}, 0; c_{i+1}, \ldots, c_n) = f(c) = 1,$$
$$g(1) = f(\theta(1), \ldots, \theta(1), 1; \iota(1), \ldots, \iota(1)) =$$
$$= f(c_1, \ldots, c_{i-1}, 1; c_{i+1}, \ldots, c_n) = f(d) = 0.$$

Это означает, что $g(x) = \neg x$. □

§5. Линейные функции

Определение. Функция $f(x_1, \ldots, x_n)$ называется *линейной*, если существуют $a_0, a_1, \ldots, a_n \in \boldsymbol{B}$ такие, что

$$f(x_1, \ldots, x_n) = a_0 + a_1 \cdot x_1 + \ldots + a_n \cdot x_n.$$

Примерами линейных функций являются $\theta(x)$, $\iota(x)$, $\neg x$, $x+y$. Функции $x \cdot y$, $x \vee y$, $x \to y$ линейными не являются. Докажем, например, нелинейность импликации $f(x, y) = x \to y$. Предположим противное. Пусть $f(x, y)$ – линейная функция.

Это означает, что $x \to y = a_0 + a_1 \cdot x + a_2 \cdot y$ для некоторых a_0, a_1, $a_2 \in \boldsymbol{B}$. Составим таблицу, задающую функции, стоящие в левой и правой части последнего равенства (см. таблицу 5.6).

Таблица 5.6

x	y	$x \to y$	$a_0 + a_1 \cdot x + a_2 \cdot y$	Следствие
0	0	1	a_0	$a_0 = 1$
0	1	1	$1 + a_2$	$a_2 = 0$
1	0	0	$1 + a_1$	$a_1 = 1$
1	1	1	0	Противоречие

Противоречие показывает, что $x \to y$ – нелинейная функция.

Класс всех линейных функций обозначим буквой L. Класс L является замкнутым. Действительно, тождественная функция является линейной. Далее очевидно, что замена аргументов у линейной функции дает линейную и суперпозиция линейных функций линейна.

Следующее утверждение называется *леммой о нелинейной функции*.

Лемма. Пусть $f(x_1, \ldots, x_n)$ – нелинейная функция. Тогда замыкание класса $K = \{f(x_1, \ldots, x_n), \theta(x), \iota(x), \neg x\}$ содержит произведение $x \cdot y$.

Доказательство. Как отмечалось в §2, каждая функция может быть задана полиномом Жегалкина. Это означает, что

$$f(x_1, \ldots, x_n) = \sum a_{i1, \ldots, ik}\, x_{i1} \cdot x_{i2} \cdot \ldots \cdot x_{ik}, \qquad (1)$$

где $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, $a_{i1}, \ldots, a_{ik} \in \boldsymbol{B}$ и суммирование ведется по всем подмножествам множества $\{1, \ldots, n\}$. Так как $f(x_1, \ldots, x_n)$ – нелинейная функция, то хотя бы одно слагаемое имеет степень, большую или равную двум. Без ограничения общности можно считать, что это слагаемое содержит произведение $x_1 \cdot x_2$. Слагаемые правой части равенства (1) разделим на 4 группы: содержащие $x_1 \cdot x_2$; содержащие x_1 и не содержащие x_2; не содержащие x_1 и содержащие x_2, не содержащие x_1 и x_2. В первой группе вынесем за скобки $x_1 \cdot x_2$, во второй – x_1, в третьей – x_2, получим, что

$$f(x_1, \ldots, x_n) = x_1 \cdot x_2 \cdot f_0(x_3, \ldots, x_n) + x_1 \cdot f_1(x_3, \ldots, x_n) + x_2 \cdot f_2(x_3, \ldots, x_n) + f_3(x_3, \ldots, x_n).$$

Если $f_0(x_3, \ldots, x_n)$ есть тождественный нуль, то выделим произведение переменных в одном из оставшихся слагае-

мых и проведем аналогичную группировку. Будем поэтому считать, что существуют элементы $a_3, \ldots, a_n$ из $\boldsymbol{B}$ такие, что $f_0(a_3, \ldots, a_n) = 1$. Пусть $c_1 = f_1(a_3, \ldots, a_n)$, $c_2 = f_2(a_3, \ldots, a_n)$, $c_3 = f_3(a_3, \ldots, a_n)$. Рассмотрим функции

$$g(x_1, x_2) = f(x_1, x_2, a_3, \ldots, a_n) = x_1 \cdot x_2 + c_1 \cdot x_1 + c_2 \cdot x_2 + c_3 \text{ и}$$

$$h(x_1, x_2) = g(x_1 + c_2, x_2 + c_1) + c_1 \cdot c_2 + c_3$$

Если $c = 1$, то $c \cdot x = x$, $x + c = \neg x$; если же с = 0, то $c \cdot x = \theta(x)$, $x + c = x$. Отсюда следует, что функции $g(x_1, x_2)$ и $h(x_1, x_2)$ принадлежат замыканию класса K. В то же время

$$h(x_1, x_2) = (x_1 + c_2)(x_2 + c_1) +$$
$$c_1(x_1 + c_2) + c_2(x_2 + c_1) + c_3 + c_1 \cdot c_2 + c_3 =$$
$$x_1 \cdot x_2 + c_2 \cdot x_2 + c_1 \cdot x_1 + c_1 \cdot c_2 +$$
$$+ c_1 \cdot x_1 + c_1 \cdot c_2 + c_2 x_2 + c_1 c_2 + c_3 + c_1 \cdot c_2 + c_3 = x_1 x_2,$$

поскольку сложение на множестве $\boldsymbol{B}$ удовлетворяет тождеству $u + u = 0$. Итак, замыкание класса K содержит произведение $x \cdot y$.□

§6. Критерий полноты классов булевых функций

Параграф посвящен доказательству (и обсуждению) критерия полноты класса булевых функций, который называется теоремой Поста.

Определение. Классы T_0, T_1, S, M и L называются *основными замкнутыми классами* функций.

Следующее утверждение в литературе называется теоремой Поста.

Теорема 5.4. Класс булевых функций K является полным тогда и только тогда, когда он не содержится ни в одном из основных замкнутых классов.

Доказательство. Докажем необходимость. Пусть K – полный класс. Если K содержится в одном из основных замкнутых классов, скажем $K \subseteq T_0$, то $[K] \subseteq [T_0]$. Но $[K]$ – класс всех булевых функций, $[T_0] = T_0$ и, следовательно, любая булева функция сохраняет 0. Противоречие показывает, что K не содержится в T_0. Аналогично доказывается, что K не содержится и в других основных замкнутых классах.

Перейдем к доказательству достаточности. Пусть K не содержится ни в одном из классов T_0, T_1, S, M и L. Тогда класс K содержит функции $f_0(x_1, \ldots, x_n)$, $f_1(x_1, \ldots, x_n)$, $f_2(x_1,$

$\ldots, x_n)$, $f_3(x_1, \ldots, x_n)$, $f_4(x_1, \ldots, x_n)$, такие, что $f_0 \notin T_0$, $f_1 \notin T_1$, $f_2 \notin S$, $f_3 \notin M$ и $f_4 \notin L$.

Докажем, что $[K]$ содержит $\neg x$ и $x \cdot y$. Так как $f_0 \notin T_0$, имеем $f_0(0, \ldots, 0) = 1$. Относительно значения $f_0(1, \ldots, 1)$ рассмотрим два случая.

Случай 1: $f_0(1, \ldots, 1) = 0$. Рассмотрим функцию $g(x) = f(x, \ldots, x)$. Тогда $g(0) = f_0(0, \ldots, 0) = 1$ и $g(1) = f_0(1, \ldots, 1) = 1$, т.е. $g(x) = \neg x$. Это означает, что замыкание класса K содержит отрицание. Классу K принадлежит несамодвойственная функция $f_2(x_1, \ldots, x_n)$. По лемме о несамодвойственной функции замыкание класса K содержит константы. Поскольку K содержит нелинейную функцию $f_4(x_1, \ldots, x_n)$, то в силу леммы о нелинейной функции $[K]$ содержит произведение. Мы доказали, что в первом случае $[K]$ содержит $\neg x$ и $x \cdot y$.

Случай 2: $f_0(1, \ldots, 1) = 1$. Пусть $g(x) = f_0(x, \ldots, x)$ и $h(x) = f_1(g(x), \ldots, g(x))$. Тогда

$$g(0) = f_0(0, \ldots, 0) = 1 = f_0(1, \ldots, 1) = g(1),$$
$$h(0) = f_1(g_1(0), \ldots, g_1(0)) = f_1(1, \ldots, 1) = 0 =$$
$$= f_1(g(1), \ldots, g(1)) = h(1),$$

поскольку f_1 не сохраняет 1. Это означает, что константы принадлежат $[K]$. Так как K содержит немонотонную функцию, то по лемме о немонотонной функции, классу $[K]$ принадлежит отрицание. Далее, f_4 – нелинейная функция из K, поэтому по лемме о нелинейной функции $[K]$ содержит и произведение $x \cdot y$.

Итак, в обоих случаях, замыканию $[K]$ принадлежит $\neg x$ и $x \cdot y$. Поскольку $\{\neg x, x \cdot y\}$ – полный класс, то K – также полный класс. □

В доказательстве достаточности устанавливается более сильное утверждение: если K не содержится ни в одном из основных замкнутых классов, то в K найдутся пять функций $f_0, \ldots, f_4$ таких, что класс $\{f_0, \ldots, f_4\}$ является полным. На самом деле, внимательное рассмотрение доказательства достаточности позволяет получить следующее утверждение.

Теорема 5.5. Каждый полный класс содержит полный подкласс, состоящий из не более, чем четырех функций.

Доказательство. Пусть K – полный класс. Тогда по теореме 5.4 класс K не содержится ни в одном из основных замкнутых классов. В K найдутся, как и в доказательстве предыдущей теоремы, функции $f_0, \ldots, f_4$. При рассмотрении

случая 1 использованы три функции f_0, f_2 и f_4, т.е. в этом случае $\{f_0, f_2, f_4\}$ – полный подкласс класса K. В случае 2 использованы четыре функции: f_0, f_1, f_3 и f_4 . В этом случае $\{f_0, f_2, f_3, f_4\}$ – полный подкласс класса K. Итак, в обоих случаях в K найдется полный подкласс, содержащий не более четырех функций. □

Приведем пример, показывающий, что в формулировке теоремы 5.5 оценку 4 нельзя заменить на 3. Рассмотрим класс

$$K = \{\theta(x), \iota(x), x \cdot y, x+y+z\}.$$

Очевидно, что $\theta(x)$ не принадлежит T_1, $\iota(x)$ не принадлежит T_0. Функция $x \cdot y$ не является, как отмечалось, ни самодвойственной, ни линейной. Нетрудно проверить, что $x+y+z$ – немонотонная функция. Следовательно, по теореме Поста K – полный класс. В то же время, любой его собственный подкласс полным не является. Действительно, если удалить функцию $\theta(x)$, то оставшиеся функции будут сохранять 1, если удалить $\iota(x)$, то будут сохранять 0. Класс функций K без функции $x \cdot y$ состоит из линейных функций, а без последней функции – из монотонных.

Параграф закончим рассмотрением следующего понятия.

Определение. Замкнутый класс K называется *предполным*, если K – неполный класс, но для любой функции $f \notin K$ класс $K \cup \{f\}$ является полным.

Теорема 5.6. Предполными являются классы T_0, T_1, S, M и L и только они.

Доказательство. Необходимость докажем на примере класса T_0. Класс T_0 содержит несамодвойственную и нелинейную функцию $x \cdot y$, немонотонную функцию $x+y$ и функцию $\theta(x)$, не сохраняющую 1. Если $f \notin T_0$, то класс $L = \{\theta(x), f, x \cdot y, x+y\}$ не содержится ни в одном из основных замкнутых классов и поэтому L – полный класс. Поскольку $L \subseteq T_0 \cup \{f\}$, класс $T_0 \cup \{f\}$ также является полным.

Докажем достаточность. Пусть K – предполный класс. Поскольку класс K не является полным, то K содержится в одном из основных замкнутых классов. Для определенности предположим, что $K \subseteq T_0$. Если $K \subset T_0$, то существует функция f такая, что $f \notin K$ и $f \in T_0$. Тогда $K \cup \{f\} \subseteq T_0$ и класс $K \cup \{f\}$ не может быть полным. Следовательно, $K = T_0$. □

Эта теорема фактически является вторым вариантом теоремы Поста.

В заключение параграфа рассмотрим вопрос о «самых маленьких» полных классах функций – классах, состоящих из одной функции.

Определение. Булева функция f называется *шефферовой*, если класс, состоящий из единственной функции f, является полным.

Предложение. Среди двухместных функций шефферовыми являются штрих Шеффера и стрелка Пирса и только они.

Доказательство предложения предоставляется читателю.

§ 7. Сокращенные ДНФ

Этот и следующий параграфы посвящены задаче минимизации дизъюнктивных нормальных форм. Задача состоит в том, чтобы по данной булевой функции найти ДНФ, которая задает эту функцию и содержит наименее возможное количество литералов.

В этом и следующем параграфах нам будет удобно отрицание обозначать чертой сверху, атомарные формулы называть пропозициональными переменными и обозначать малыми латинскими буквами. Будем считать, что элементарные конъюнкции не содержат повторяющихся и противоположных литералов (иногда такие элементарные конъюнкции называют правильными). Число переменных элементарной конъюнкции будем называть ее *рангом*. Условимся также о следующем обозначении. Пусть $\sigma \in \boldsymbol{B}$. Тогда x^σ есть x, если $\sigma = 1$, и есть $\bar{x}$, если $\sigma = 0$.

Определение. Две ДНФ, определяющие равные булевы функции, называются *эквивалентными*.

Напомним, что в первом параграфе равными были названы функции, у которых существенные переменные одни и те же, и которые совпадают на любом наборе значений существенных переменных. Например, ДНФ $x_1x_2 \vee x_1\bar{x}_2$ и $x_1x_3 \vee x_1\bar{x}_3$ эквивалентны, а ДНФ $x_1x_2 \vee x_1x_3$ и $x_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3$ эквивалентными не являются.

Определение. Элементарная конъюнкция g называется *импликантой* булевой функции f, если для любых значений переменных из равенства $g = 1$ следует равенство $f = 1$.

Ясно, что если функция f задана некоторой ДНФ, всякая элементарная конъюнкция этой ДНФ является импликантой функции f.

Если g – элементарная конъюнкция (не являющаяся литералом), содержащая литерал L, то элементарную конъюнкцию, которая получается из g удалением литерала L, будем обозначать через g_L.

Определение. Импликанта g функции f называется *простой*, если для любого ее литерала L элементарная конъюнкция g_L не является импликантой функции f.

Другими словами, простая импликанта – это такая импликанта функции f, которая после удаления любого литерала перестает быть импликантой функции f.

Пример 1. Пусть функция f задается ДНФ $x_1x_2 \vee x_1x_2x_3$. Убедимся в том, что элементарная конъюнкция $g = x_1x_2$ является простой импликантой функции f. Ясно, что она является импликантой функции f. Если же удалить одну из переменных конъюнкции g, скажем x_1, то получившаяся элементарная конъюнкция $g_1 = x_2$ уже не будет импликантой функции f. Действительно, если $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, то $g_1 = 1$ и $f = 0$. Итак, g – простая импликанта функции f. Другая импликанта функции f – элементарная конъюнкция $h = x_1x_2x_3$ не является простой, так как после удаления переменной x_3 «превращается» в импликанту g. □

Импликант у данной функции (которая не является тождественно ложной) бесконечно много. Действительно, если f – функция от переменных $x_1, x_2, \ldots, x_n$ и $f(a_1, a_2, \ldots, a_n) = 1$ для $a_1, a_2, \ldots, a_n \in \{0, 1\}$, то элементарная конъюнкция $x_1^{a1} x_2^{a2} \ldots x_n^{an}$ является импликантой функции f. С другой стороны, если g – импликанта функции f и x_l – переменная, отличная от переменных функции f и не содержащаяся в элементарной конъюнкции g, то элементарная конъюнкция gx_l – импликанта функции f. Осталось напомнить, что множество переменных бесконечно. В то же время множество простых импликант булевой функции конечно, поскольку простая импликанта должна содержать только существенные переменные булевой функции.

Введем основное понятие данного параграфа.

Определение. ДНФ называется *сокращенной*, если она является дизъюнкцией всех своих простых импликант.

Пример 2. ДНФ $g_1 = x_1 \vee x_2x_3$ является сокращенной, так как она имеет только две простых импликанты x_1 и x_2x_3

и является их дизъюнкцией. ДНФ $g_2 = x_1x_2 \vee \bar{x}_1x_2x_3$ сокращенной не является. Нетрудно проверить, что g_2 имеет единственную простую импликанту – переменную x_2. □

Как было сказано, цель данного и следующего параграфов – изучить способ нахождения ДНФ, которая задает эту функцию и содержит наименее возможное количество литералов. В качестве первого (и основного) шага изучим способ построения сокращенной ДНФ по данной булевой функции f.

Мы будем предполагать, что *f не является тождественно ложной и задается СДНФ*.

Рассмотрим следующие преобразования, применяемые к паре элементарных конъюнкций:

1) $xg \vee \bar{x}g \to xg \vee \bar{x}g \vee g$ (неполное склеивание),

2) $xg \vee g \to g$ (поглощение),

3) $\bar{x}g \vee g \to g$ (поглощение),

где g – элементарная конъюнкция, x – переменная.

Применять эти преобразования будем следующим образом. Пусть исходная СДНФ содержит n переменных. Применяем преобразования 1 – 3 в указанном порядке к элементарным конъюнкциям ранга n до тех пор, пока эти преобразования применимы. Затем их применяем к элементарным конъюнкциям ранга $n - 1$ и т. д.

Следующее утверждение в литературе называется теоремой Куайна.

Теорема 5.7. Пусть булева функция представлена СДНФ f. Тогда в результате применения к f описанной выше процедуры получается сокращенная ДНФ.

Рассмотрим примеры.

Пример 3. Пусть

$$f = x_1x_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1x_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3.$$

Применяя преобразование 1, получаем

$$f_1 = f \vee x_2x_3 \vee x_1x_2 \vee \bar{x}_1x_2 \vee x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_3.$$

Так как элементарные конъюнкции формулы f участвовали в преобразовании 1, после применения преобразований 2 и 3 получим ДНФ

$$f_2 = x_2x_3 \vee x_1x_2 \vee \bar{x}_1x_2 \vee x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_3.$$

Действуем преобразованием 1 на формулу f_2, получим

$$f_3 = f_2 \vee x_2.$$

После преобразований 2 и 3 формула f_3 превратится в формулу

$$f_4 = \bar{x}_1 \bar{x}_3 \vee x_2.$$

К формуле f_4 не применимо ни одно из преобразований 1 – 3. Следовательно, f_4 – сокращенная ДНФ формулы f_1. □

Пример 4. Пусть

$$f = x_1x_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3.$$

Применим к формуле f последовательно преобразования 1, 2 и 3, получим формулы

$$f_1 = f \vee x_2x_3 \vee x_1x_3 \vee x_1x_2,$$
$$f_2 = x_2x_3 \vee x_1x_3 \vee x_1x_2.$$

К формуле f_2 не применимо ни одно из преобразований 1 – 3, следовательно, f_2 – сокращенная ДНФ формулы f_1. □

Этот пример показывает, что преобразование 1 (неполное склеивание) нельзя заменить на преобразование 1′: $xg \vee \bar{x}g \to g$ (полное склеивание). Действительно, после проведения преобразования 1′, формула f превратилась бы в формулу $x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3$ и дальнейшее применение преобразований 1′, 2 и 3 стало бы невозможным.

Напомним, что для проведения преобразований 1 – 3 исходная ДНФ должна быть совершенной. Следующий пример показывает, что это условие существенно.

Пример 5. Рассмотрим ДНФ

$$g = x_1x_2x_3 \vee x_1x_4 \vee x_2\bar{x}_4.$$

К ней преобразования 1 – 3 неприменимы. Тем не менее она сокращенной не является, поскольку импликанта $x_1x_2x_3$ не является простой (можно вычеркнуть переменную x_3). Для того, чтобы получить сокращенную ДНФ, эквивалентную g, надо взять СДНФ

$$h = x_1x_2x_3x_4 \vee x_1x_2x_3\bar{x}_4 \vee x_1\bar{x}_2x_3x_4 \vee x_1x_2\bar{x}_3x_4 \vee$$
$$\vee x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_3\bar{x}_4 \vee x_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4,$$

эквивалентную g, и преобразования 1 – 3 применить к h. В результате получается сокращенная ДНФ $x_1x_2 \vee x_1x_4 \vee x_2\bar{x}_4$ эквивалентная g. □

Покажем на примерах, как найти сокращенную ДНФ, если исходная булева функция задана картой Карно и на n-мерном кубе

Пример 6. Пусть булева функция f задана картой Карно на рис. 5.9.

x_1x_2 \ x_3x_4	00	01	11	10
00	1	1		1
01		1	1	
11	1	1	1	
10		1		

Рис. 5.9

Выделим на этой карте максимальные прямоугольники, состоящие из «единиц» и имеющие площадь 2^l клеток для некоторого l. В нашем случае таких прямоугольников будет пять (см. рис. 5.10).

Напомним, что карта Карно представляет из себя тор, поэтому две «единицы» с «координатами» 0000 и 0010 образуют максимальный прямоугольник требуемой площади. На рисунке он отмечен двумя прямоугольниками «без стороны». Каждый из максимальных прямоугольников определяет элементарную конъюнкцию, которая указана на рисунке. Этими элементарными конъюнкциями исчерпываются

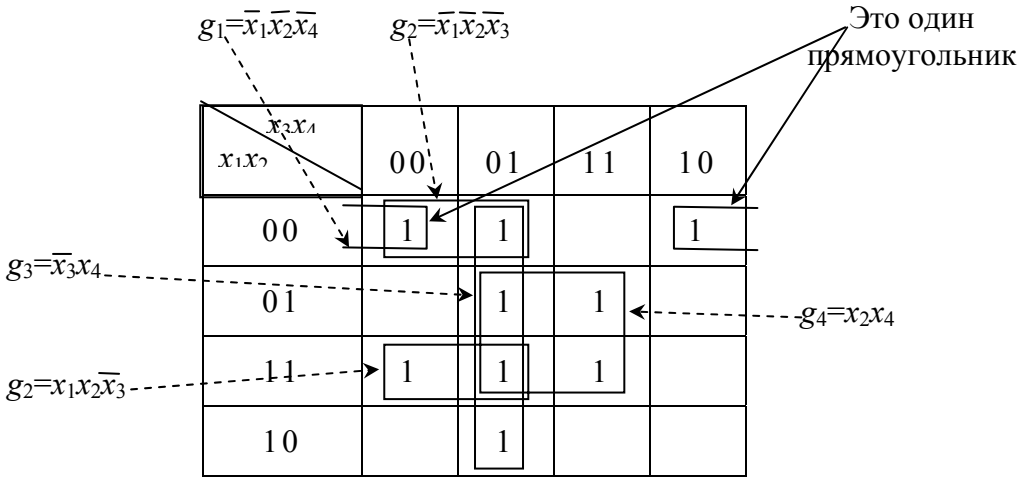


Рис. 5.10

все простые импликанты функции f. Удалим, например, у элементарной конъюнкции $g_1 = \bar{x}_1 \bar{x}_2 \bar{x}_4$ последний литерал $\bar{x}_4$. Получим элементарную конъюнкцию $h = \bar{x}_1 \bar{x}_2$, которая уже не будет импликантой функции f. Действительно, если $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, то $h = 1$, но $f = 0$. Итак, сокращенная ДНФ функции, заданной на рис. 5.9, есть дизъюнкция $g_1 \vee g_2 \vee g_3 \vee g_4 \vee g_5$ (см. рис. 5.10). □

Пример 7. Функцию f, рассмотренную в предыдущем примере, зададим на 4-х мерном кубе (см. рис. 5.11).

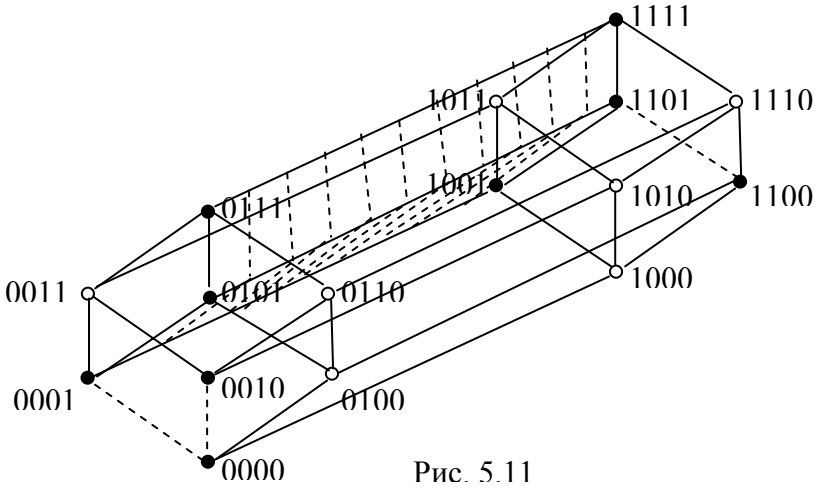


Рис. 5.11

Пусть $N_f = \{(x_1, x_2, x_3, x_4) | f(x_1, x_2, x_3, x_4) = 1\}$. Выделим в N_f грани максимальной размерности. Одномерные грани максимальной размерности на рис. 5.11 изображены прерывистой линией, двумерные грани максимальной размерности, содержащиеся в N_f, заштрихованы пунктирными линиями. Грани максимальной размерности определяют элементарные конъюнкции, входящие в сокращенную ДНФ. □

§ 8. Минимальные ДНФ

Сформулируем, наконец, в явном виде определение минимальной ДНФ, о котором неявно говорилось в предыдущем параграфе.

Определение. ДНФ называется *минимальной*, если она содержит наименьшее число литералов среди всех ДНФ, ей эквивалентных.

Пример 1. ДНФ $g_1 = x_1 \vee x_2x_3$ является минимальной. Убедимся в этом. ДНФ g содержит два литерала. Пусть ДНФ h эквивалентна g_1. Все три переменные ДНФ g_1 существенны. Следовательно, h должна содержать эти переменные. Но это означает, что h содержит, по крайней мере, три литерала. Итак, ДНФ g_1 является минимальной.

ДНФ $g_2 = x_1 \vee \bar{x}_1x_2$ минимальной не является, так как она эквивалентна ДНФ $x_1 \vee x_2$, содержащей меньше литералов. □

Предложение. Минимальная ДНФ состоит из простых импликант.

Доказательство. Докажем, что всякая минимальная ДНФ является дизъюнкцией своих простых импликант. Пусть минимальная ДНФ f равна $g_1 \vee g_2 \vee \ldots \vee g_k$, где g_1, g_2,

..., g_k – элементарные конъюнкции, и одна из этих элементарных конъюнкций, скажем g_1, не является простой импликантой функции, определяемой формой f. Тогда существует литерал, после вычеркивания которого g_1 остается импликантой функции f. Можно считать, что этот литерал является переменой x. Это означает, что $g_1 = g_1'x$ и g_1' – импликанта функции f. Рассмотрим ДНФ $f' = g_1' \vee g_2 \vee \ldots \vee g_k$. Убедимся в том, что и f эквивалентны. Переменным ДНФ f придадим некоторые значения. Если $f = 1$ при этих значениях переменных, то очевидно, что $f' = 1$ при тех же значениях переменных. Пусть теперь $f' = 1$. Тогда одна из элементарных конъюнкций g_1', g_2, ... или g_k принимает значение 1. Если $g_i = 1$ при $2 \le i \le k$, то $f = 1$, так как f содержит g_i. Если же $g_1' = 1$, то по-прежнему $f = 1$, поскольку g_1' – импликанта f. Итак, ДНФ f и f' эквивалентны. Это противоречит тому, что f – минимальная ДНФ, так как f' содержит на один литерал меньше. Следовательно, все элементарные конъюнкции ДНФ f являются ее простыми импликантами. □

В предыдущем параграфе мы отмечали, что первым шагом к получению минимальной ДНФ, задающей данную функцию, является нахождение сокращенной ДНФ. Вторым шагом к этому является нахождение ядра сокращенной ДНФ. Дадим соответствующие определения.

Определение. Пусть g и h элементарные конъюнкции. Будем говорить, что g *покрывает* h, если любой литерал из g содержится в h.

Другими словами, элементарная конъюнкция g *покрывает* элементарную конъюнкцию h, если g является импликантой h. Например, $g_1 = x_1x_2$ покрывает $h = x_1x_2x_3$, а $g_1 = \bar{x}_1x_2$ не покрывает h.

Определение. Простая импликанта называется *ядровой*, если она покрывает некоторую элементарную конъюнкцию исходной СДНФ, не покрываемую другой простой импликантой. Множество ядровых импликант называется *ядром* сокращенной ДНФ.

Из пяти простых импликант сокращенной ДНФ, заданной на рис. 5.10, все, кроме g_2, являются ядровыми.

Отметим, что ядро сокращенной ДНФ может быть пустым, а может содержать все простые импликанты (см. рис. 5.12 и 5.13).

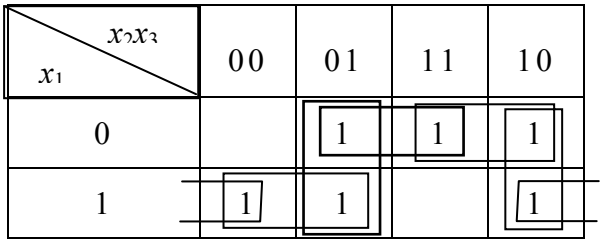


Рис. 5.12

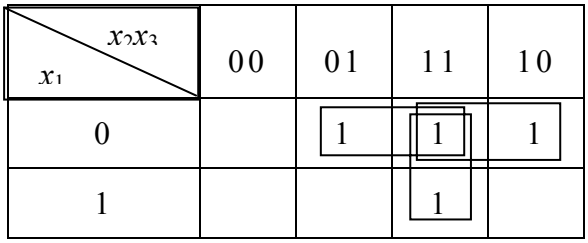


Рис. 5.13

Изложим третий, завершающий шаг способа нахождения минимальных ДНФ. Покажем, как находятся так называемые тупиковые ДНФ, а из них уже простым перебором – минимальные ДНФ.

Определение. Простая импликанта называется *избыточной*, если ее можно удалить без потери эквивалентности исходной СДНФ.

Пример 2. Рассмотрим функцию f, задаваемую картой Карно на рис. 5.14.

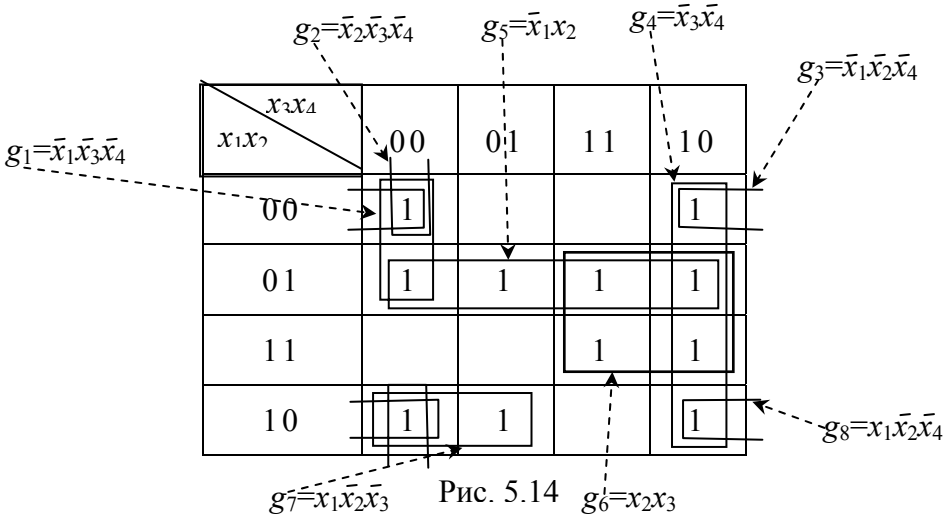


Рис. 5.14

Как видно на рисунке, сокращенная ДНФ содержит 8 простых импликант. Ядро сокращенной ДНФ состоит из трех простых импликант: g_5, g_6 и g_7. Остальные импликанты являются избыточными. Каждую из них можно удалить без потери эквивалентности СДНФ, задающей функцию f. □

Определение. ДНФ, состоящая из простых импликант исходной СДНФ, содержащая все ядровые импликанты и не содержащая (своих) избыточных импликант, называется *тупиковой*.

Например, ДНФ $g_1 \vee g_4 \vee g_5 \vee g_6 \vee g_7$ и $g_3 \vee g_5 \vee g_6 \vee g_7 \vee g_8$ являются тупиковыми (см. рис. 5.14).

Сформулируем очевидное

Предложение 2. Всякая минимальная ДНФ является тупиковой.

Задачу нахождения минимальных ДНФ, задающих данную булеву функцию можно решить следующим образом. Сначала найти все тупиковые ДНФ (их, как правило, оказывается немного). Затем простым просмотром выбрать их них минимальные.

Тупиковая ДНФ получается из сокращенной удалением максимального количества избыточных импликант сокращенной ДНФ так, чтобы сохранилась эквивалентность исходной СДНФ.

На примере булевой функции f, заданной картой Карно на рис. 5.14, покажем, как это можно сделать. Тупиковая ДНФ должна содержать дизъюнкцию ядровых импликант, т. е. $g_5 \vee g_6 \vee g_7$. Этой дизъюнкцией не покрываются три единицы карты, которые получаются при следующих наборах значений аргументов: 0000, 0010, 1010. Составим таблицу, строки которой индексируются этими наборами, а столбцы – избыточными импликантами сокращенной ДНФ (см. таблицу 5.7).

Таблица 5.7

	g_1	g_2	g_3	g_4	g_8
0000	+	+	+		
0010			+	+	
1010				+	+

В клетках таблицы ставим знак «+», если единица карты покрывается импликантой. В противном случае клетку оставляем пустой. Надо найти все минимальные подмножества столбцов, «плюсы» которых покрывают все строки. В нашем примере такими подмножествами будут следующие: $\{g_1, g_4\}$, $\{g_2, g_4\}$, $\{g_3, g_4\}$, $\{g_3, g_8\}$. Следовательно, булева функция имеет четыре тупиковых ДНФ:

$$h_1 = g_5 \vee g_6 \vee g_7 \vee g_1 \vee g_4,$$
$$h_2 = g_5 \vee g_6 \vee g_7 \vee g_2 \vee g_4,$$
$$h_3 = g_5 \vee g_6 \vee g_7 \vee g_3 \vee g_4,$$
$$h_4 = g_5 \vee g_6 \vee g_7 \vee g_3 \vee g_8.$$

ДНФ h_1, h_2, h_4 являются минимальными, все они содержат по 12 литералов. ДНФ h_3 содержит 13 литералов, и поэтому не является минимальной.

§ 9. Функции k-значной логики. Способы задания

Напомним, что k – натуральное число, большее 1 (число истинностных значений), и $\boldsymbol{B}_k = \{0, 1, \ldots, k-1\}$.

Определение. *Функцией k-значной логики* называется (всюду определенная) функция вида

$$f\colon \boldsymbol{B}_k^{\,n} \to \boldsymbol{B}_k,$$

где n – число аргументов (переменных), $\boldsymbol{B}_k^{\,n}$ – декартова степень множества $\boldsymbol{B}_k$.

Другими словами, функция k-значной логики – это отображение множества n-элементных последовательностей, составленных из элементов множества $\{0, 1, \ldots, k-1\}$, в это множество. Допускается случай $n = 0$; такие функции называются *константами*. Класс всех функций k-значной логики обозначается через $\boldsymbol{P}_k$.

Как и в случае булевых функций, будем считать, что

– зафиксировано бесконечное множество переменных (аргументов),

– если дана функция k-значной логики, то зафиксировано обозначение ее переменных.

Функцию из $\boldsymbol{P}_k$ от n переменных можно задать таблицей, содержащей $n+1$ столбец. В первых n строках таблицы записываются всевозможные наборы значений аргументов. Нетрудно понять, что таких наборов будет k^n. Следовательно, число всех функций от переменных $x_1, x_2, \ldots, x_n$ равно k^m, где $m = k^n$. Указанное число является очень большим. Например, число функций от переменных x_1, x_2 3-значной логики равно $3^9 = 19683$. В силу этого, возникают (в отличие от булева случая) трудности при использовании табличного задания функций, и такой способ задания обычно не применяется.

Функции k-значной логики мы будем задавать термами некоторой сигнатуры. Приведем список конкретных функций из $\boldsymbol{P}_k$, которые можно считать элементарными функциями. Функции, которые мы будем рассматривать в дальнейшем, будут суперпозициями элементарных функций. Та-

ким образом, имена функций из списка и составят сигнатуру.

1. $\neg x = x + 1 \ (mod\ k)$. При $k = 2$ функция $\neg x$ совпадает с обычным отрицанием, т. е. функция является обобщением отрицания в двузначной логике. Мы будем называть ее *циклическим отрицанием*.

2. $\sim x = (k - 1) - x$. Как и предыдущая, функция $\sim x$ является обобщением обычного отрицания. Она называется *отрицанием Лукасевича*.

На рисунках 5.15 и 5.16 приведены графики функций $\neg x$ и $\sim x$ соответственно при k =6.

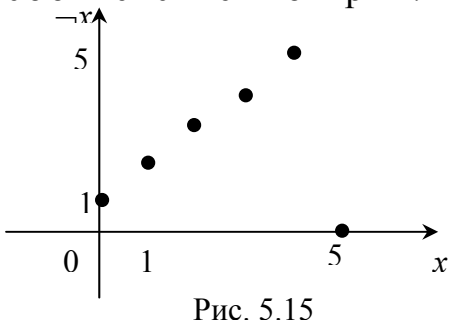


Рис. 5.15

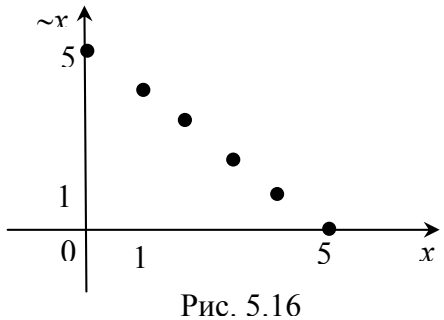


Рис. 5.16

3. Пусть $0 \le l \le k$. Следующую функцию будем называть *малой характеристической функцией* числа l:

$$j_l(x) = \begin{cases} 1, \text{ если } x = l, \\ 0, \text{ если } x \ne l. \end{cases}$$

4. Пусть, как и выше, $0 \le l \le k$. Функцию $J_l(x)$, определенную ниже, будем называть *большой характеристической функцией* числа l:

$$J_l(x) = \begin{cases} k\text{-}1, \text{ если } x = l, \\ 0, \text{ если } x \ne l. \end{cases}$$

На рисунках 5.17 и 5.18 приведены графики функций $j_2(x)$ и $J_2(x)$ соответственно при k =6.

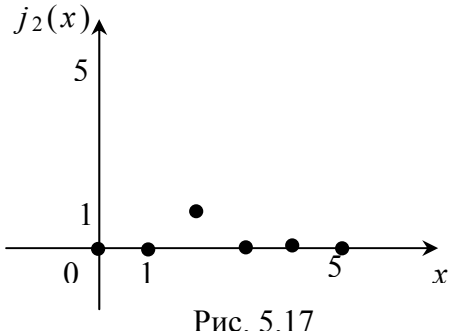


Рис. 5.17

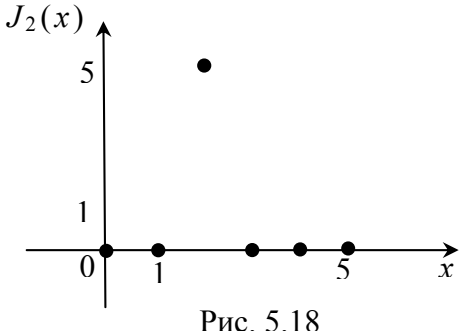


Рис. 5.18

5. $min(x_1, x_2)$. Функцию можно считать обобщением конъюнкции. Иногда будем ее обозначать через $x_1 \& x_2$.

6. $max(x_1, x_2)$. Эту функцию можно считать обобщением дизъюнкции. Иногда будем ее обозначать через $x_1 \vee x_2$.

7. $x_1 \cdot x_2$ $(mod\ k)$. В двузначном случае функция совпадает с конъюнкцией, поэтому ее можно считать вторым обобщением конъюнкции.

8. $x_1 + x_2$ $(mod\ k)$. Относительно этой операции множество $\boldsymbol{B}_k$ является абелевой группой. На $\boldsymbol{B}_k$ естественным образом определяется вычитание (по модулю k): $x - y = x + (k-1)y$.

Условимся, что арифметические операции – сложение, вычитание, умножение – на множестве $\boldsymbol{B}_k$ проводятся только по модулю k, и *в записи этих операций явно указывать модуль не будем*.

9. $$x \div y = \begin{cases} x - y, \text{ если } x \geq y, \\ 0, \text{ если } x < y. \end{cases}$$

Функция $x \div y$ иногда называется «*усеченной*» *разностью*.

10. $p_i(x_1, x_2, \ldots, x_n) = x_i$, где $1 \leq i \leq k$. Эти функции будем называть *проекциями*. Обратим внимание на то, что $p_1(x_1)$ – тождественная функция.

Введенные операции, конечно, обладают рядом свойств. Сложение, умножение, функции $min(x_1, x_2)$ и $max(x_1, x_2)$ ассоциативны и коммутативны. Отрицание Лукасевича связано с последними двумя функциями равенствами

$$\sim min(x_1, x_2) = max(\sim x_1, \sim x_2),$$
$$\sim max(x_1, x_2) = min(\sim x_1, \sim x_2),$$

которые будем называть законами де Моргана. В то же время, ряд свойств, привычных для нас в двузначной логике, в k-значной логике не выполняется. В частности, нет закона снятия двойного циклического отрицания, т. е. $\neg(\neg x) \neq x$.

Как и в двузначном случае, вводится понятие фиктивной и существенной переменной функции и понятие равенства двух функций. Следовательно, функции f и g из $\boldsymbol{P}_k$ равны, если существенные переменные этих функций одни и те же, и значения функций f и g совпадают на любом наборе значений существенных переменных.

§ 10. Функции k-значной логики. Замкнутость и полнота

Определения замкнутости и полноты класса функций k-значной логики полностью совпадают с соответствующими

определениями для $\boldsymbol{P}_2$. Это означает, что класс L функций k-значной логики замкнут, если он содержит тождественную функцию и замкнут относительно суперпозиции и переименования переменных. Класс L является полным, если любая функция k-значной логики может быть получена из функций, принадлежащих L и тождественной функции с помощью суперпозиции и переименования переменных.

Параграф посвящен доказательству полноты некоторых классов функций.

Теорема 5.8. Класс функций

$F_1 = \{0, 1, \ldots, k\text{-}1, J_0(x), \ldots, J_{k-1}(x), min(x_1, x_2), max(x_1, x_2)\}$

является полным в $\boldsymbol{P}_k$.

Доказательство. Утверждение теоремы очевидным образом следует из равенства

$$f(x_1, \ldots, x_n) = max\{min(J_{a1}(x_1), J_{a2}(x_2), \ldots, J_{an}(x_n), f(a_1, a_2, .., a_n)) \mid a_1, \ldots, a_n \in \boldsymbol{B}_k\}. \quad (1)$$

Докажем это равенство. Пусть $x_1 = b_1$, $x_2 = b_2$, ..., $x_n = b_n$. Возьмем вектор $(a_1, a_2, \ldots, a_n)$ из $\boldsymbol{B}_k^{\,n}$.

Если $(a_1, a_2, \ldots, a_n) \neq (b_1, b_2, \ldots, b_n)$; скажем $a_i \neq b_i$ для некоторого i, то $J_{ai}(x_i) = J_{ai}(b_i) = 0$ и

$$min(J_{a1}(b_1), J_{a2}(b_2), \ldots, J_{an}(b_n), f(a_1, .., a_n)) = 0.$$

Пусть $(a_1, a_2, \ldots, a_n) = (b_1, b_2, \ldots, b_n)$, т. е. $a_i = b_i$ для всех i. Тогда

$$min(J_{a1}(b_1), J_{a2}(b_2), \ldots, J_{an}(b_n), f(a_1, a_2, .., a_n)) =$$
$$= min(k-1, k-1, \ldots, k-1, f(b_1, b_2, .., b_n)) = f(b_1, b_2, .., b_n).$$

Следовательно,

$$max\{min(J_{a1}(b_1), J_{a2}(b_2), \ldots, J_{an}(b_n), f(a_1, a_2, .., a_n)) \mid a_1, \ldots, a_n \in \boldsymbol{B}_k\} = f(b_1, b_2, .., b_n).$$

Равенство (1) доказано. □

Теорема 5.9. Класс функций

$$F_2 = \{\neg x, max(x_1, x_2)\}$$

является полным в $\boldsymbol{P}_k$.

Доказательство проведем в четыре этапа.

Этап 1: докажем, что все константы принадлежат $[F_2]$. Напомним, что $\neg x = x+1$. Тогда $\neg(\neg x) = x+2$, $\neg(\neg(\neg x)) = x+3$ и т. д. Рассмотрим функцию

$$f(x) = max(x, x+1, \ldots, x+(k-1)).$$

Легко видеть, что функция $f(x)$ тождественно равна $k-1$. Тогда $0 = \neg f(x)$, $1 = \neg(\neg f(x))$ и т. д. Мы видим, что замыкание класса F_2 содержит все константы.

Этап 2: докажем, что $[F_2]$ содержит все функции $J_0(x)$, $J_1(x)$, ..., $J_{k-1}(x)$. Убедимся в том, что

$$J_l(x) = 1 + max\{x+a \mid a \neq k - (l+1)\}.$$

Сделаем это для $l = 0$. Для остальных значений l доказательство равенства проходит аналогичным образом. Сравним $J_0(x)$ и

$$1 + max\{x+a \mid a \neq k - 1\} = 1 + max\{x, x+1, \ldots, x+(k\text{-}2)\}.$$

Если $x = 0$, то $1 + max\{0, 0+1, \ldots, 0+(k\text{-}2)\} = 1 + (k\text{-}2) = k\text{-}1 = J_0(0)$. Если же $x \neq 0$, то $1 + max\{x, x+1, \ldots, x+(k\text{-}2)\} = 1 + (k\text{-}2) = 0 = J_0(x)$.

Этап 3: докажем, что $[F_2]$ принадлежат все одноместные функции. Рассмотрим функции

$$e_l(x, a) = \begin{cases} a, \text{ если } x = l, \\ \\ 0, \text{ если } x \neq l. \end{cases}$$

Эти функции принадлежат замыканию класса F_2, поскольку

$$e_l(x, a) = max(J_l(x), k-1-a) + a + 1. \qquad (2)$$

Докажем равенство (2). Если $x \neq l$, то $e_l(x, a) = 0$ и $max(J_l(x), k-1-a) + a + 1 = max(0, k-1-a) + a + 1 = 0$.

Если же $x = l$, то $e_l(x, a) = a$ и $max(J_l(l), k-1-a) + a + 1 = max(k-1, k-1-a) + a + 1 = a$.

Равенство (2) доказано.

Пусть $g(x)$ – некоторая одноместная функция и $x = b$. Тогда в последовательности

$$e_0(b, g(0)), e_1(b, g(1)), \ldots, e_{k\text{-}1}(b, g(k\text{-}1))$$

все элементы, кроме $e_b(b, g(b))$, равны 0, а $e_b(b, g(b)) = g(b)$. Это означает, что

$$g(x) = max(e_0(x, g(0)), e_1(x, g(1)), \ldots, e_{k\text{-}1}(x, g(k\text{-}1))).$$

Следовательно, функция $g(x)$ принадлежат замыканию класса F_2.

Этап 4: покажем, что $[F_2]$ содержит $min(x_1, x_2)$. В третьем этапе было доказано, что $\sim x \in [F_2]$. Осталось воспользоваться отмечавшимся выше законом де Моргана:

$$min(x_1, x_2) = \sim max(\sim x_1, \sim x_2).$$

Итак, все функции полного класса F_1 принадлежат замыканию класса F_2. Это означает, что F_2 – полный класс. □

Теорема 5.10. Класс F_3, состоящий из одной функции $max(x_1, x_2) + 1$ является полным в $\boldsymbol{P}_k$.

Теорема легко следует из теоремы 8, поскольку

$$max(x, x) + 1 = x + 1 = \neg x \text{ и}$$

$$max(x_1, x_2) + \underbrace{1 + 1 + \ldots + 1}_{k \text{ раз}} = max(x_1, x_2).$$

Определение. Функция k-значной логики называется *функцией Вебба*, если класс, состоящий из одной этой функции, является полным.

Теорема 5.10 утверждает, что $max(x_1, x_2) + 1$ – функция Вебба.

Теорема 5.11. Полными в $\boldsymbol{P}_k$ являются следующие классы функций

1) $F_4 = \{\neg x, min(x_1, x_2)\}$,

2) класс F_5, состоящий из одной функции $min(x_1, x_2) + 1$.

Полнота класса F_4 доказывается аналогично теореме 5.9. Второй пункт теоремы 5.11 легко следует из первого.

Теорема 5.12. Любой класс, полный в $\boldsymbol{P}_k$, содержит конечный подкласс, также полный в $\boldsymbol{P}_k$.

Доказательство. Пусть L – полный класс. Тогда функция $max(x_1, x_2) + 1$ получается из функций этого класса и тождественной функции с помощью суперпозиции и переименования переменных. Другими словами, эта функция определяется термом, в запись которого входят функции из L (и возможно, тождественная функция). Но в соответствующем терме содержится лишь конечное множество функций из L. Это множество и будет полным классом, поскольку $max(x_1, x_2) + 1$ – функция Вебба. □

§ 11. Классы сохранения отношений

Данный параграф является подготовительным к следующему параграфу. В нем рассматривается некоторый способ задания замкнутых классов функций.

Определение. Пусть r есть s-местное отношение, заданное на $\boldsymbol{B}_k$. Функция $f(x_1, x_2, \ldots, x_n)$ *сохраняет отношение* r, если из того, что

$$(a_{11}, a_{12}, \ldots, a_{1s}) \in r,$$
$$(a_{21}, a_{22}, \ldots, a_{2s}) \in r,$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$(a_{n1}, a_{n2}, \ldots, a_{ns}) \in r,$$

следует, что

$$(f(a_{11}, a_{21}, \ldots, a_{n1}), f(a_{12}, a_{22}, \ldots, a_{n2}), \ldots,$$
$$f(a_{1s}, a_{2s}, \ldots, a_{ns})) \in r.$$

Например, если r – отношение обычного порядка, то функция $min(x_1, x_2)$ сохраняет это отношение, а функция $x_1 + x_2$ не сохраняет. Действительно, если $a_{11} \le a_{12}$ и $a_{21} \le a_{22}$, то $min(a_{11}, a_{21}) \le min(a_{12}, a_{22})$. В то же время, $1 \le 1$ и $0 \le k-1$, но $1+0 > 1+(k-1) = 0$.

Любая функция сохраняет пустое и универсальное отношения.

Обозначим через $\boldsymbol{F}(r)$ класс функций, сохраняющих отношение r. Если R – множество отношений, то

$$\boldsymbol{F}(R) = \cap\{\boldsymbol{F}(r) \mid r \in R\}.$$

Теорема 5.13. Для любого множества отношений R класс функций $\boldsymbol{F}(R)$ является замкнутым.

Доказательство теоремы легко следует из определений.

Далее в качестве примеров будут рассмотрены классы булевых функций.

Пример 1: r – обычное отношение порядка, заданное на $\boldsymbol{B}_2$, т. е. $r = \{(0,0), (0,1), (1,1)\}$. Возьмем n-местную булеву функцию $f(x_1, x_2, \ldots, x_n)$. Сохраняемость этой функцией отношения r означает, что из неравенств

$$a_1 \le b_1,\ a_2 \le b_2,\ \ldots\ ,\ a_n \le b_n$$

следует неравенство

$$f(a_1, a_2, \ldots, a_n) \le f(b_1, b_2, \ldots, b_n).$$

Мы получили определение монотонной функции. Следовательно, $\boldsymbol{F}(r)$ – класс монотонных функций.□

Пример 2: $r = \{0\}$. Сохраняемость функцией $f(x_1, x_2, \ldots, x_n)$ отношения r означает, что из равенств

$$a_1 = 0,\ a_2 = 0,\ \ldots,\ a_n = 0$$

следует равенство $f(a_1, a_2, \ldots, a_n) = 0$. Другими словами, функция $f(x_1, x_2, \ldots, x_n)$ удовлетворяет условию $f(0, 0, \ldots, 0) = 0$. Это означает, что , $\boldsymbol{F}(r)$ – класс функций, сохраняющих 0. □

Пример 3: $r = \{1\}$. В этом случае получим класс функций, сохраняющих 1. □

Пример 4: $r = \{(0, 1), (1, 0)\}$. Возьмем n-местную булеву функцию $f(x_1, x_2, \ldots, x_n)$. Тот факт, что эта функция сохраняет отношение r, означает, что из условий

$$(a_1, b_1) \in r,\ (a_2, b_2) \in r,\ \ldots,\ (a_n, b_n) \in r \qquad (1)$$

следует условие

$$(f(a_1, a_2, \ldots, a_n), f(b_1, b_2, \ldots, b_n)) \in r. \qquad (2)$$

Заметим, что отношение r можно определить и так: $r = \{(0, \neg 0), (1, \neg 1)\}$. Тогда условия (1) и (2) можно записать следующим образом:

$$b_1 = \neg a_1,\ b_2 = \neg a_2,\ \ldots,\ b_n = \neg a_n, \qquad (1')$$

$$f(b_1, b_2, \ldots, b_n) = \neg f(a_1, a_2, \ldots, a_n) \qquad (2')$$

Из (1′) и (2′) следует равенство

$$f(\neg a_1, \neg a_2, \ldots, \neg a_n) = \neg f(a_1, a_2, \ldots, a_n),$$

которое есть не что иное, как определение самодвойственной функции. Следовательно, $\boldsymbol{F}(r)$ – класс самодвойственных функций. □

Пример 5: $r = \{(x_1, x_2, x_3, x_4) \mid x_1 + x_2 = x_3 + x_4\}$. Убедимся в том, что это на первый вгляд странное отношение приводит к классу линейных функций. Рассмотрим линейную функцию $f(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n + c_0$. Убедимся в том, что она сохраняет отношение r. Пусть

$$a_{11} + a_{12} = a_{13} + a_{14},$$
$$a_{21} + a_{22} = a_{23} + a_{24},$$
$$\ldots$$
$$a_{n1} + a_{n2} = a_{n3} + a_{n4}.$$

Первое равенство умножим на c_1, второе – на c_2 и т. д. Б последнее равенство умножим на c_n. Полученные равенства сложим. Мы получим, что

$$c_1(a_{11} + a_{12}) + c_2(a_{21} + a_{22}) + \ldots + c_n(a_{n1} + a_{n2}) =$$
$$c_1(a_{13} + a_{14}) + c_2(a_{23} + a_{24}) + \ldots + c_n(a_{n3} + a_{n4}) \text{ или}$$
$$(c_1a_{11} + c_2a_{21} + \ldots + c_na_{n1}) + (c_1a_{12} + c_2a_{22} + \ldots + c_na_{n2}) =$$
$$(c_1a_{13} + c_2a_{23} + \ldots + c_na_{n3}) + (c_1a_{14} + c_2a_{24} + \ldots + c_na_{n4}).$$

Последнее равенство можно записать так

$$f(a_{11}, a_{21}, \ldots, a_{n1}) + f(a_{12}, a_{22}, \ldots, a_{n2}) =$$
$$f(a_{13}, a_{23}, \ldots, a_{n3}) + f(a_{14}, a_{24}, \ldots, a_{n4}).$$

Это означает, что функция $f(x_1, x_2, \ldots, x_n)$ сохраняет отношение r.

Теперь докажем, что всякая функция, сохраняющая r, есть сумма одноместных функций и константы. Пусть функция $g(x_1, x_2, \ldots, x_n)$ сохраняет r. По определению отношения r имеем, что

$$(x_1, 0, x_1, 0) \in r,$$
$$(x_2, 0, 0, x_2) \in r,$$
$$\ldots$$
$$(x_n, 0, 0, x_n) \in r.$$

Тогда

$$g(x_1, x_2, \ldots, x_n) + g(0, 0, \ldots, 0) =$$
$$g(x_1, 0, \ldots, 0) + g(0, x_2, \ldots, x_n) \text{ или}$$
$$g(x_1, x_2, \ldots, x_n) =$$
$$g(x_1, 0, \ldots, 0) + g(0, x_2, \ldots, x_n) - g(0, 0, \ldots, 0).$$

Мы видим, что n-местная функция представима в виде суммы одноместной функции, константы и $(n\text{-}1)$-местной функции. Рассуждая аналогичным образом, $(n\text{-}1)$-местную функцию можно представить в виде суммы одноместной функции, константы и $(n\text{-}2)$-местной функции и т. д. В ито-

ге получим, что функция $g(x_1, x_2, \ldots, x_n)$ есть сумма одноместных функций и константы. Осталось заметить, что все одноместные функции (и константы) линейны. Следовательно, $\boldsymbol{F}(r)$ – класс линейных функций. □

§ 12. Критерий Розенберга

Естественно поставить вопрос о том, существует ли обобщение теоремы Поста для булевых функций на случай функций k-значной логики? Данный параграф посвящен ответу на этот вопрос. Как и в случае теоремы Поста, ответ будет дан в двух вариантах.

Как и для в случае булевых функций, *предполным* классом функций k-значной логики называется неполный замкнутый класс L, удовлетворяющий условию:

для любой функции $f \notin L$
класс $L \cup \{f\}$ является полным.

В булевом случае предполных классов, как известно, пять. С. Яблонским получено описание предполных классов трехзначной логики. Их оказалось 18.

Следующее утверждение известно в литературе как теорема А. Кузнецова.

Теорема 5.14. Для любого k число предполных в $\boldsymbol{P}_k$ классов конечно.

Доказывать теорему здесь не будем. С ее доказательством можно познакомиться по книге [Я, стр. 54-56].

Из доказательства этой теоремы нельзя получить описание предполных классов даже для 3-значной логики.

Одна из основных проблем здесь – найти подходящий способ задания замкнутых классов. В качестве такого способа мы будем использовать классы сохранения отношений.

Рассмотрим ряд определений.

Определение. n-местное отношение r при $n>1$, заданное на $\boldsymbol{B}_k$, называется *вполне рефлексивным*, если для любых a_1, a_2, …, $a_n \in \boldsymbol{B}_k$ выполняется условие:

$a_i = a_j$ при $i \neq j$ влечет $(a_1, a_2, \ldots, a_n) \in r$.

Любое одноместное отношение вполне рефлексивно.

Для двухместных (бинарных) отношений понятия рефлексивности и вполне рефлексивности совпадают. В качестве примера вполне рефлексивного трехместного отношения можно привести отношение компланарности трех векторов обычного пространства.

Определение. n-местное отношение r при $n>1$, заданное на $\boldsymbol{B}_k$, называется *вполне симметричным*, если для любых $a_1, a_2, \ldots, a_n \in \boldsymbol{B}_k$ и любой перестановки π индексов выполняется условие:

$(a_1, a_2, \ldots, a_n) \in r$ влечет $(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}) \in r$.

Любое одноместное отношение вполне симметрично.

Для двухместных (бинарных) отношений понятия симметричности и вполне симметричности совпадают. Вполне симметричность отношения означает независимость от порядка компонент вектора аргументов. Примерами вполне симметричных отношений являются приведенное выше отношение компланарности векторов, отношение для трех точек образовать треугольник, отношение взаимной простоты трех натуральных чисел (или чисел из $\boldsymbol{B}_k$).

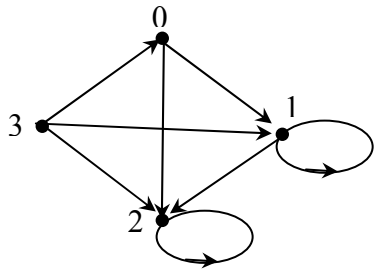


Рис. 5.18

Определение. *Центром* n-местного r отношения при $n > 1$ называется множество

$Z(r) = \{b \in \boldsymbol{B}_k | (x_1, \ldots, x_{n-1}, b) \in r$ для любых $x_1, \ldots, x_{n-1}$ из $\boldsymbol{B}_k\}$.

Центром одноместного отношения является оно само.

Рассмотрим двухместное отношение r, заданное на $\boldsymbol{B}_4$ и изображенное на рис. 5.18 в виде графа ($(a, b) \in r \Leftrightarrow$ существует дуга из a в b). Центр этого отношения состоит из одного числа 2. Если на рис. 5.18 дуги заменить на ребра, т. е. сделать отношение r симметричным, то $Z(r)$, будет равен $\{1, 2\}$.

Пусть $(G, \cdot)$ – группа и r – отношение перестановочности, т. е.

$$(x, y) \in r \Leftrightarrow x \cdot y = y \cdot x.$$

Тогда $Z(r)$ – центр группы G в обычном смысле, т. е. множество элементов, которые перестановочны с любым элементом группы.

Определение. Отношение r называется *центральным*, если оно вполне рефлексивно, вполне симметрично и имеет нетривиальный центр, т.е. непустой и не совпадающий с основным множеством.

Отношение, изображенное на рис. 13 является центральным. Отношение перестановочности на группе является центральным тогда и только тогда, когда группа коммутативна.

Определение. Пусть $\{\tau_1, \tau_2, \ldots, \tau_m\}$ – семейство эквивалентностей, заданных на $\boldsymbol{B}_k$. Это семейство называется *l-регулярным*, если выполняются следующие условия:

1) каждая из эквивалентностей имеет l классов,

2) если $A_1, A_2, \ldots, A_m$ – классы эквивалентностей $\tau_1, \tau_2, \ldots, \tau_m$ соответственно, то $A_1 \cap A_2 \cap \ldots \cap A_m \neq \varnothing$.

Приведем пример регулярного семейства эквивалентностей, заданного на $\boldsymbol{B}_9$. Возьмем квадратную матрицу третьего порядка, состоящую из различны элементов множества $\boldsymbol{B}_9$. Тогда строки матрицы будут классами разбиения эквивалентности τ_1, а столбцы – эквивалентности τ_2. Семейство эквивалентностей $\{\tau_1, \tau_2\}$ является 3-регулярным.

Определение. l-местное отношение r называется *l-регулярным*, если существует l-регулярное семейство эквивалентностей S такое, что выполняется условие

$$(x_1, x_2, \ldots, x_l) \in r \Leftrightarrow$$

$$(\forall \tau \in S)[(x_1, x_2) \in \tau \vee (x_1, x_3) \in \tau \vee \ldots \vee (x_{l-1}, x_1) \in \tau\}.$$

Заметим, что всякое l-регулярное отношение является вполне рефлексивным и вполне симметричным.

Мы готовы сформулировать критерий полноты класса функций k-значной логики, полученный американским математиком И. Розенбергом. Как было сказано в начале параграфа, мы сделаем это в двух вариантах. Это будут теоремы 5.15 и 5.16.

Теорема 5.15. Класс C функций k-значной логики является полным тогда и только тогда, C не содержится в классе сохранения ни одного из следующих отношений:

1) ограниченный частичный порядок,

2) граф нетождественной перестановки, являющейся произведением простых циклов,

3) четырехместное отношение $x_1 + x_2 = x_3 + x_4$, заданное на абелевой группе $(\boldsymbol{B}_k, +)$ простой экспоненты,

4) собственное отношение эквивалентности, т. е. отличное от отношения равенства и универсального отношения).

5) l-местное центральное отношение, где $1 \leq l \leq k$-1,

6) l-регулярное отношение, где $3 \leq l \leq k$.

Доказательство теоремы является очень сложным (что понятно по формулировке) и здесь рассматриваться не будет. С доказательством можно познакомиться по статье [].

Теорема 5.16. Класс C функций k-значной логики является предполным тогда и только тогда, C является классом

сохранения отношения одного из типов 1) – 6), указанных в формулировке теоремы 5.15.

Рассмотрим несколько примеров применения теоремы 5.15.

Пример 1. Применим эту теорему в случае $k = 2$, т. е. для класса булевых функций.

Ограниченных частичных порядков, заданных на $\boldsymbol{B}_2 = \{0, 1\}$ все два: естественный $r_1 = \{(0,0), (0,1), (1,1)\}$ и обратный естественному $r_2 = \{(1,1), (1,0), (0,0)\}$. Нетрудно проверить, что $\boldsymbol{F}(r_1) = \boldsymbol{F}(r_2)$. Класс $\boldsymbol{F}(r_1)$ найден в предыдущем параграфе (пример 1) – это класс M монотонных функций.

Нетождественная перестановка, заданная на $\{0, 1\}$ всего одна: $\pi(0) = 1$, $\pi(1) = 0$. Граф этой перестановки есть $r = \{(0, 1), (1, 0)\}$. В § 11 показано, что класс сохранения такого отношения есть класс S самодвойственных функций (пример 4).

Структуру абелевой группы на множестве $\boldsymbol{B}_2 = \{0, 1\}$ можно ввести двумя способами:

1) в качестве операции сложения взять обычное сложение по модулю 2,

2) операцию сложения определить так, что нейтральным элементом будет 1: $0 \oplus 0 = 1$, $0 \oplus 1 = 1 \oplus 0 = 0$, $1 \oplus 1 = 1$.

Четырехместное отношение, определяемое первой абелевой группой, обозначим через r_1, определяемое второй – через r_2. Легко проверить, что $r_1 = r_2$. Как показано в предыдущем параграфе (пример 5), $F(r_1)$ – класс L линейных функций.

Не существует собственного отношения эквивалентности, заданного на двухэлементном множестве.

Рассмотрим l-местное центральное отношение r. В силу ограничения $1 \le l \le k$-1, r – одноместное центральное отношение, т. е. непустое собственное подмножество множества $\{0, 1\}$. Таких подмножеств два и . Они, как показано в § 11 (примеры 2 и 3) определяют классы T_0 и T_1 соответственно.

В силу ограничения $2 < l \le k$, l-регулярных отношений в случае булевых функций не существует.

Итак, критерий Розенберга дает уже известные нам предполные классы функций: M, S, L, T_0 и T_1. □

Пример 2. Применим теорему 5.15 для доказательства полноты следующего класса функций k-значной логики:

$$G = \{\neg x, x \div y\}.$$

Пусть r – отношение, заданное на $\boldsymbol{B}_k$. Предположим, что функции класса G сохраняют r. Рассмотрим шесть случаев, со-

ответствующих шести типам отношений из формулировки теоремы 14.

Случай 1: *r – ограниченный частичный порядок*. Наименьший элемент обозначим буквой *a*, наибольший – буквой *b*. Тогда поскольку

$$(b, b) \in r \text{ и}$$
$$(0, b) \in r,$$

получаем, что

$$(b \div 0, b \div b) \in r, \text{ т. е.}$$
$$(0, b) \in r.$$

Так как *r* антисимметрично, из условий $(0, b) \in r$ и $(0, b) \in r$ получаем равенство $b = 0$. Далее аналогичным образом доказываем, что $a = 0$. Противоречие $b = 0 = a$.

Случай 2: *r – граф нетоджественной перестановки множества* $\boldsymbol{B}_k$. Тогда *r* не сохраняется константой 0, принадлежащей замыканию *G*, поскольку $0 = x \div x$. Действительно, в противном случае из условия $(a, b) \in r$ следует, что $(a \div a, b \div b) \in r$ т. е. $(0, 0) \in r$. Перестановка в случае 2 является произведением простых циклов, поэтому ее граф петель не имеет.

Случай 3: *r – четырехместное отношение* $x_1 + x_2 = x_3 + x_4$, *заданное на абелевой группе* $(\boldsymbol{B}_k, \oplus)$. Обозначим буквой θ нейтральный элемент этой группы. Поскольку функция $x \div y$ сохраняет *r*, из условий

$$(\theta, 0, 0, \theta) \in r,$$
$$(0, \theta, 0, \theta) \in r$$

следует, что $(\theta, 0, 0, 0) \in r$, т. е. что $\theta \oplus 0 = 0 \oplus 0$. Отсюда следует равенство $\theta = 0$. Далее, пусть *x* – произвольный элемент из $\boldsymbol{B}_k$. Из условий

$$(x, 0, x, 0) \in r,$$
$$(0, 0, x, y) \in r,$$

где *y* – противоположный к *x* элемент в группе $(\boldsymbol{B}_k, \oplus)$, следует, что $(x, 0, 0, 0) \in r$. Это означает, что $x \oplus 0 = 0 \oplus 0$ и $x = 0$. Противоречие показывает, что функция $x \div y$ не сохраняет *r*.

Случай 4: *r – собственное отношение эквивалентности, заданное на* $\boldsymbol{B}_k$. Существует класс эквивалентности, содержащий два различных элемента *a* и *b*. Считаем, что $a < b$. Поскольку функция $x \div y$ сохраняет *r*, из условий

$$(a, b) \in r,$$
$$(a, a) \in r$$

следует, что $(0, b - a) \in r$. (Здесь $b - a$ есть обычная разность чисел *b* и *a*.) Так как $(1, 1) \in r$, получаем последовательно $(0, b$

$- a - 1) \in r$, $(0, b - a - 2) \in r$, …, $(0, 1) \in r$. Пусть x – произвольный элемент из $\boldsymbol{B}_k$. Тогда $(x, x) \in r$ и $(x, x - 1) \in r$, поскольку $(x, x) \div (0,1) = (x, x - 1)$. Аналогично получим, что $(x, x - 2) \in r$, …, $(x, 0) \in r$. Следовательно, отношение эквивалентности r содержит только один класс, т. е. $r = \boldsymbol{B}_k \times \boldsymbol{B}_k$. Противоречие, так как r – собственное отношение эквивалентности.

Случай 5: *r есть l-местное центральное отношение*. Напомним, что центральное отношение имеет нетривиальный центр, т. е. непустой и не совпадающий с $\boldsymbol{B}_k$.

Предположим вначале, что $l = 1$. Пусть $a \in r$. Тогда множество $\{a, a + 1, a + 2, a + k - 1\}$ равно $\boldsymbol{B}_k$. Это означает, что $r = Z(r) = \boldsymbol{B}_k$. Противоречие показывает, что $\neg x$ не сохраняет отношение r.

Пусть теперь $l > 1$. Возьмем a из $Z(r)$. Тогда по определению центра для любых $x_1, x_2, \ldots, x_{l-1} \in \boldsymbol{B}_k$ выполняется условие

$$(x_1, x_2, \ldots, x_{l-1}, a) \in r.$$

Поскольку $\neg x$ сохраняет r, получаем, что

$$(x_1 + 1, x_2 + 1, \ldots, x_{l-1} + 1, a + 1) \in r.$$

Так как функция $\neg x$ – перестановка множества $\boldsymbol{B}_k$, отсюда следует, что $a + 1 \in Z(r)$. Аналогично получаем, что $a + 2$, …, $a + k - 1 \in Z(r)$ т. е. что $Z(r) = \boldsymbol{B}_k$. Противоречие показывает, что и в случае функция $\neg x$ не сохраняет r.

Случай 6: *r есть l-регулярное отношение и* $l > 2$. Пусть r определяется l-регулярным семейством эквивалентностей $S = \{\tau_1, \tau_2, \ldots, \tau_m\}$. Напомним, что

$$(x_1, x_2, \ldots, x_l) \in r \Leftrightarrow$$

$$(\forall \tau \in S)[(x_1, x_2) \in \tau \vee (x_1, x_3) \in \tau \vee \ldots \vee (x_{l-1}, x_1) \in \tau\} \quad (1).$$

Возьмем по одному элементу из каждого класса эквивалентности τ_1. Пусть это будут элементы $a_1, a_2, \ldots, a_l$. Обозначим через b_i элемент $k - 1 - a_i$ $(1 \le i \le l)$. Тогда поскольку функция $x \div y$ сохраняет r, из условий

$$(a_1, k\text{-}1, k\text{-}1 \ldots, k\text{-}1) \in r,$$
$$(0, b_2, 0, \ldots, 0) \in r$$

следует, что $(a_1, a_2, 0, \ldots, 0) \in r$. Далее из условий

$$(a_1, a_2, k\text{-}1, \ldots, k\text{-}1) \in r,$$
$$(0, 0, b_3, \ldots, 0) \in r$$

получаем, что $(a_1, a_2, a_3, k\text{-}1, \ldots, k\text{-}1) \in r$ и т. д. В итоге будем иметь, что $(a_1, a_2, \ldots, a_l) \in r$. Это противоречит тому, что элементы $a_1, a_2, \ldots, a_l$ лежат в разных классах эквивалентности τ_1 и условию (1) при $\tau = \tau_1$.

Мы доказали, что класс $G = \{\neg x, x \div y\}$ является полным.

Обратим внимание на то, что циклическое отрицание $\neg x$ использовалось только в случае 5, и фактически использовалась только взаимная однозначность этой функции. Следовательно, класс функций

$$\{f(x),\ x \div y\}$$

будет полным для любой перестановки f множества $\boldsymbol{B}_k$. □

Пример 3. Докажем, что класс

$$H = \{1,\ x + y,\ x \cdot y\}$$

является полным тогда и только тогда, когда k – простое число.

Предположим вначале, что число k не является простым, т. е. $k = pm$, где p – простое число и $m > 1$. На множестве $\boldsymbol{B}_k$ рассмотрим следующее бинарное отношение r:

$$(x, y) \in r \Leftrightarrow x - y \text{ нацело делится на } p.$$

Легко проверить, что r – собственное отношение эквивалентности и что все функции из H сохраняют это отношение. Следовательно, класс H не является полным.

Пусть теперь k – простое число. Возьмем отношение r, заданное на $\boldsymbol{B}_k$. Предположим, что функции класса H сохраняют r. Рассмотрим шесть случаев, соответствующих шести типам отношений из формулировки теоремы 14.

Случай 1: r – *ограниченный частичный порядок*. Наименьший элемент обозначим буквой a, наибольший – буквой b. Тогда

$$(a, b) \in r \text{ и}$$
$$(-a, -a) \in r.$$

Так как сложение сохраняет отношение , получаем, что

$$(0, b - a) \in r.$$

Складывая вектор $(0, b - a)$ сам с собой необходимое число раз получим, что $(0, a) \in r$, поскольку $b - a \neq 0$ и k – простое число. Отсюда следует, что $a = 0$, так как a – наименьший элемент. Аналогичным образом получаем равенство $b = 0$ и вместе с ним противоречие $a = 0 = b$.

Случай 2: r – *граф нетоджественной перестановки множества* $\boldsymbol{B}_k$. Тогда r не сохраняется константой 1, принадлежащей H. Действительно, в противном случае из условия $(a, b) \in r$ следует, что $(1, 1) \in r$. Перестановка в случае 2 является произведением простых циклов, поэтому ее граф петель не имеет.

Случай 3: r – *четырехместное отношение* $x_1 + x_2 = x_3 + x_4$, *заданное на абелевой группе* $(\boldsymbol{B}_k, \oplus)$. Тогда

$$(0, 1, 0, 1) \in r \text{ и}$$
$$(1, 0, 0, 1) \in r,$$

поскольку группа абелева. Так как умножение сохраняет отношение r, перемножив эти два четырехмерных вектора по компонентно, получим, что

$$(0, 0, 0, 1) \in r.$$

Последнее условие означает, что $0 \oplus 0 = 0 \oplus 1$. Это равенство дает противоречие $0 = 1$, поскольку в группе $(\boldsymbol{B}_k, \oplus)$ можно сокращать.

Случай 4: *r – собственное отношение эквивалентности, заданное на $\boldsymbol{B}_k$*. Существуют два различных эквивалентных элемента. Обозначим их через a и b. Тогда

$$(a, b) \in r \text{ и}$$
$$(-a, -a) \in r.$$

Складывая эти две пары, получим, что $(0, b - a) \in r$, поскольку сложение сохраняет отношение r. Возьмем произвольный элемент x из $\boldsymbol{B}_k$. Складывая пару $(0, b - a)$ саму с собой необходимое число раз, получим, что $(0, x) \in r$. Это означает, что отношение r имеет только один класс эквивалентности. Получили противоречие с тем, что r – собственное отношение эквивалентности.

Случай 5: *r есть l-местное центральное отношение*. Легко видеть, что замыкание класса H содержит все константы. Следовательно, одноместных центральных отношений, которые сохраняются функциями из H не существует, поскольку в этом случае $Z(r) = \boldsymbol{B}_k$. Пусть r является l-местным центральным отношением при $1 < l < k$. Возьмем элемент a из $Z(r)$. Пусть b – произвольный элемент из $\boldsymbol{B}_k$. Докажем, что $b \in Z(r)$. Пусть x_1, x_2, …, x_{l-1} – произвольный набор элементов из $\boldsymbol{B}_k$. Так как k – простое число, существует число m, такое что $ma = b$. В группе $(\boldsymbol{B}_k, +)$ уравнение $my = x$ всегда имеет решение (относительно y). Пусть $y_1, y_2, \ldots, y_{l-1}$ – элементы из $\boldsymbol{B}_k$, такие что $my_i = x_i$ для $i = 1, \ldots, l\text{-}1$. Тогда

$$(y_1, y_2, \ldots, y_{l-1}, a) \in r,$$

поскольку $a \in Z(r)$. Сложим вектор $(y_1, y_2, \ldots, y_{l-1}, a)$ сам с собой раз, получим, что

$$(x_1, x_2, \ldots, x_{l-1}, b) \in r.$$

Следовательно, $b \in Z(r)$. Получили равенство $Z(r) = \boldsymbol{B}_k$, которое противоречит определению центрального отношения.

Случай 6: *r есть l-регулярное отношение и $l > 2$*. Пусть r определяется l-регулярным семейством эквивалентностей $S = \{\tau_1, \tau_2, \ldots, \tau_m\}$. Напомним, что

$$(x_1, x_2, \ldots, x_l) \in r \Leftrightarrow$$
$$(\forall \tau \in S)[(x_1, x_2) \in \tau \vee (x_1, x_3) \in \tau \vee \ldots. \vee (x_{l-1}, x_1) \in \tau\} \quad (1).$$

Возьмем по одному элементу из каждого класса эквивалентности τ_1. Пусть это будут элементы $a_1, a_2, \ldots, a_l$. Тогда

$$(a_1, 0, \ldots, 0) \in r,$$
$$(0, a_2, \ldots, 0) \in r,$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$(0, 0, \ldots, a_l) \in r,$$

поскольку r вполне рефлексивно и $l > 2$. Складывая эти l-мерные векторы, получим, что

$$(a_1, a_2, \ldots, a_l) \in r.$$

Получили противоречие с условием (1) при $\tau = \tau_1$.

Мы доказали, что для каждого отношения любого из шести типов из формулировки теоремы 14 существует функция из класса $H = \{1, x + y, x \cdot y\}$, которая это отношение не сохраняет. Следовательно, H – полный класс. □

Задачи

1. Показать, что переменная x_1 булевой функции f является фиктивной:

1) $f = (x_2 \to \cdot x_1) \cdot (x_2 \downarrow x_2)$,
2) $f = (x_1 \leftrightarrow x_2) \vee (x_1 \mid x_2)$,
3) $f = ((x_1 + x_2) \to x_3) \cdot \neg (x_3 \to x_2)$.

2. Доказать, что $[K \cap L] \subseteq [K] \cap [L]$, $[K] \cup [L] \subseteq [K \cup L]$.

3. Показать, что функция f принадлежит замыканию класса C булевых функций:

1) $f = x$, $C = \{x + y\}$,
2) $f = x + y + z$, $C = \{x \leftrightarrow y\}$,
3) $f = x \vee y$, $C = \{\neg x \vee \neg y\}$,
4) $f = x + y + z$, $C = \{\neg x, xy \vee xz \vee yz\}$,
5) $f = x + y$, $C = \{x \cdot \neg y, x \vee \neg y\}$.

4. Содержит ли замыкание класса $\{\iota(x), x \cdot y, x \vee y\}$ функции $\theta(x)$, $\neg x$?

5. Выяснить, является ли функция f двойственной к функции g

а) $f = x+y$, $g = x \leftrightarrow y$, б) $f = x \to y$, $g = y \to x$,
в) $f = x \to y$, $g = \neg x \cdot y$, г) $f = (x+y) \cdot z$, $g = (x+y) \cdot (z+1)$,
д) $f = (x \cdot y) \vee (x \cdot z) \vee (y \cdot z)$, $g = x \cdot y + x \cdot z + y \cdot z$,
е) $f = x \leftrightarrow y$, $g = (\neg x \cdot y) \vee (x \cdot \neg y)$.

6. Выяснить, будут ли следующие функции самодвойственными: $\nu(x)$, xy, $(x \cdot y) \vee (x \cdot z) \vee (y \cdot z)$, $(x \vee y) \cdot (x \vee z) \cdot (y \vee z)$, $x+y+z$.

7. Показать, что не существует самодвойственной функции, существенно зависящей от двух переменных.

8. Доказать монотонность функций $x\cdot(y\vee z)$, $x\vee(y\cdot z)$, $max(x, y, z)$, $min(x, y, z)$.

9. Выяснить, будут ли следующие функции монотонны: $yx+y$, $xy+y+x$, $x\leftrightarrow y$, $x \to (y\to x)$, $(x\cdot y)\vee(x\cdot z)\vee(y\cdot z)$, $x\to(x\to y)$.

10. Доказать, что функция, двойственная монотонной, сама монотонна.

11. Система функций C называется *базисом* замкнутого класса K, если замыкание системы C совпадает с K, но замыкание любой собственной подсистемы системы C уже не совпадает с K. Доказать, что система $\{\theta(x), \iota(x), x\cdot y, x\vee y\}$ образует базис в классе всех монотонных функций.

12. Выяснить, будут ли следующие функции линейными: $x\downarrow y$, $\neg(x \leftrightarrow \neg y)$, $(x\leftrightarrow y) \leftrightarrow z$, $x \to (y\to x)$.

13. Сведением к известным полным классам доказать полноту классов функций двузначной логики:

а) $\{x\to y, \neg x\}$, б) $\{x\to y, \theta(x)\}$,
в) $\{x\downarrow y\}$, г) $\{x\to y, x+y\}$,
д) $\{x\vee y, x+y, \iota(x)\}$, е) $\{x\leftrightarrow y, \theta(x), x\vee y\}$.

14. Используя теорему Поста, доказать полноту классов функций двузначной логики:

а) $\{x\vee y, \neg x\}$, б) $\{x\to y, \neg x\}$,
в) $\{x\cdot y+x, x+y, \iota(x))\}$, г) $\{x+y, x\leftrightarrow(y\cdot z)\}$,
д) $\{x\cdot y, x+y, x\leftrightarrow(x\cdot y)\}$, е) $\{x\leftrightarrow y, \theta(x), x\vee y\}$.

15. Выяснить, будут ли полными в $\boldsymbol{B}_2$ следующие классы функций:

а) $\{xy, x+y+1\}$,
б) $\{\theta(x), \iota(x), (x\cdot y)\vee z\}$,
в) $\{\theta(x), \iota(x), x\leftrightarrow y\}$,
г) $\{x\cdot y+x, x\leftrightarrow y, \theta(x)\}$,
д) $\{x\vee y, x\cdot y+x\cdot z\}$,
е) $\{\neg x, (x\cdot y)\vee(x\cdot z)\vee(y\cdot z)\}$,
ж) $\{\iota(x), \neg x, x+y+max(x, y, z)\}$,
з) $\{x+y, x\vee y\vee\nu(z)\}$.

16. Для функции

$$f = x_1x_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3$$

найти сокращенную ДНФ тремя способами:

1) используя преобразования неполного склеивания и поглощения;

2) по карте Карно;

3) по геометрическому изображению.

17. Решить задачу 16 для функции

$$f = x_1x_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1x_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3.$$

18. Решить задачу 16 для функции

$$f = x_1x_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1x_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3.$$

19. Для функции, заданной вектором значений

$$f = (1110\ 0111\ 0100\ 1101)$$

найти сокращенную ДНФ двумя способами:

1) по карте Карно;

2) по геометрическому изображению.

20. Выяснить, будет ли ДНФ тупиковой, минимальной:

а) $f_1 = x_1x_2 \vee x_2$, б) $f_2 = x_1x_2 \vee \bar{x}_2$,

в) $f_3 = \bar{x}_1x_2 \vee x_1\bar{x}_2$, г) $f_3 = x_1x_2 \vee x_1\bar{x}_2 \vee \bar{x}_1\bar{x}_2$.

21. Для функции, заданной картой Карно

x_1x_2 \ x_3x_4	00	01	11	10
00			1	
01	1		1	1
11	1		1	1
10	1	1	1	

найти тупиковые и минимальные ДНФ.

22. Для функции, заданной картой Карно

x_1x_2 \ x_3x_4	00	01	11	10
00	1	1		1
01		1	1	
11	1	1	1	1
10	1	1		

найти тупиковые и минимальные ДНФ.

23. Для функции, заданной картой Карно

x_1x_2 \ x_3x_4	00	01	11	10
00			1	

01	1	1	1	1
11	1	1		1
10			1	

найти тупиковые и минимальные ДНФ.

24. Сколько тупиковых и минимальных ДНФ будет иметь функция, заданная следующей картой Карно

x_1x_2 \ x_3x_4	00	01	11	10	
00		1	1	1	
01		1		1	
11	1	1	1	1	
10			1	1	?

25. Сколько тупиковых кратчайших и минимальных ДНФ будет иметь функция, заданная следующей картой Карно

x_1x_2 \ x_3x_4	00	01	11	10	
00	1	1	1		
01	1		1	1	
11		1	1	1	
10		1	1		?

26. Сведением к известным полным классам функций, доказать полноту в $\boldsymbol{P}_k$ следующих классов:

1) $C_1 = \{J_0(x), J_1(x), \ldots, J_{k-1}(x), x^2, x \div y\}$,
2) $C_2 = \{k - 1, x + y, x \div y\}$,
3) $C_3 = \{\sim x, x + 2, x \div y\}$,
4) $C_4 = \{-x, 1 - x^2, x \div y\}$,
5) $C_5 = \{j_0(x), x + y\}$
6) $C_6 = \{J_0(x)\ x + y, x \cdot y^2\}$
7) $C_7 = \{1, x^2 - y, min(x, y)\}$
8) $C_8 = \{\neg x, j_0(x), min(x, y)\}$.

27. Доказать полноту классов функций из задачи 26, используя критерий Розенберга.

28. Показать, что следующие классы неполны в $\boldsymbol{P}_k$ при $k > 2$, подобрав одноместное отношение или отношение эквивалентности, которое сохраняется функциями класса:

1) $C_1 = \{\sim x, min(x, y), x \cdot y^2\}$,
2) $C_2 = \{2, max(x, y), x \div y\}$,
3) $C_3 = \{0, 1, \sim x, min(x, y)\}$.

29. Для заданных k исследовать на полноту в $\boldsymbol{P}_k$ следующие классы функций:

1) $k = 3$, $C_1 = \{1, J_0(x), J_2(x), min(x, y), max(x, y)\}$,
2) $k = 3$, $C_2 = \{1, 2, J_2(x), min(x, y), max(x, y)\}$,
3) $k = 4$, $C_3 = \{1, 2, J_0(x), J_1(x), min(x, y), max(x, y)\}$,
4) $k = 4$, $C_4 = \{1, 2, J_0(x), J_3(x), min(x, y), max(x, y)\}$.

30. С помощью критерия Розенберга доказать, число предполных классов трехзначной логики равно 18.

Ответы, указания и решения

2. Приведем доказательство включения $[K \cap L] \subseteq [K] \cap [L]$. Пусть $\{M_i \mid i \in I\}$ – семейство замкнутых классов, содержащих $K \cap L$, $\{K_j \mid j \in J\}$ – семейство замкнутых классов, содержащих K. Тогда, поскольку $K \cap L \subseteq K$, то все K_j содержат $K \cap L$, т.е.

$$\{K_j \mid j \in J\} \subseteq \{M_i \mid i \in I\}.$$

Отсюда следует, что $\cap\{K_j \mid j \in J\} \supseteq \cap\{M_i \mid i \in I\}$. Это означает, что $[K \cap L] \subseteq [K]$. Аналогично доказывается включение $[K \cap L] \subseteq [L]$. Следовательно, $[K \cap L] \subseteq [K] \cap [L]$.

4. Приведем решение задачи. Пусть $K = \{\iota(x), xy, x \vee y\}$. Функции $\theta(x)$ и $\nu(x)$ не принадлежат $[K]$, поскольку все функции из K (а следовательно, и из $[K]$) сохраняют единицу, а функции $\theta(x)$ и $\nu(x)$ единицу не сохраняют.

5. Указание. Построить таблицу, задающую двойственную к f функцию и функцию g. В случаях а), в), д) и е) ответ положительный, в остальных случаях – отрицательный.

6. Указание. Построить таблицу, задающую функцию f и двойственную к ней. Функции $\neg x$, $(xy) \vee (xz) \vee (yz)$ и $(x \vee y) \cdot (x \vee z) \cdot (y \vee z)$ самодвойственны (последние две просто равны), остальные нет.

7. Указание. Рассмотреть таблицы, задающие функции, существенно зависящие от двух переменных.

8. Указание. Проставить значения функций в вершинах диаграммы частично упорядоченного множества B^3.

9. Монотонными являются функции $xy+y+x$, $x \to (y \to x)$, $(xy) \vee (xz) \vee (yz)$.

11. Приведем решение задачи. Класс функций $\{\theta(x), \iota(x), xy, x \vee y\}$ обозначим буквой C. Пусть $f(x_1, \ldots, x_n)$ – мо-

нотонная функция. Если $f(x_1, \ldots, x_n)$ тождественно равна 0, то $f(x_1, \ldots, x_n) = \theta(x_1)\cdot\ldots\cdot\theta(x_n)$. Если же эта функция тождественно равна 1, то $f(x_1, \ldots, x_n) = \iota(x_1)\cdot\ldots\cdot\iota(x_n)$. Следовательно, в этих двух случаях функция f принадлежит $[C]$. Будем считать, что $f(x_1, \ldots, x_n)$ принимает оба значения. Рассмотрим диаграмму B^n частично упорядоченного множества B^n и в вершинах диаграммы проставим значения функции $f(x_1, \ldots, x_n)$. Поскольку функция f монотонна, то из того, что в некоторой вершине функция f принимает значение 1, следует, что и всюду выше она принимает то же значение. (В частности, в нижней вершине диаграммы f принимает значение 0, в верхней – значение 1.) Отсюда следует, что существуют вершины диаграммы a^1, a^2, …, a^k такие, что $f(b) = 1$ тогда и только тогда, когда $a^i \le b$ для некоторого $i \in \{1, \ldots, k\}$.

Другими словами, $\{a \in B^n \mid f(a) = 1\} = D_1 \cup D_2 \cup \ldots \cup D_k$, где $D_i = \{a \in B^n \mid a^i \le a\}$.

Рассмотрим вектор a^1. Пусть у этого вектора единицы стоят на местах i_1, i_2, …, i_l. Для упрощения обозначений предположим, что $i_1 = 1$, …, $i_l = l$. Пусть $g_1(x_1, \ldots, x_n) = x_1\cdot x_2\cdot\ldots\cdot x_l$. Легко видеть, что $D_1 = \{a \in B^n \mid g_1(a) = 1\}$. Аналогичным образом получаем существование функций $g_2(x_1, \ldots, x_n)$, …, $g_k(x_1, \ldots, x_n)$, таких, что $D_2 = \{a \in B^n \mid g_2(a) = 1\}$, …, $D_k = \{a \in B^n \mid g_k(a) = 1\}$. Тогда поскольку множество $\{a \in B^n \mid f(a) = 1\}$ есть объединение $D_1 \cup \ldots \cup D_k$, получаем равенство:

$$f(x_1, \ldots, x_n) = g_1(x_1, \ldots, x_n) \vee \ldots \vee g_k(x_1, \ldots, x_n).$$

Мы доказали, что замыкание класса C совпадает с M, т.е. с классом всех монотонных функций. Покажем, что C – базис M. Если из C удалить $\theta(x)$ то из оставшихся функций нельзя выразить $\theta(x)$, поскольку они сохраняют 1. Аналогично доказывается, что нельзя удалить $\iota(x)$.

Докажем, что замыкание класса $C' = C\setminus\{x\cdot y\}$ не содержит $x\cdot y$. Предположим противное: пусть $x\cdot y$ выражается (с помощью суперпозиции и переименования аргументов) через функции класса C' и пусть $g(x, y)$ – самое короткое выражение для этой функции. (В частности это означает, что $x\cdot y = g(x, y)$). Тогда $g(x, y) \ne \theta(h(x, y))$ для некоторого $h(x, y)$, поскольку в случае равенства $g(x, y) = \theta(h(x, y))$ произведение xy тождественно равно 0. Аналогично заключаем, что $g(x, y) \ne \iota(h(x, y))$. Следовательно, $x\cdot y = g(x, y) = g_1(x, y)$

$\vee\, g_2(x, y)$. Отсюда следует, что $g_1(0, 0) = g_1(1, 0) = g_1(0, 1) = 0$ и $g_2(0, 0) = g_2(1, 0) = g_2(0, 1) = 0$, поскольку $0\cdot 0 = 1\cdot 0 = 0\cdot 1 = 0$. Далее, так как $1 = 1\cdot 1 = g_1(1, 1)\vee g_2(1, 1)$, имеем $g_1(1, 1) = 1$ или $g_2(1, 1) = 1$. Это означает, что в качестве $g(x, y)$ можно взять $g_1(x, y)$ или $g_2(x, y)$, что противоречит выбору функции $g(x, y)$. Следовательно, $x\cdot y$ не принадлежит замыканию класса $C\setminus\{x\cdot y\}$. Аналогично доказывается, что $x\vee y \notin [C\setminus\{x\vee y\}]$.

Мы доказали, что $\{\theta(x), \iota(x), x\cdot y, x\vee y\}$ – базис класса монотонных функций.

12. Указание. См. решение аналогичной задачи в начале §4. Линейными являются все функции, кроме первой.

14. Приведем решение задачи 14 г. Пусть $K = \{x+y, x\leftrightarrow(yz)\}$. В силу теоремы Поста, надо в классе K найти функцию, не сохраняющую 0, функцию, не сохраняющую 1, несамодвойственную, немонотонную, нелинейную функции. Функция $x\leftrightarrow(yz)$ не сохраняет 0, поскольку $0\leftrightarrow(0\cdot 0) = 1$. Функция $x+y$ не сохраняет 1. Она же несамодвойственна (см.таблицу 5.8) и, как уже отмечалось, немонотонна.

Таблица 5.8

x	y	$x+y$	$\neg x+\neg y$	$\neg(\neg x+\neg y)$
0	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	0	1

Докажем, что функция $x\leftrightarrow(y\cdot z)$ не линейна. Предположим, что $x\leftrightarrow(y\cdot z) = a_0+a_1\cdot x+a_2\cdot y+a_3\cdot z$. Составим таблицу, задающую функции из левой и правой части равенства (см. таблицу 5.9).

Полученное противоречие показывает, что $x\leftrightarrow(y\cdot z)$ – нелинейная функция. (Разумеется, таблицу можно было бы ограничить первыми четырьмя строками).

Таблица 5.9

x	y	z	$a_0+a_1\cdot x+a_2\cdot y+a_3\cdot z$	Следствие
0	0	0	a_0	$a_0 = 1$
0	0	1	$1+a_3$	$a_3 = 0$
0	1	0	$1+a_2$	$a_2 = 0$
0	1	1	1	Противоречие
1	0	0		
1	0	1		
1	1	0		

1	1	1		

15. Указание. Применить теорему Поста. В случаях г), ж), и з) классы будут полными. В остальных случаях – нет.

21. Тупиковыми являются следующие ДНФ:

$$f_1 = x_3x_4 \vee x_2\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3,$$

$$f_2 = x_3x_4 \vee x_2\bar{x}_4 \vee x_1\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2x_4,$$

f_1 – минимальная ДНФ.

23. Тупиковыми являются следующие ДНФ:

$$f_1 = x_2\bar{x}_3 \vee x_2x_3\bar{x}_4 \vee \bar{x}_2x_3x_4 \vee \bar{x}_1x_2x_3,$$

$$f_2 = x_2\bar{x}_3 \vee x_2x_3\bar{x}_4 \vee \bar{x}_2x_3x_4 \vee \bar{x}_1x_2,$$

f_2 – минимальная ДНФ.

24. Тупиковых ДНФ пять, минимальная – одна.

26. Указание. В задачах 26.1 – 26.4 воспользоваться тем, что $x \div (x \div y) = min(x, y)$.

27. Указание. В задачах 26.1 – 26.4 использовать пример 2 из § 13, в задачах 26.5 – 26.6 – пример 3 из того же параграфа.

Глава 6. Алгоритмы и машины Тьюринга

Основной вопрос, который будет обсуждаться в этой главе – это вопрос о том, что такое алгоритм. В связи с чем возникает такой вопрос? Рассмотрим следующую простую задачу. Как узнать, имеет ли уравнение с целыми коэффициентами и одним неизвестным целочисленное решение? Уравнение в общем виде можно записать так:

$$a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n = 0. \qquad (1)$$

Число a_n называется свободным членом уравнения. Нетрудно убедиться в том, что если $a_n \neq 0$ и уравнение (1) имеет целочисленное решение, то решение является делителем свободного члена. Действительно, если x_0 – решение уравнения (1), то выполняется числовое равенство

$$a_0x_0^n + a_1x_0^{n-1} + \ldots + a_{n-1}x_0 + a_n = 0,$$

которое можно записать так:

$$(a_0x_0^{n-1} + a_1x_0^{n-2} + \ldots + a_{n-1})x_0 = -a_n.$$

Мы видим, что левая часть последнего равенства нацело делится на x_0. Следовательно, и правая часть тоже нацело делится на x_0. Очевидно, что если x_0 является делителем числа $-a_n$, то x_0 является делителем числа a_n. Следовательно, x_0 – делитель свободного члена уравнения (1). Решить поставленную выше задачу можно следующим образом. Если $a_n = 0$, то уравнение имеет целое решение – число 0. Если $a_n \neq 0$, то надо выписать все делители свободного члена (не забывая и отрицательные) и подстановкой в уравнения убедиться, не являются ли они решениями.

Только что был изложен *алгоритм* решения сформулированной выше задачи. Следовательно, существует алгоритм, определяющий наличие целочисленного решения по произвольному уравнению с целыми коэффициентами и одним неизвестным. А существует ли такой алгоритм для уравнений с несколькими неизвестными? Это уже очень сложная математическая задача, известная в истории математики под названием *десятая проблема Гильберта*. Проблема была сформулирована в начале двадцатого века. Попытки ее решения и решения подобных про-

блем привели к осознанию того, что алгоритма может и не быть. А чтобы доказать, что не существует алгоритма решения той или иной задачи, необходимо точное определение понятия алгоритма. Оно будет дано в первом параграфе в виде так называемой машины Тьюринга.

В этой главе нам будет удобно считать, что множество натуральных чисел $\boldsymbol{N}$ содержит 0, т.е. $\boldsymbol{N} = \{0, 1, 2, \ldots, n, \ldots\}$.

§1. Понятие машины Тьюринга

Прежде, чем давать определение машины Тьюринга, проведем неформальное обсуждение понятия алгоритма. На вопрос о том, что такое алгоритм на самом приблизительном уровне можно ответить так: алгоритм – система правил, инструкций, точно описывающих процесс решения задачи. В этом “определении” довольно много неясного: неясно, что такое правило, инструкция, неясно, что значит точно описать процесс решения задачи.

Развернем понятие алгоритма более подробно, зафиксировав основные требования, предъявляемые к понятию алгоритма.

1. Отметим, прежде всего, что алгоритм применяется к некоторым исходным данным (или входным данным) и служит для получения определенных результатов (или выходных данных). В процессе работы алгоритм использует промежуточные результаты (или промежуточные данные). Алгоритм, следовательно, оперирует с *данными*. Поскольку целью параграфа, как объявлено выше, является формализация понятия алгоритма, то надо уточнить, что в этом случае понимается под данными. Чаще всего в качестве данных используются числа, массивы, списки. В то же время, изображение графа в виде совокупности точек и соединяющих эти точки непрерывных линий менее естественно воспринимается в качестве данных для алгоритма (несмотря на наличие довольно мощных графических средств в современных системах программирования).По вопросу о представлении данных мы примем следующее решение. Данные будут представляться в виде цепочек над некоторым алфавитом, который будем называть *внешним*.

2. Данные алгоритма для своего размещения требуют *памяти*. В этом случае мы поступим так: память мы будем представлять в виде ленты, разделенной на ячейки. Ячейка содержит один из символов внешнего алфавита. Будем считать, что лента бесконечна в обе стороны (или конечна, но расширяема дополнительными ячейками при необходимости).

3. Алгоритм состоит из конечного числа *элементарных шагов* (или *правил*, или *инструкций*). Элементарный шаг при формализации “превратится” в команду. Определение команды будет дано ниже.

4. Алгоритм работает *детерминировано, тактами*. Каждый такт состоит в выполнении элементарного шага (или команды). После выполнения очередного элементарного шага должно быть ясно, какой очередной шаг выполнять (причем только один) или остановиться.

5. К алгоритмам обычно предъявляют требование *результативности*, т.е. остановки после конечного числа тактов с указанием, что считать результатом алгоритма. Это означает, что если предлагается алгоритм вычисления функции $f(x)$, то желательно убедиться в сходимости этого алгоритма для любого x из области определения $f(x)$. Требование результативности имеет существенное отличие от требований, изложенных в предыдущих пунктах. Дело в том, что выполнимость этого требования в отличие от предыдущих нельзя проверить просмотром описания алгоритма. Метода проверки того, что алгоритм закончит работу на входных данных, как мы увидим ниже, не существует.

Исходя из вышесказанного, определим *машину Тьюринга* как физическое устройство, состоящее из

1) *управляющего устройства*, которое может находиться в одном из фиксированного множества *внутренних состояний* $Q = \{q_1, q_2, \ldots, q_n\}$;

2) *ленты*, разбитой на ячейки; в каждой ячейке может быть записан один символ *внешнего алфавита* $A = \{a_1, \ldots, a_m\}$; лента бесконечна в обе стороны; один из символов внешнего алфавита называется *пустым*; в каждый момент на ленте записано конечное число непустых символов;

3) *считывающей и записывающей головки*, которая обозревает одну ячейку ленты, записывает в нее новый символ из A (он может совпадать с прежним), сдвигается влево или вправо на одну ячейку или остается на месте.

Управляющее устройство при этом может изменить свое внутреннее состояние. Среди элементов множества Q выделены два: начальное и заключительное. В начальном состоянии машина находится перед началом работы, перейдя в заключительное состояние, машина останавливается.

Управляющее устройство будем обозначать прямоугольником, в котором записано внутреннее состояние, а обозреваемую ячейку стрелкой так, как это указано на рис. 6.1

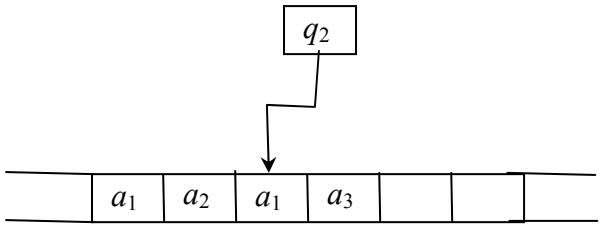


Рис. 6.1

Машина Тьюринга работает, детерминировано, тактами. Такт состоит в выполнении одной *команды* – выражения вида $q_i a_j \to q_k a_l d_m$, где $d_m \in \{L, R, H\}$. Эта команда содержит *условие* (левая часть команды до стрелки) и *действие* (правая часть команды после стрелки). Команда выполняется следующим образом. Если на начало очередного такта внутреннее состояние машины равно q_i, обозреваемый символ есть a_j, то после выполнения команды $q_i a_j \to q_k a_l d_m$, т.е. к концу такта, внутреннее состояние становится равным q_k, в обозреваемую ячейку записывается символ a_l, и головка сдвигается на одну ячейку вправо, если $d_m = R$, сдвигается влево, если $d_m = L$, остается на месте, если $d_m = H$.

Рисунки 6.2 и 6.3 иллюстрируют выполнение двух команд

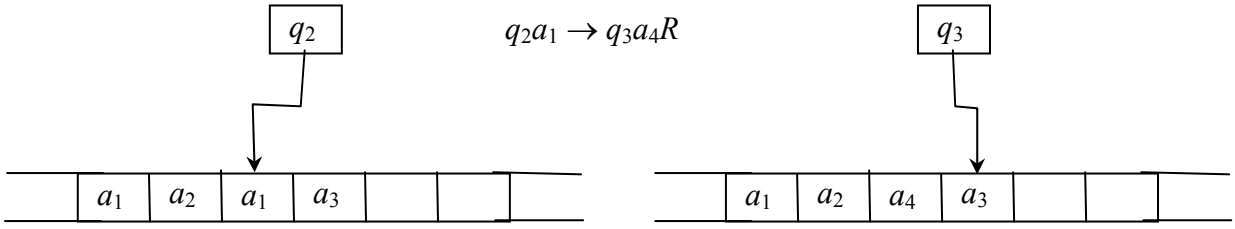


Рис. 6. 2

Работа машины Тьюринга определяется программой, а *программа* – это множество команд с различными условиями, т.е. различными левыми частями.

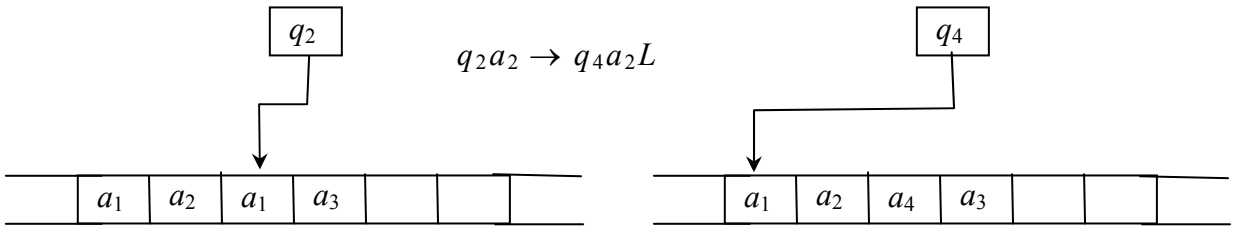


Рис.6. 3

Осталось уточнить, как машина начинает работу и как ее заканчивает. Будем считать, что машина начинает работу, обозревая самую левую непустую ячейку и находясь в начальном внутреннем состоянии. С определением того, как машина за-

канчивает работу, есть небольшая заминка. Дело в том, что если программу определить как любое множество команд (с различными условиями), то может возникнуть ситуация, когда внутреннее состояние есть q, обозревается ячейка, в которой записан символ a, но нет команды с условием qa. Подобную ситуацию будем называть *ситуацией неопределенности*. С определением условий остановки машины Тьюринга есть две возможности. Первая – считать, что программа есть любое множество команд, и что машина останавливается, приходя в заключительное состояние или попадая в ситуацию неопределенности. Вторая – считать, что для любого символа a внешнего алфавита и любого незаключительного состояния q есть команда с условием qa. Машина останавливается только тогда, когда приходит в заключительное состояние. Во втором случае, таким образом, вводится ограничение на определение программы. Мы не будем фиксировать, какую из двух возможностей выбираем в этой главе. Интуитивно ясно, что “вычислительные возможности” машин не зависят от выбора того или другого случая. (В конце параграфа 2 мы это докажем.)

Работа машины Тьюринга не всегда заканчивается. Рассмотрим соответствующие примеры. Предположим, что программа машины содержит следующие команды

$$q_1a\to q_2aR,\ q_2b\to q_1bL,\ q_3\lambda\to q_3\lambda R.$$

Если машина на начало очередного такта находится в состоянии q_1 обозревает ячейку, в которой записан символ a, а в следующей записан символ b, то произойдет зацикливание на участке ленты, содержащем эти символы, и машина будет работать бесконечно. Зацикливание работы машины может произойти еще в одном случае, если машина находится в состоянии q_3, обозревает пустую ячейку (с символом λ) и все следующие ячейки так же пусты, то машина также зацикливается, но уже на бесконечном участке ленты.

Итак, будем считать формальным аналогом интуитивного понятия алгоритм понятие машины Тьюринга. И словосочетание “алгоритм, решающий задачу” будем понимать как “машина Тьюринга, решающая задачу.” Конечно, надо дать формальное определение понятий “задача”, “машина, решающая задачу”. Один из подходов здесь состоит в том, что задачу сводят к вопросу о принадлежности элементов некоторым множествам, а решение задачи на машине Тьюринга – к вопросу о распознаваемости принадлежности элементов множеству. Подробнее об этом будет сказано в следующей главе.

В заключение параграфа обсудим одно понятие, которое часто будет использоваться в дальнейшем.

Пусть внешний алфавит машины Тьюринга состоит из символов $a_1, \ldots, a_m, \lambda$, где λ – пустой символ. Предположим, далее, что на начало некоторого такта возникла ситуация, изображенная на рис.5.4. Обратим внимание, что все пусты все ячейки, до той, в которой записан символ a_1, и после той, в которой записан символ a_3. Тогда цепочка $c = a_1 a_2 q_i a_4 \lambda a_3$ называется *конфигурацией* (соответствующей, изобра-женной на рис. 6.4 ситуации). Используя конфигурации, довольно естественно описывать действия команд.

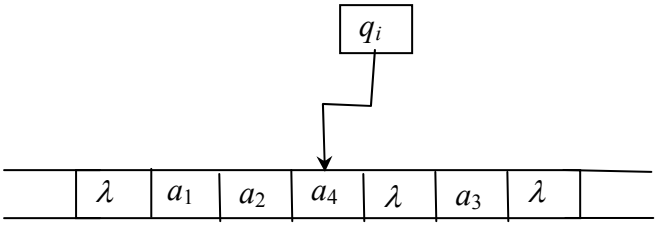


Рис. 6. 4

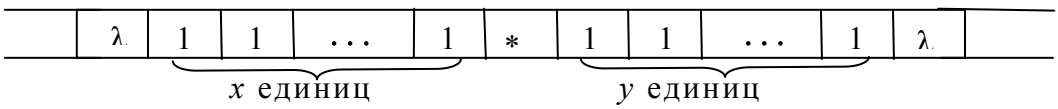
Так если в обсуждаемом случае машина содержит команду $q_i a_4 \to q_j a_3 L$, то ее выполнение означает переход от конфигурации c к конфигурации $c' = a_1 q_j a_2 a_3 \lambda a_3$.

§2. Примеры машин Тьюринга

Приведем примеры машин, которые выполняют некоторые содержательные действия, и обсудим способы задания программы машины Тьюринга.

Внутренние состояния машин будем, как правило, нумеровать натуральными числами. Начальное состояние в этом случае будет состоянием с наименьшим номером, заключительное – с наибольшим. *Примем еще и следующее соглашение: если головка при выполнении команды не сдвигается, то символ H в этой команде не писать.*

Пример 1. Рассмотрим вначале машину Тьюринга, которая складывает два ненулевых натуральных числа, записанных в единичном коде. Эта машина, имея в начальный момент ленту



перерабатывает ее в ленту

и останавливается. Внешний алфавит машины, таким образом, состоит из символов 1, *, λ, причем λ – пустой символ.

Как будет работать машина? Она, двигаясь вправо, проходит аргумент x, символ $*$ заменяет на 1, далее проходит аргумент y до первой пустой ячейки и, возвратившись на ячейку назад “стирает” у аргумента y одну единицу.

Легко понять, что это можно осуществить следующей программой:

$$q_1 1 \to q_1 1R, \quad q_1 * \to q_1 1R,$$
$$q_1 \lambda \to q_2 \lambda L, \quad q_2 1 \to q_3 \lambda. \qquad \square$$

На машине Тьюринга из примера 1 обсудим способы задания машин. В этом примере программа машины задана списком команд. Однако иногда удобнее задавать ее таблицей. Программа, написанная в предыдущем абзаце, может быть задана таблицей 6.1.

Таблица 6.1

	1	*	λ
q_1	$q_1 1R$	$q_1 1R$	$q_2 \lambda L$
q_2	$q_3 \lambda$		
q_3			

Таблица читается следующим образом. Если взять строку, обозначенную символом q, и столбец, обозначенный символом a, то в соответствующей клетке записана команда с условием qa. Например, в клетке, расположенной во второй строке и первом столбце таблицы, записано $q_3\lambda$. Это означает, что машина Тьюринга содержит команду $q_2 1 \to q_3 \lambda$. А команды с условием $q_2 *$ машина не содержит.

Для упрощения вида таблицы, описывающей программу машины, условимся о следующем. Поскольку перейдя в заключительное состояние, машина никаких действий не совершает, то *не будем заключительному состоянию отводить строку таблицы*. Далее, *если команда не изменяет внутреннее состояние или обозреваемый символ, то их в клетке писать не будем*. С учетом этих соглашений таблица 6.1 «превратится» в таблицу 6.2.

Таблица 6.2

	1	*	λ
q_1	R	R	$q_2 \lambda L$
q_2	$q_3 \lambda$		

Рассмотрим на этом же примере еще один способ описания программы машины – описание с помощью диаграмм. *Диаграмма* представляет собой ориентированный граф с помеченными вершинами и дугами и имеет вид, изображенный на рис. 6.5.

Вершины диаграммы представляют собой внутренние состояния. Если машина имеет команду $q_i a_j \to q_k a_l d_m$, то от вершины q_i к вершине q_k проводится дуга, которая помечается выражением $a_j \to a_l d_m$, если $a_j \neq a_l$, и выражением $a_j \to d_m$, если $a_j = a_l$. Условимся несколько команд вида $q_i a_1 \to q_k a_1' d$, …, $q_i a_p \to q_k a_p' d$ изображать одной дугой с пометкой a_1, …,$a_p \to a_1'$, …, $a_p' d$ (или a_1, …, $a_p \to d$, когда $a_1 = a_1'$, …, $a_p = a_p'$). Начальное состояние на диаграмме обозначается “двойным” кружком, а заключительное – “жирным” кружком.

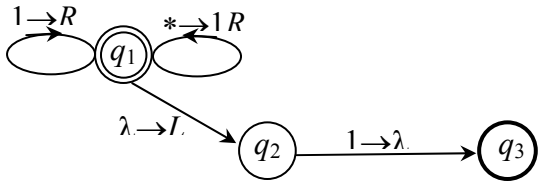


Рис. 6 . 5

Пример 2. Напишем программу еще одной простой машины Тьюринга – машины, осуществляющей проверку на четность. Перед началом работы на входной ленте записано x единиц (остальные ячейки пусты). Если x четно, то «просмотрев» x, машина должна записать 1, если x нечетно, то записать 0. На стадии просмотра машина будет затирать аргумент, поочередно переходя из состояния q_0 (x четно) и q_1 (x нечетно). Заключительное состояние – q_2. Программа машины приведена в таблице 6. 2.

Таблица 6. 2

	1	λ	
q_0	$q_1 \lambda R$	$q_2 1$	
q_1	$q_0 \lambda R$	$q_2 0$	□

Пример 3. Составим диаграмму машины Тьюринга, которая обращает непустое входное слово алфавите $\{a, b\}$. Более точно, машина должна начальную конфигурацию $q_0 w$ переводить в конфигурацию $q w^r$, где q – заключительное состояние, w – непустое слово в алфавите $\{a, b\}$, w^r – слово w, записанное в обратном порядке. Назовем такую машину «конвертором». Основная идея работы машины состоит в том, что обращение слова w надо начинать с конца этого слова. Машина будет считывать очередную букву $x \in \{a, b\}$, ставить на ее месте маркер x_1, уходить вправо и в первую пустую ячейку записывать x. Диаграмма машины приведена на рис. 6. 6.

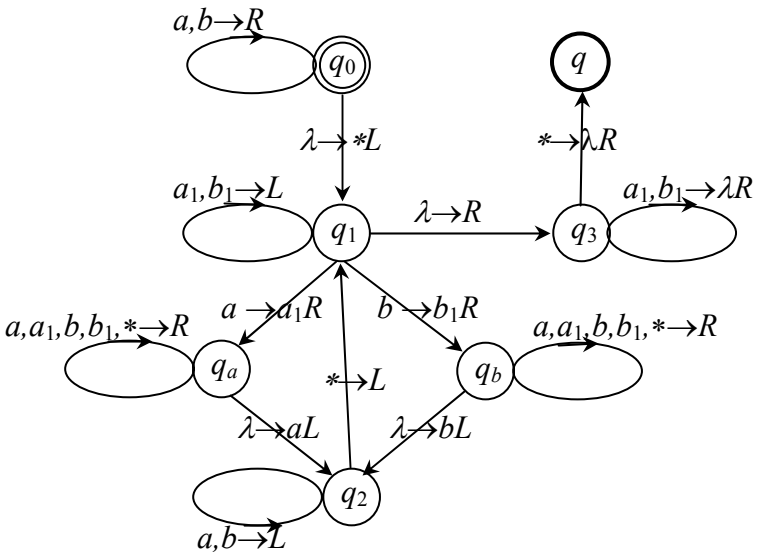


Рис. 6.6

Как мы видим на диаграмме, машина имеет 8 состояний. В состоянии q_1 машина находит первый с конца слова w еще «не обращенный» символ $x \in \{a, b\}$, заменяет его маркером x_1 и переходит в состояние q_a или q_b в зависимости от считанного символа. В состоянии q_x она находит первую пустую ячейку, записывает в нее символ x и в состоянии q_2 идет влево по входной ленте. При переходе через символ * машина приходит в состояние q_1 и цикл повторяется. Если в состоянии q_1 она читает символ *, то это означает, что обращение слова w завершено. Тогда машина в состоянии q_3 меняет x_1 на x и в состоянии q останавливается.

Пример 4. Рассмотрим машину Тьюринга, которая осуществляет перевод натуральных чисел из единичного кода в двоичный код. Эта машина должна, имея на входе x единиц, записать в начале ленты двоичное разложение числа x. Машину зададим диаграммой (см. рис. 6. 7).

Вначале машина работает, как машина проверки на четность из примера 2, только она вместо каждой нечетной единицы ставит символ *, а нечетные единицы пропускает (состояния q_0, q_1, и q_2). Если на первой пустой ячейке машина находится в состоянии q_1, то число просмотренных единиц нечетно, т. е. это число, деленное

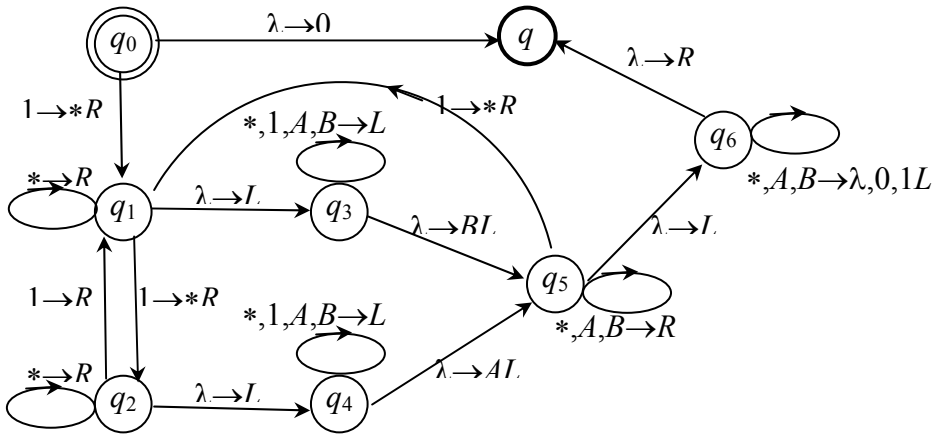


Рис. 6.7

на 2, дает в остатке 1. В этом случае машина слева от входного слова в первой пустой ячейке печатает символ B (состояния q_1, q_3, и q_5). Если же при движении вправо первую пустую ячейку машина обозревает в состоянии q_2, то число просмотренных единиц четно. Тогда машина слева от входного слова в первой пустой ячейке печатает символ A (состояния q_2, q_4, и q_5). Перевод заканчивается, когда все единицы входного слова заменены символом *. Это будет в случае, когда машина в состоянии q_5 будет обозревать пустую ячейку. Тогда машина движется влево, затирает символ *, вместо A печатает 0, вместо B – 1 и останавливается (состояния q_5, q_6, и q). □

Пример 5. В качестве пятого примера рассмотрим машину Тьюринга, которую назовем “переводчиком”. Внешний алфавит этой машины есть $\{a_1, a_2, b_1, b_2, *, \lambda\}$. Она, начиная работу на ленте

	a_{i1}	a_{i2}	. . .	a_{ik}	*	λ	

перерабатывает ее в ленту

	a_{i1}	a_{i2}	. . .	a_{ik}	*	b_{i1}	b_{i2}	. . .	b_{ik}	λ	

и останавливается. Машина должна осуществлять “пословный перевод” цепочки $a_{i1}a_{i2}...a_{ik}$ в цепочку $b_{i1}b_{i2}...b_{ik}$, сохраняя при этом «оригинальный текст».

Как должна работать такая машина? Она должна запомнить обозреваемый символ a_i, то место, где он считан, найти первую пустую ячейку и записать туда символ b_i. Скажем, символ a_1 запоминается внутренним состоянием q_1, символ a_2 – состоянием q_2, а обозреваемая ячейка символом внешнего алфавита λ. Пусть q_0 – начальное состояние.

Программа машина должна, таким образом, содержать команды

$$q_0a_1 \to q_1\lambda R \text{ и } q_0a_2 \to q_2\lambda R.$$

Далее, как сказано выше, машина в состоянии q_1 или q_2 должна дойти, двигаясь вправо, до первой пустой ячейки. Это могут "обеспечить" следующие команды

$$q_1x \to q_1xR \text{ и } q_2x \to q_2xR,$$

где $x \in \{a_1, a_2, b_1, b_2, *\}$. Если встретится первая пустая ячейка, то записать в нее символ b_1 или b_2. Добавим для выполнения этого в программу команды

$$q_1\lambda \to q_3b_1L \text{ и } q_2\lambda \to q_4b_2L \ .$$

Теперь машина должна вернуться, символ λ заменить на a_i. На этом один цикл работы машины закончится. Символы q_3 и q_4 "обеспечивают" это возвращение:

$$q_3x \to q_3xL \text{ и } q_4x \to q_4xL \ ,$$

где $x \in \{a_1, a_2, b_1, b_2, *\}$. И, наконец, добавим команды

$$q_3\lambda \to q_0a_1R,\ q_4\lambda \to q_0a_2R,\ q_0* \to q_5* \ .$$

Символ q_5 является заключительным состоянием машины.

Мы написали программу машины Тьюринга, названной переводчиком. Эта программа представлена таблицей 6.3.

Таблица 6.3

	a_1	a_2	b_1	b_2	$*$	λ
q_0	$q_1\lambda R$	$q_2\lambda R$			q_5	
q_1	R	R	R	R	R	q_3b_1L
q_2	R	R	R	R	R	q_4b_2L
q_3	L	L	L	L	L	q_0a_1R
q_4	L	L	L	L	L	q_0a_2R

.□

Параграф содержит еще два примера машин Тьюринга, которые на первый взгляд могут показаться «экзотическими». Эти примеры включены, чтобы облегчить понимание универсальной машины, о которой речь пойдет в § 5.

Пример 6. Через $h_x(w)$ обозначим начальный отрезок слова w до первого вхождения буквы x, а через $t_x(w)$ – конечный отрезок слова w, который начинается первым вхождением буквы x. Если w не содержит x, то $h_x(w) = w$, а $t_x(w) = \varepsilon$. Ясно, что $w = h_x(w)t_x(w)$. Пусть $A = \{a, b, c, m, \lambda\}$, λ – пустой символ, $\Sigma = \{a, b, c\}$, $w \in \Sigma^*$, $w \neq \varepsilon$. Слово w записано на ленте. Задача состоит в том, чтобы в слове w найти первое вхождение буквы c и сдвинуть $t_c(w)$ на одну ячейку вправо, а в освободившуюся ячейку (там, где было первое вхождение буквы c) записать m. Диаграмма машины M, решающей эту задачу, приведена на рис. 6.8.

Исполняя команды $q_0a \to q_0aR$ и $q_0b \to q_0bR$, машина находит первое вхождение буквы c. На месте этого вхождения машина ставит маркер m, переходит в состояние q_c, запоминая тем самым прочитанный символ c, и сдвигается вправо. Затем она, считывая очередной символ x, печатает на ленте предшествующий символ (это индекс внутреннего состояния), переходит во внутреннее состояние q_x и сдвигается вправо. Если M встречает пустой символ λ, то она печатает на ленте предшествующий символ и останавливается, переходя в состояние q. Если же слово w не содержит буквы c, то машина его не изменяет. Для ссылок машину M назовем “переносчиком”.

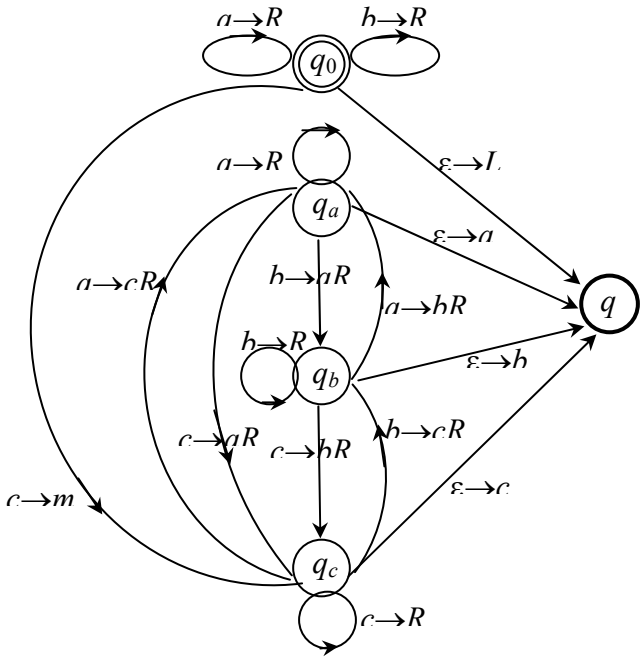


Рис. 6. 8

Пример 7. Пусть $A = \{a_1, a_2, b_1, b_2, a, m, 0, 1, *, \lambda\}$, λ – пустой символ, $\Sigma = \{a_1, a_2, a\}$, u, v – непустые цепочки над Σ. Необходимо построить машину M, которая определяет, равны ли отрезки $h_a(u)$ и $h_a(v)$ или различны. Более точно, машина должна начальную конфигурацию q_0mu*v переводить в конфигурацию $q0u*v$, если $h_a(u) \neq h_a(v)$, и в конфигурацию $q1u*v$, если $h_a(u) = h_a(v)$. Диаграмма машины приведена на рис. 6.9. и 6.10.

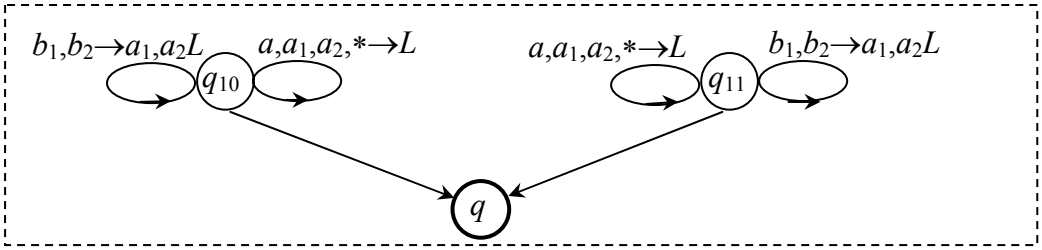


Рис. 6. 9

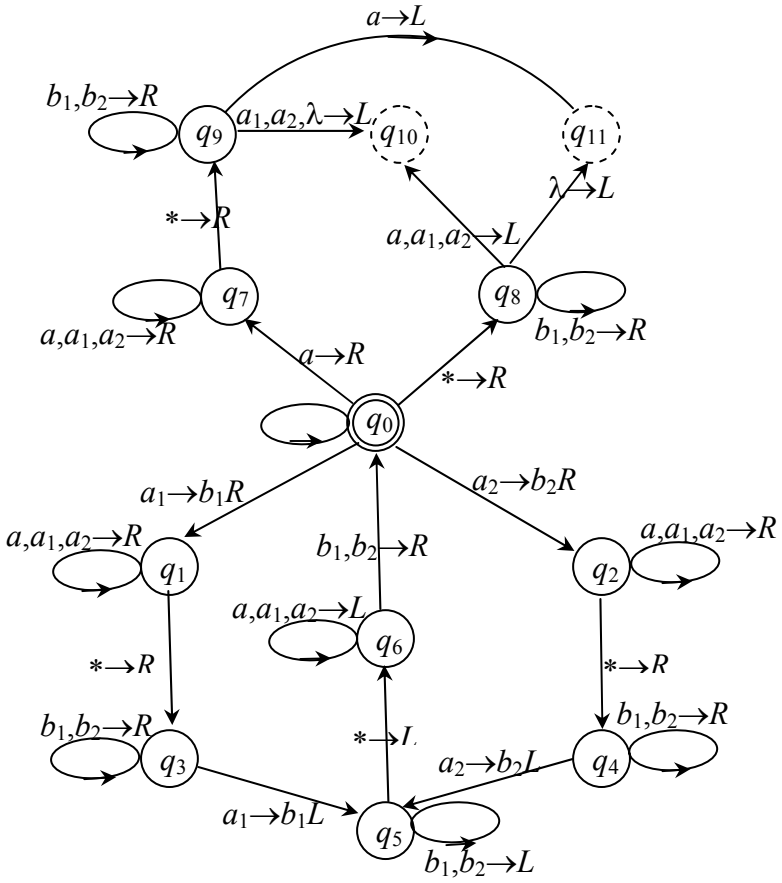


Рис. 6. 10

Машина начинает работу в состоянии q_0. Пусть первый символ, следующий за маркером m, есть a_1. Машина запоминает это двояким образом: a_1 заменяет на b_1 и переходит в состояние q_1. Затем M проходит оставшуюся часть слова u, оставляя ее без изменений, и при переходе через символ $*$ меняет состояние на q_3. Если в состоянии q_3 первый встретившийся ей символ из $\Sigma \cup \{\lambda\}$ будет a_1, то она заменяет его на b_1 и переходит в состояние q_5. В состоянии q_5 машина возвращается до символа $*$, меняет q_5 на q_6, находит первый символ из Σ в измененном слове u и переходит в состояние q_0. На этом завершается один цикл проверки. Если же в состоянии q_3 первый встретившийся машине символ есть a, a_2 или λ, то машина в состоянии q_{10} меняет символы b_1, b_2 на a_1, a_2 соответственно (т.е. восстанавливает цепочки u, v), маркер m заменяет на 0 и останавливается в состоянии q.

Аналогичные действия совершает машина, если она в состоянии q_0 считывает a_2 (см. состояния q_2, q_4 на диаграмме).

Рассмотрим случай, когда в состоянии q_0 машина считывает символ a. Тогда машина в состоянии q_7 проходит оставшуюся часть слова u до “звездочки”, переходит в состояние q_9. Если

встретившийся ей символ из $\Sigma \cup \{\lambda\}$ есть a, то это означает, что $h_a(u) = h_a(v)$, машина в состоянии q_{11} возвращается в начало цепочки, символы b_1, b_2 заменяет на a_1, a_2 соответственно, на месте маркера m печатает 1 и останавливается. Если же первый встретившийся символ есть a_1, a_2 или λ, то это означает, что $h_a(u) \neq h_a(v)$, машина переходит в состояние q_{10} и выполняет действия, которые были описаны выше.

Осталось рассмотреть случай, когда в состоянии q_0 машина обозревает символ $*$. Это означает, что слово u не содержит символа a, т.е. $h_a(u) = u$. Если измененное в процессе работы машины слово v содержит символы из Σ, то $h_a(u) \neq h_a(v)$, машина уходит в состояние q_{10} и на месте маркера m печатает 0. Если же измененное слово состоит только из символов b_1 и b_2, то $h_a(u) = h_a(v)$, машина переходит в состояние q_{11} и на месте маркера m печатает 1. □

Две машины с одним и тем же внешним алфавитом назовем *эквивалентными,* если они, начиная работу на одинаковых лентах, либо завершают ее также на одинаковых лентах, либо обе работают бесконечно.

Следующее утверждение часто оказывается полезным.

Теорема 6.1. Для любой машины Тьюринга T существует эквивалентная ей машина Тьюринга S, удовлетворяющая условиям:

1) если S имеет команды вида $qa \to q'a'$, то q' – заключительное состояние;

2) если S имеет команды вида $qa \to q'a'd$, где d – это L или R, то состояние q' не является заключительным.

Доказательство. Пусть внутренний алфавит машины T есть $Q = \{q_1, \ldots, q_k\}$, q_1 – начальное, q_k – заключительное состояния.

Напомним соглашение о том, что если в действии команды стоит символ d, то $d = L$ или $d = R$.

Машина S получается из машины T в результате преобразования команд. Если программа машины T содержит пару команд $qa \to q'a'$ и $q'a' \to q''a''d$, то заменим эти две команды на одну $qa \to q''a''d$. Будем проводить это преобразование до тех пор, пока не исчезнут указанные выше пары. Далее все команды вида $qa \to q'a'$, где $q' \neq q_k$, заменим на $qa \to q_k\, a'$. И наконец, в случае, когда программа содержит команды вида $C = qa \to q_k\, a'd$, где d – это L или R, к внутренним состояниям добавляем новые внутренние состояния q_c и каждую из этих команд заме-

няем на две команды $qa \to q_c a'd$ и $q_c a \to q_k a$. В результате выполнения этих преобразований и получится требуемая машина S. □

§3. Функции, вычислимые на машинах Тьюринга

В этом параграфе рассматриваются функции, заданные на множестве натуральных чисел $\boldsymbol{N}$, вообще говоря, многоместные и не обязательно всюду определенные. Натуральные числа, как правило, будем задавать в единичном коде. Так, запись 1^x обозначает последовательность, состоящую из x "штук" единиц.

Параграф посвящен изучению функций натурального аргумента, вычислимых на машинах Тьюринга. Пример такой функции уже был в §2, где было показано, что функция x_1+x_2 вычислима на машине Тьюринга. Однако часто необходимо, чтобы головка при остановке обозревала самую левую непустую ячейку. В этом случае говорят, что машина правильно вычисляет функцию.

Дадим точное определение.

Определение. Машина Тьюринга *M правильно вычисляет* функцию $f(x_1, \ldots, x_n)$, если M, начиная работу в конфигурации

$$q_0 1^{x1} * 1^{x2} * \ldots * 1^{xn},$$

заканчивает ее в конфигурации

$$q_s 1^{f(x1,\ldots,\,xn)},$$

если $f(x_1, \ldots, x_n)$ определена, и работает бесконечно, если $f(x_1, \ldots, x_n)$ не определена. (Здесь q_0 – начальное, q_s – заключительное состояния.)

Другими словами, машина T начиная работу в состоянии q_0, обозревая первую ячейку на ленте, изображенной на рис. 5.8, заканчивает ее в состоянии q_s, обозревая первую ячейку на ленте, изображенной на рис. 6.11, если $f(x_1, \ldots, x_n)$ определена, и работает бесконечно в противном случае.

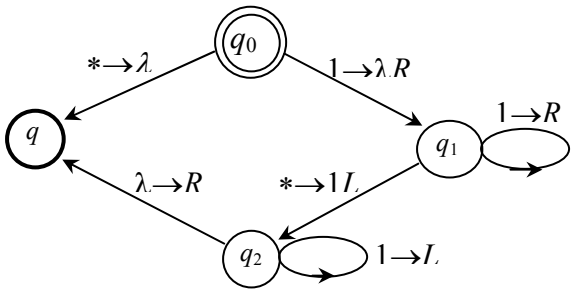


Рис. 6.11

Пример 1. Как уже было сказано, во втором параграфе был фактически приведен пример машины, вычисляющей функцию $f(x_1, x_2) = x_1+x_2$. Не было выполнено лишь два условия: первое – головка при остановке не пришла в крайнее левое положение; второе – рас-

сматривалось сложение ненулевых чисел. ”Подправим” эту машину (ее программа представлена в таблице 6.1) так, чтобы выполнялись отмеченные условия, получим машину Тьюринга, программа которой задается диаграммой на рис. 6.11 □

Пример 2. В качестве второго примера приведем диаграмму машины Тьюринга M, вычисляющей функцию $y = 2x$ (см. рис. 6.12). Эта машина в качестве своей части содержит машину «переводчик» из предыдущего параграфа. Пример демонстрирует такой прием в «технологии программирования на машинах Тьюринга», как уход на подпрограмму.

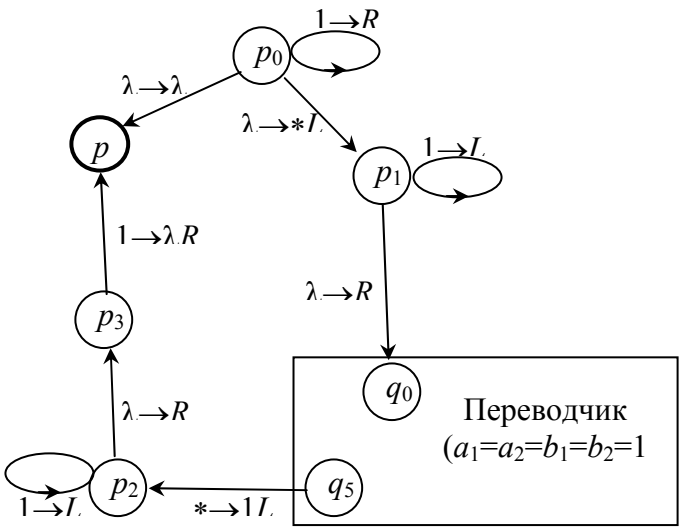


Рис. 6.12

Как видно на диаграмме, машина в случае, когда $x = 0$, просто переходит в заключительное состояние p и останавливается. Если же $x \neq 0$, то машина M в конце последовательности единиц ставит символ * и возвращается к первой единице. Затем она работает как переводчик при $a_1 = a_2 = b_1 = b_2 = 1$. Напомним, что переводчик останавливается в состоянии q_5 на символе *. Машина M на месте * ставит 1, находит самую левую единицу, ее затирает и останавливается. □

Чтобы привести более сложные примеры, нам понадобятся два утверждения – теорема 6.2 и теорема 6.3. Для простоты теоремы сформулированы для одноместных функций, хотя их утверждения справедливы и для многоместных функций.

Теорема 6.2. Пусть функции $y = f(x)$ и $y = g(x)$ правильно вычислимы. Тогда их суперпозиция $g(f(x))$ тоже правильно вычислима.

Доказательство. Пусть S – машина, вычисляющая функцию $y = f(x)$, а T – машина, вычисляющая функцию $y = g(x)$ и $U = \{u_1, \ldots, u_k\}$, $V = \{v_1, \ldots, v_l\}$ – множества внутренних состояний этих машин (u_1, v_1 – начальные состояния, u_k, v_l – заключительные). Можно считать, что $U \cap V = \varnothing$. Условимся, что машины S и T удовлетворяют утверждению теоремы 5.1. Рассмотрим машину M, множество внутренних состояний которой есть $\{u_1, \ldots, u_{k-1},$

$v_1, \ldots, v_l\}$, u_1 – начальное, v_l – заключительное состояния. Внешний алфавит машины M равен объединению внешних алфавитов машин S и T. Команды машины M получаются также объединением команд машин S и T со следующей "корректировкой": команда машины S вида $u_i a \to u_k a'$ заменяются на $u_i a \to v_1 a'$.

Если функция $y = f(x)$ определена, то машина S, начиная работу в конфигурации $u_1 1^x$, заканчивает ее в конфигурации $u_k 1^{f(x)}$. Предположим, что на последнем такте S применила команду $u_i a \to u_k a'$. Машина M на всех тактах, кроме последнего, работает так же, как и S, а на последнем такте применяет команду $u_i a \to v_1 a'$. Это означает, что машина M конфигурацию $u_1 1^x$ переводит в конфигурацию $v_1 1^{f(x)}$ и далее работает как машина T при вычислении $g(z)$, где $z = f(x)$. Если функция $g(z)$ определена, то машина T (и M) заканчивает работу в конфигурации $v_l 1^{g(f(x))}$.

Если же $f(x)$ или $g(z)$ при $z = f(x)$ не определены, то машина M работает бесконечно. □

Аналогично понятию правильной вычислимости функции натурального аргумента можно ввести понятие правильной вычислимости предиката, заданного на множестве натуральных чисел $\boldsymbol{N}$.

Определение. Предикат $P(x_1, x_2, \ldots, x_n)$, заданный на множестве $\boldsymbol{N}$, *правильно вычислим* на машине Тьюринга M, если M, начиная работу в конфигурации

$$q_0 1^{x1} * 1^{x2} * \ldots * 1^{xn},$$

заканчивает в конфигурации $q_s 1$, если $P(x_1, x_2, \ldots x_n)$ истинно, и в конфигурации $q_s 0$, если $P(x_1, x_2, \ldots, x_n)$ ложно. (Здесь q_0 – начальное, q_s – заключительное состояния.)

Сделаем замечание, касающееся понятия правильной вычислимости предиката. Правильная вычислимость предиката $P(x_1, x_2, \ldots, x_n)$ равносильна правильной вычислимости характеристической функции этого предиката, т.е. функции

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, \text{ если } P(x_1, x_2, \ldots, x_n) \text{ истинно,} \\ 0 \text{ в противном случае.} \end{cases}$$

Несмотря на это, будем пользоваться сформулированным выше определением.

Приведем два примера правильной вычислимости предиката.

Пример 3. Предикат $P(x)$ – «x есть четное число» правильно вычислим. Рассмотрим машину, которая читает входное слово, переходя поочередно в состояния q_0 и q_1 и затирая единицы. Если первую (справа от x) пустую ячейку машина будет обозревать в состоянии q_0, то x – четное число; если в состоянии q_1, то x – нечетное число. Диаграмма машины приведена на рис. 6.13. □

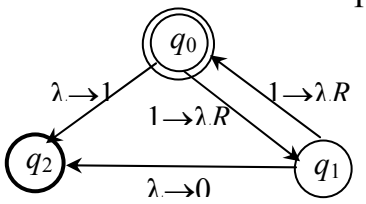


Рис. 6.13

Пример 2. Покажем, что предикат $P(x, y)$ = «x меньше или равно y» правильно вычислим. Машина Тьюринга входное слово $1^x * 1^y$ будет «обрабатывать» следующим образом. Она будет поочередно затирать сначала левую единицу первого аргумента, уходить вправо, затем правую единицу второго аргумента и возвращаться влево. Если при этом вначале «обнулится» первый аргумент, то $x \le y$; если второй, то – $x > y$. Диаграмма машины приведена на рис. 6. 14. □

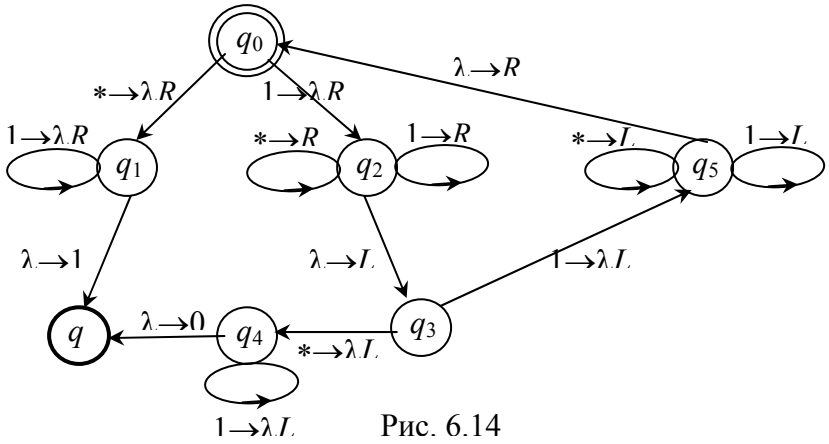


Рис. 6.14

Определение. Пусть $y = f(x_1, x_2,\ldots, x_n)$ и $y = g(x_1, x_2,\ldots, x_n)$ – функции, а $P(x_1, x_2,\ldots, x_n)$ – предикат, заданные на $\boldsymbol{N}$. Функция

$$h(x_1, x_2,\ldots, x_n) = \begin{cases} f(x_1, x_2,\ldots, x_n), \text{ если } P(x_1, x_2,\ldots, x_n) \text{ истинно,} \\ g(x_1, x_2,\ldots, x_n), \text{ если } P(x_1, x_2,\ldots, x_n) \text{ ложно,} \end{cases}$$

называется *ветвлением* функций $y = f(x_1, x_2,\ldots, x_n)$ и $y = g(x_1, x_2,\ldots, x_n)$ по предикату $P(x_1, x_2,\ldots, x_n)$.

Ближайшей целью является доказательство того, что ветвление правильно вычислимых функций по правильно вычислимому предикату является правильно вычислимым. Для упрощения обозначений в следующей теореме это будет сделано только в одноместном случае.

Теорема 6.3. Если функции $y = f(x)$, $y = g(x)$ и предикат $P(x)$ правильно вычислимы, то и ветвление $f(x)$ и $g(x)$ по $P(x)$ правильно вычислимо.

Доказательство. Обозначим через S, T и R – машины Тьюринга, правильно вычисляющие $f(x)$, $g(x)$ и $P(x)$ соответственно. Пусть $U = \{u_1, \ldots, u_k\}$, $V = \{v_1, \ldots, v_l\}$ и $W = \{w_1, \ldots, w_m\}$ – внутренние алфавиты этих машин. Рассмотрим еще машину Тьюринга C с множеством внутренних состояний $D = \{d_1, \ldots, d_e\}$, которая конфигурацию $d_1 1^x$ переводит в конфигурацию $1^x\#1^x$, где # – символ, не принадлежащий внешним алфавитам машин S, T и R. Машину C можно "сконструировать" аналогично машине Тьюринга, названной "переводчиком" в §2. Будем считать, что внутренние алфавиты машин не имеют общих элементов и выполняется соглашение о том, что состояние с наименьшим индексом – начальное, с наибольшим – заключительное. Условимся также и о том, что машины T, S, R и C удовлетворяют утверждению теоремы 6.1.

Машина Тьюринга M, правильно вычисляющая ветвление, строится следующим образом. Множество внутренних состояний машины есть

$$Q = \{u_1, \ldots, u_{k-1}, v_1, \ldots, v_{l-1}, w_1, \ldots, w_m, d_1, \ldots, d_{e-1}, q_0, q_1, q\},$$

где $q_0, q_1, q \notin U \cup V \cup W \cup D$, d_1 – начальное, q – заключительное состояния. Внешний алфавит машины M – объединение внешних алфавитов четырех машин.

Вначале машина M работает так же, как и машина C, т.е. копирует входное слово. Затем на втором экземпляре входного слова M работает как машина R. Для этого команды машины C включаются в программу машины M с заменой команд вида $d_i a \to d_e a'$ на $d_i a \to w_1 a'$. Машина M, работая как машина R, продолжает работу с конфигурации $1^x\#w_1 1^x$. Если $P(x)$ истинно, M должна на первом экземпляре входного слова работать, как S, а если $P(x)$ ложно, то как T. Для этого к командам машины M добавим команды машины R.

Если $P(x)$ истинно, то на последнем такте машина R (и M) применила команду вида $w_i a \to w_m 1$. Если $P(x)$ ложно, то R (и M) применила команду вида $w_i a \to w_m 0$. Добавим к машине M команды $w_m 1 \to q_1 \lambda L$, $w_m 0 \to q_0 \lambda L$ и команды, позволяющие M затереть символ # и сдвинуться в крайнюю левую ячейку (в состоянии q_0 и q_1). Добавим к M также команды $q_1 1 \to u_1 1$, $q_1 \lambda \to u_1 \lambda$, $q_0 1 \to v_1 1$, $q_0 \lambda \to v_1 \lambda$, позволяющие "перейти" к машинам S или T для вычисления функций $y = f(x)$ или $y = g(x)$. Осталось команды машины M пополнить командами машин S и T, заменив ко-

манды вида $u_i a \to u_k a'$ и $v_j a \to v_l a'$ на соответственно команды $u_i a \to q a'$ и $v_j a \to q a'$.□

Выше было показано, что сумма $x+y$ правильно вычислима. Очевидно, что функция $x+a$ тоже правильно вычислима. Из теоремы 6.2 следует правильная вычислимость любой линейной функции $f(x_1, \ldots, x_n) = a_1 x_1 + \ldots + a_n x_n + a$. Далее, несложно построить машину Тьюринга, вычисляющую разность $x - y$. Тогда из теоремы 6.3 следует правильная вычислимость функции

$$|x-y| = \begin{cases} x-y, \text{ если } x \geq y, \\ \\ y-x \text{ в противном случае.} \end{cases}$$

Умножение $x \cdot y$ является правильно вычислимой функцией. (Программа соответствующей машины есть в книге О.П. Кузнецова и Г.М. Адельсона-Вельского [КА-В].) Тогда из теоремы 6.2 следует, что любой многочлен является правильно вычислимой функцией.

Несложно доказать правильную вычислимость функций $d(x, y)$ – частное от деления x на y и $r(x, y)$ – остаток от деления x на y (см. [КА-В]).

Начиная с этого момента мы будем рассматривать только правильно вычислимые функции, поэтому прилагательное «правильная» в словосочетании «правильно вычислимая функция» будем, как правило, опускать.

Аналогично понятию вычислимой функции натурального аргумента, можно ввести более общее понятие – понятие вычислимой словарной функции. Пусть W – некоторое подмножество внешнего алфавита A машины Тьюринга, не содержащее пустого символа. Тогда функция f: $W^* \times W^* \times \ldots \times W^* \to W^*$ называется словарной. Она так же, как и функция натурального аргумента, может быть не всюду определена.

Определение вычислимой словарной функции похоже на определение вычислимой функции натурального аргумента. Тем не менее, приведем это определение.

Определение. Словарная функция $f(x_1, \ldots, x_n)$, заданная на множестве W, называется *правильно вычислимой* на машине M, если M, начиная работу на ленте в конфигурации $q_0 x_1 * x_2 * \ldots * x_n$ заканчивает ее в конфигурации $q_s f(x_1, \ldots, x_n)$, если $f(x_1, \ldots, x_n)$ определена, и работает бесконечно, если $f(x_1, \ldots, x_n)$ не определена. (Здесь q_0 – начальное, q_s – заключительное состояния, а $x_1, \ldots, x_n$ – элементы множества W^*.)

Понятно, что вычислимая функция натурального аргумента является вычислимой словарной функцией. В качестве других примеров вычислимых функций можно привести $f(x) = x^r$, $g(x_1, x_2) = x_2x_1$. Правильная вычислимость первой функции установлена в примере 3. Правильную вычислимость второй предлагается доказать читателю.

§4. Универсальные машины Тьюринга

Основная цель этого параграфа – убедиться в существовании машины Тьюринга U, которая может моделировать работу любой другой машины Тьюринга. Такую машину U будем называть *универсальной*.

Примем два соглашения.

Во-первых, условимся, что внешний алфавит любой машины Тьюринга есть подмножество множества $\boldsymbol{A} = \{a_0, a_1, \ldots, a_n, \ldots\}$, внутренний алфавит – подмножество множества $\boldsymbol{Q} = \{q_0, q_1, \ldots, q_n, \ldots\}$. Во-вторых, условимся индексы в обозначениях символов внешнего алфавита и внутренних состояний писать в двоичном коде и записывать их обычным шрифтом. Например, команда $q_3a_2 \to q_2a_5L$ запишется так: $q11a10 \to q10a101L$.

Внешний алфавит B машины U будет содержать символы 0, 1, q, a, L, R, $\Rightarrow$, $*$, ;, m, λ (и еще некоторые символы, обозначения которых вводить не будем). Внутренний алфавит машины U нам будет удобно представить как объединение непересекающихся множеств P_1 и P_2, $P_1 = \{p_0, \ldots, p_k\}$, $P_2 = \{p_{k+1}, \ldots, p_l\}$ для некоторых k и l. Чтобы выполнить первое соглашение, будем считать, что имеется соответствие между элементами множеств B и $\boldsymbol{A}$, которое устанавливается следующей таблицей

0	1	q	a	L	R	$\Rightarrow$	$*$	;	m	λ
a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}

Считаем, что $P \cup R \subseteq \boldsymbol{Q}$, но соответствие между обозначениями элементов из $P \cup R$ и из $\boldsymbol{Q}$ фиксировать не будем.

Пусть M – некоторая машина Тьюринга имеющая внешний алфавит A и внутренний алфавит Q, $A \subseteq \boldsymbol{A}$, $Q \subseteq \boldsymbol{Q}$. Машина U в начальный момент имеет на ленте программу машины M, команды которой разделены символом “точка с запятой” и начальную конфигурацию, отделенную от программы разделителем $*$ (см.рис. 6.11).

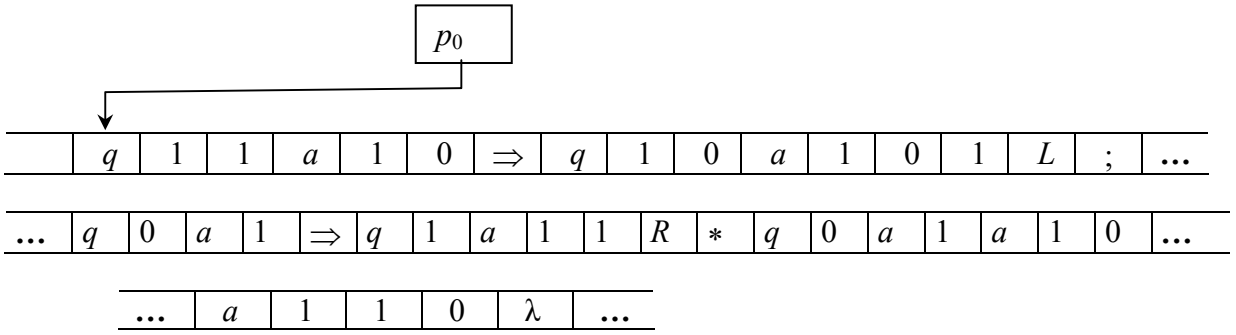


Рис. 6.15

Отметим, что на рис. 6.15 изображена одна лента (а не три), в начале ленты (т. е. начиная с первого непустого символа) записана программа машины M:

$$q_3a_2 \to q_2a_5L;\ldots;q_0a_1 \to q_1a_3R;$$

а затем после символа * записана начальная конфигурация $q_0a_1a_2\ldots a_6$. (Мы на ленте вместо “одинарной” стрелки в записи команды пишем “двойную” для того, чтобы отличать стрелку в командах моделируемой машины M от стрелки в командах машины U.) Обозначим часть входной цепочки от начала до символа * через v, а оставшуюся после * часть входной цепочки – через w.

Моделирование одного такта работы машины M можно разделить на два этапа, которые назовем “поиск” и “исполнение” На этапе поиска машина U работает примерно так же, как и контролер из §2. В начале работы машина U находится в состоянии p_0 и обозревает первую ячейку, в которой записан символ q. На первом такте U заменяет его на m, исполняя команду $p_0q \to p_1mR$. Далее машина U определяет, совпадает ли отрезок слова v от m до первого вхождения символа ⇒ с отрезком слова w от символа q до второго после q символа a или символа λ. В ситуации, изображенной на рис. 5.11 эти отрезки равны соответственно $v' = 11a10$ и $w' = 0a1$. Именно при этой проверке совпадения U работает аналогично контролеру из §2. Машине U для выполнения проверки может понадобиться больше внутренних состояний, чем контролеру. Если отрезки v' и w' совпадают, то это означает, что найдена команда машины M, условие которой истинно на текущей конфигурации. Машина U в этом случае уходит на этап исполнения этой команды, находясь в состоянии r_1 и обозревая маркер m. Если же указанные отрезки не совпадают, то машина U на место маркера m записывает символ q, находит символ q следующий за ближайшей “точкой с запя-

той” и повторяет этап поиска, находясь во внутреннем состоянии p_0.

В случае, когда не существует команды, условие которой истинно на текущей конфигурации, машина M останавливается в ситуации неопределенности. Для машины U это будет означать, что она находится в состоянии p_0 и обозревает символ $*$. Машина U в этом случае в текущей конфигурации $a_{i1}\ldots a_{ik}q_l a_{ik+1}\ldots a_{im}$ убирает подцепочку, соответствующую q_l (напомним, что число l представ-лено в двоичном коде), сдвигает $a_{ik+1}\ldots a_{im}$ влево до a_{ik} и останавливается. После символа $*$ на ленте записано состояние ленты машины M на момент останова.

Опишем этап исполнения при моделировании команды машины М. Сделаем это на примере машины, система команд и начальная конфигурация которой представлена на рис. 6.16.

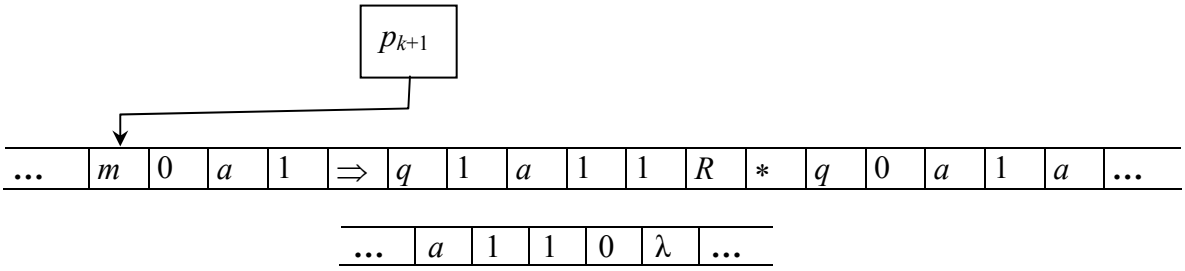


Рис. 6.16

На начальной конфигурации истинно условие команды, которая в списке команд записана последней (см. рис. 6.16).

Машина U маркер m заменяет на символ q, сдвигается вправо и, пройдя “двойную стрелку”, переходит в состояние p_{k+2}. В этом состоянии она идет далее вправо до символа a, следующего за q, в текущей конфигурации меняет подцепочку $a1$ на $a11$. Для этого ей придется сделать сдвиг отрезка цепочки слова w, начинающегося с a, на одну ячейку вправо. Сделать это можно аналогично машине, названной в §2 переносчиком. Затем машина идет влево до символа q, затирает цепочку $q0$ в текущей конфигурации и между $a1$ и a10 записывает $q1$. Далее машина U проверяет, является ли состояние, в которое перешла моделируемая машина M заключительным или нет. (В нашем случае – это состояние q_1.) Если q_1 незаключительное состояние, то машина возвращается в начало входной ленты, переходя в состояние p_0 (см.рис. 6.17).

Если состояние q_1 является заключительным, то машина U затирает символ $*$, список команд машины M, сдвигает заключительное состояние ленты машины M в начало “своей” ленты и останавливается.

Описание универсальной машины U закончено. Разумеется, такая машина не является единственной. Различные универсальные машины могут отличаться как алфавитами (внешним и внутренним), так и способами реализации отдельных блоков

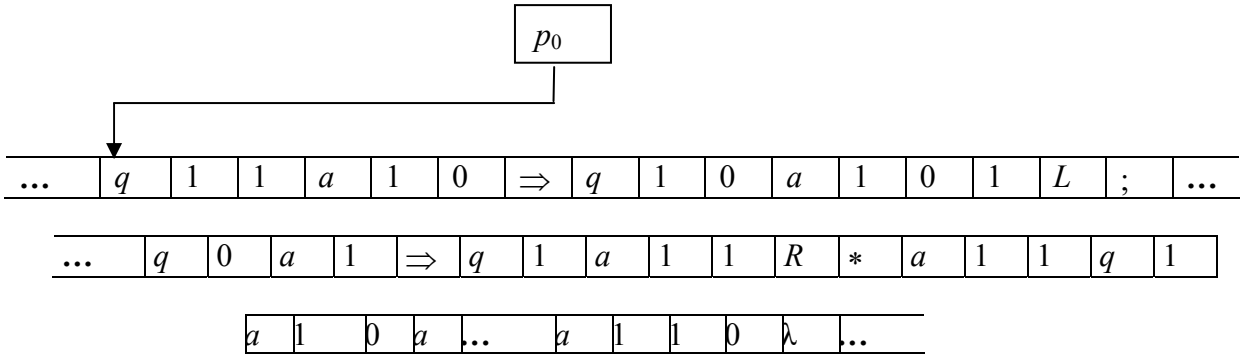


Рис. 6.17

«Сконструированная» выше универсальная машина имеет очень большой внешний алфавит и большое количество внутренних состояний. Однако, оказывается, существуют «малые» универсальные машины. Минским «построена» универсальная машина Тьюринга, у которой 4 символа внешнего алфавита и 7 внутренних состояний [Мин, стр. 328-334].

§5. Алгоритмически неразрешимые проблемы

В конце прошлого – начале нынешнего века были поставлены ряд вопросов о существовании алгоритмов решающих ту или иную задачу. Наиболее известным из таких вопросов является так называемая десятая проблема Гильберта, упомянутая в начале главы. Напомним, что эта проблема есть вопрос о том, существует ли алгоритм, который по любому уравнению (от нескольких неизвестных) с целыми коэффициентами определяет имеет ли уравнение целочисленное решение или нет. Поскольку интуитивное понятие алгоритма мы формализовали в виде машины Тьюринга, то вопрос о существовании алгоритма можно заменить на вопрос о существовании такой машины.

Обсуждению подобных вопросов и посвящен настоящий параграф.

Мы начнем с проблемы остановки машины Тьюринга. *Проблема остановки* – вопрос о том, существует ли машина Тьюринга M_0, которая по программе любой машины Тьюринга X и слову w определяет остановится ли X начиная работу на входном слове w или нет.

Ответ на этот вопрос содержится в следующем утверждении.

Теорема 6.4. Проблема остановки решается отрицательно.

Доказательство. Считаем, что действуют соглашения, принятые в начале §5: внешний алфавит любой машины – подмножество множества $\boldsymbol{A} = \{a_0, a_1, a_2, \ldots\}$, внутренний – подмножество множества $\boldsymbol{Q} = \{q_0, q_1, q_2, \ldots\}$; индексы записываются в двоичном коде. Предположим противное. Пусть существует машина Тьюринга M_0, которая начиная работу на входной ленте, изображенной на рис. 6.18,

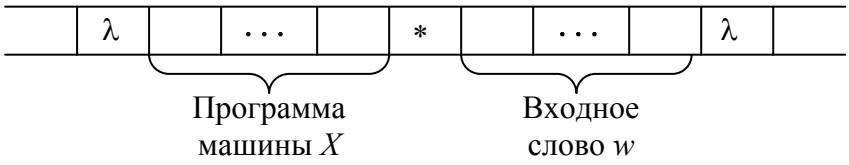


Рис. 6.18

останавливается и печатает 1, если X останавливается, начиная работу на слове w, останавливается и печатает 0, если X не останавливается на слове w.

Рассмотрим еще машину M_1, которая, начиная работу на ленте, изображенной на рис.6.18, останавливается и печатает 0, если M_0 печатает 0, начиная на той же ленте, и работает бесконечно, если M_0 печатает 1.

Введем третью машину – M_2. Машина M_2, имея на входе ленту, изображенную на рис. 6.19, дублирует входное слово (см. рис. 6. 20) и далее работает, как машина M_1.

Рассмотрим, как будет работать машина M_2, если $X = M_2$, т.е. если в качестве входного слова машине M_2 предлагается собственная программа. Тогда

M_2 останавливается на входном слове “программа M_2” $\Leftrightarrow$
M_1 останавливается на входном слове
“программа M_2 * программа M_2” $\Leftrightarrow$
M_0 печатает 0 на входном слове
“программа M_2 * программа M_2” $\Leftrightarrow$
M_2 не останавливается на входном слове “программа M_2”.

Рис. 6.19

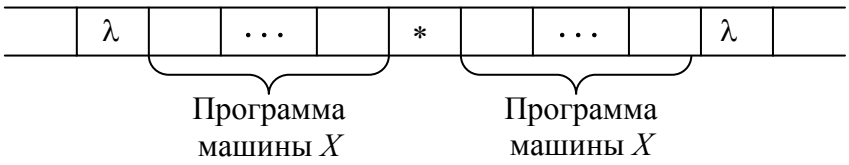


Рис. 6.20

Получили противоречие: M_2 останавливается, начиная работу на своей программе тогда и только тогда, когда она не останавливается. Следовательно, машины Тьюринга, решающей проблему остановки не существует. □

В следующем параграфе нам понадобится частный случай проблемы остановки, так называемая проблема самоприменимости. Эта проблема представляет собой вопрос о том, существует ли машина Тьюринга, которая по данной машине X определяет, остановится ли X, начиная работу на своей собственной программе.

Теорема 6.5. Проблема самоприменимости решается отрицательно.

Доказательство этой теоремы проводится так же, как и теоремы 6.4, поэтому здесь не приводится.

Третье утверждение, которое мы здесь приведем, называется теоремой Райса. Теорема связана с вопросом алгоритмической распознаваемости тех или иных свойств функций, вычислимых на машинах Тьюринга. Свойство функций называется *нетривиальным*, если существует вычислимая на машине Тьюринга функция, обладающая этим свойством, и вычислимая функция, таким свойством не обладающая. Примерами подобных свойств являются ограниченность, периодичность, быть постоянной функцией, иметь данное число значений, быть всюду определенной функцией.

Теорема 6.6. Никакое нетривиальное свойство вычислимых функций не распознается на машинах Тьюринга.

С доказательством теоремы можно познакомиться по книге [КА-В] из списка литературы.

Вопросы, связанные с алгоритмической разрешимостью тех или иных задач, возникают в различных областях математики. Приведем пример.

Рассмотрим задачу распознавания тождественной истинности формул логики предикатов. Напомним, что формула (без свободных переменных) называется тождественно истинной,

если она истинна при любой интерпретации. Следующее утверждение называется теоремой Черча.

Теорема 6.7. Не существует алгоритма, распознающего тождественно истинные формулы среди всех формул логики предикатов.

Доказательство здесь приводить не будем. С ним можно ознакомиться по книге А.И.Мальцева, приведенной в списке литературы.

§6. Рекурсивные и рекурсивно перечислимые множества

Различают два способа алгоритмического задания множеств. (Для определенности будем говорить о множествах натуральных чисел.) Если существует алгоритм, который по любому натуральному числу n определяет, принадлежит ли n данному множеству A или не принадлежит, то говорят, что A задается *распознающим* алгоритмом. Если же существует алгоритм, который работает бесконечно долго и время от времени выдает элементы множества A, то в этом случае говорят, что A задается *порождающим* алгоритмом. В точных терминах в первом случае говорят о рекурсивности, во втором – о рекурсивной перечислимости. Дадим необходимые определения.

Определение. Множество A натуральных чисел называется *рекурсивным*, если характеристическая функция этого множества вычислима на машине Тьюринга.

Напомним, что характеристической функцией множества A называется функция

$$f_A(x) = \begin{cases} 1, \text{ если } x \in A, \\ 0, \text{ если } x \notin A. \end{cases}$$

Довольно очевидно, что рекурсивность можно определить и через введенные ранее понятия. Действительно, на A можно смотреть как на множество истинности одноместного предиката. Тогда рекурсивность множества A равносильна правильной вычислимости соответствующего предиката.

Определение. Множество A называется *рекурсивно перечисли-мым*, если A есть область значения подходящей одноместной всюду определенной вычислимой функции или $A=\varnothing$.

Другими словами, A – рекурсивно перечислимо, если $A=\varnothing$ или существует всюду определенная функция $y = g(x)$, вычислимая на машине Тьюринга и такая, что

$$A = \{g(0), g(1), g(2), \ldots\}$$

Разумеется, возможны равенства $g(k) = g(l)$ при некоторых k и l.

Условимся область определения функции $y = f(x)$ обозначать через $D(f)$, а область значений – через $R(f)$.

Интуитивно чувствуется, что рекурсивность более сильное требование, нежели рекурсивная перечислимость. Сформулируем это наблюдение в виде следующего утверждения.

Теорема 6.8. Если множество A рекурсивно, то A и A' рекурсивно перечислимы.

Доказательство. Случай, когда $A = \varnothing$ или $A = \boldsymbol{N}$ тривиален, поэтому его рассматривать не будем. Предположим, что $A \neq \varnothing$ и $A \neq \boldsymbol{N}$. Зафиксируем элементы $a \in A$, $b \in A'$. Рассмотрим функции

$$g_1(x) = \begin{cases} x, \text{ если } f_A(x)=1, \\ a, \text{ если } f_A(x)=0, \end{cases} \quad \text{и} \quad g_2(x) = \begin{cases} x, \text{ если } f_A(x)=0, \\ b, \text{ если } f_A(x)=1. \end{cases}$$

Функции $g_1(x)$ и $g_2(x)$ всюду определены по построению. С другой стороны, имея машину Тьюринга, вычисляющую $f_A(x)$, легко построить машины, вычисляющие функции $g_1(x)$ и $g_2(x)$. Убедимся в этом для первой функции.

Пусть машина M вычисляет функцию $f_A(x)$. Тогда машина M_1, вычисляющая функцию $g_1(x)$, строится следующим образом. Вначале машина M_1 конфигурацию q_0x перерабатывает в конфигурацию $x*q_0x$. Затем на втором экземпляре аргумента x M_1 работает также, как M. Если $x \in A$, то $f_A(x)=1$, и M_1 приходит к конфигурации $x*q'1$. В этом случае M_1 затирает “звездочку” и “единицу”, уходит на крайнюю левую ячейку и останавливается. Если же $x \notin A$, то $f_A(x)=0$ и машина приходит к конфигурации $x*q'0$. В этом случае M_1 затирает x, $*$ и 0, печатает число a и останавливается на крайней левой ячейке. Вычислимость функции $g_1(x)$ установлена. Аналогично доказывается вычислимость функции $g_2(x)$.

Нетрудно понять, что область значений функции $g_1(x)$ есть A, функции $g_2(x)$ – A'. Докажем это утверждение для первой функции. Если $x \in A$, то $f_A(x)=1$ и $g_1(x)=x$. Это означает, что $x \in R(g_1)$. Обратно, пусть $x \in R(g_1)$. Тогда $g(x)=x$ или $x=a$. В первом случае $f_A(x)=1$ и $x \in A$; во втором также $x \in A$ (поскольку

$a \in A$). Следовательно, $A = R(g_1)$. Аналогично доказывается, что $A' = R(g_2)$. □

Итак, доказано, что всякое рекурсивное множество является рекурсивно перечислимым. Естественно поставить вопрос, справедливо ли обратное? Ответ на этот вопрос содержится в теореме 6.9.

Проведем так называемую “арифметизацию” машин Тьюринга, в результате которой каждая машина M получит в соответствие натуральное число – номер машины M. Как и в § 4, будем считать, что внешний алфавит любой машины есть подмножество множества $\boldsymbol{A} = \{a_0, a_1, \ldots, a_n, \ldots\}$, а внутренний алфавит – подмножество множества $\boldsymbol{Q} = \{q_0, q_1, \ldots, q_n, \ldots\}$. Кроме того, индексы в обозначениях элементов внешнего и внутреннего алфавитов будем записывать в двоичном коде. Поставим в соответствие символам 0, 1, q, a, L, R, $\rightarrow$, ; десятичные цифры 0, 1, 2, 3, 4, 5, 6, 7 соответственно. Заменим в записи команд все символы указанными цифрами. Полученную последовательность цифр будем воспринимать как десятичный код некоторого натурального числа – *номера команды*.

Например, команда $q_2a_1 \rightarrow q_0a_3R$ будет иметь номер 210316203115. Номер команды C будем обозначать через $n(C)$. Если программа машины M состоит из команд C_1, C_2, …, C_t, то *номером* $n(M)$ *машины* M будем называть число $n(C_1)7n(C_2)7\ldots7n(C_t)$, т.е. последовательность цифр, воспринимаемую как десятичный код числа $n(M)$. У машины Тьюринга может быть несколько номеров, поскольку команды машины можно упорядочивать по-разному. Отметим, что существуют натуральные числа, которые не являются номером никакой машины Тьюринга. Таковыми являются, например, числа, имеющие в десятичном разложении цифру 9.

Отметим вначале, что существует машина Тьюринга M_0, которая по единичному коду числа x определяет, является ли x номером какой-нибудь машины Тьюринга. Если x является номером некоторой машины, то M_0 печатает на ленте программу этой машины (для этого машине M_0 надо “уметь” делить с остатком на 10) и останавливается на самой левой непустой ячейке. Если же x не является номером никакой машины, то M_0 печатает на ленте 0 и останавливается в заключительном состоянии.

Теорема 6.9. Существует рекурсивно перечислимое множество, которое не является рекурсивным.

Доказательство. Пусть множество A есть множество номеров тех машин Тьюринга, которые останавливаются, начиная работу на собственной программе. Из теоремы 6.5 предыдущего параграфа легко следует, что множество A нерекурсивно.

Докажем рекурсивную перечислимость множества A. Если $A = \varnothing$, то A является рекурсивно перечислимым по определению. Пусть $A \neq \varnothing$. Зафиксируем в A некоторый элемент c.

Рассмотрим предикат $H(k, l)$, определяемый следующим образом: $H(k, l) = 1 \Leftrightarrow$ число k является номером некоторой машины K и эта машина, проработав на собственной ленте не более, чем l тактов, останавливается. Предикат $H(k, l)$ всюду определен. Убедимся в том, что он вычислим. Вычисляющая его машина T_0 может работать следующим образом. Сначала она работает так же, как машина M_0 на слове 1^k, т. е. на первом аргументе предиката. Если M_0 останавливается и печатает 0 (это означает, что k не является номером никакой машины Тьюринга), то T_0 также останавливается и печатает 0. Пусть M_0 останавливается и печатает на ленте программу машины K, имеющей номер k. Тогда машина T_0 работает так же как универсальная машина U, моделируя выполнение машиной K на собственной программе не более l тактов. Если моделируемая машина K при этом останавливается, то T_0 печатает 1 и останавливается. Если же машина K, сделав l тактов, не остановилась, то T_0 останавливается и печатает 0.

Предикат $H(k, l)$ будем представлять себе в виде бесконечной таблицы

k	l			
	0	1	2	…
0	$H(0, 0)$	$H(0, 1)$	$H(0, 2)$	…
1	$H(1, 0)$	$H(1, 1)$	…	…
2	$H(2, 0)$	…	…	…
…	…	…	…	…

Занумеруем клетки таблицы следующим образом: (0, 0), (0, 1), (1, 0), (2, 0), (1, 1), и т. д.

Рассмотрим теперь еще одну машину Тьюринга, которую будем обозначать буквой T. Машина T начинает работу, имея на ленте слово 1^x. Вначале она по данному числу x находит клетку таблицы с номером x. Пусть это будет (k, l). Тогда машина T работает так же, как машина T_0, вычисляющая предикат $H(k, l)$. Если T_0 останавливается и печатает 0, то T также останавливается и печатает число c. Если же T_0 завершает работу и печатает 1, то T также завершает работу и печатает число x. Из описания машины T видно, что она вычисляет некоторую одноместную

всюду определенную функцию, область значения которой равна множеству A. Это означает, что A рекурсивно перечислимо. □

§7. Многоленточные машины

В многих приложениях часто удобно считать, что машина Тьюринга содержит не одну ленту, а несколько лент, разбитых на ячейки и бесконечных в обе стороны. Как и обычная (одноленточная) машина, многоленточная машина имеет внешний алфавит, в котором выделен пустой символ, и внутренний алфавит (алфавит внутренних состояний). В этом алфавите выделено начальное и заключительное состояния. В ячейках всех лент записаны символы внешнего алфавита (по одному в ячейке), причем в любой момент времени на любой ленте только конечное число ячеек занято непустыми символами. Команда m-ленточной машины есть выражение вида

$$(q, a_1, a_2, \ldots, a_m) \to (q', (a_1', d_1), (a_2', d_2), \ldots, (a_m', d_m)),$$

где q и q' – элементы внутреннего алфавита, буквы a с индесами и штрихами – элементы внешнего алфавита, $d_i \in \{L, R, H\}$ для $1 \le i \le m$. Команда выполняется следующим образом. Пусть на начало очередного такта машина находится в состоянии q, на i-той ленте обозревается символ a_i. Тогда в результате выполнения команды машина перейдет в состояние q', в обозреваемую ячейку запишется символ a_i' и по i-той ленте произойдет сдвиг влево, если $d_i = L$, вправо, если $d_i = R$, и машина будет обозревать прежнюю ячейку, если $d_i = H$. Как и в одноленточном случае, программа – это множество команд с различными левыми частями (левая часть команды – выражение до «стрелки»). Многоленточная машина начинает работу и заканчивает ее так же, как и одноленточная машина.

Предположим, что на начало выполнения очередного такта трехленточная машина находилась в ситуации, изображенной на рис.6.21. Тогда после выполнения команды

$$(q, a1, a2, a3) \to (r, (b_1, H), (b_2, R), (b_3, L))$$

машина будет находиться в ситуации, изображенной на рис. 6.22.

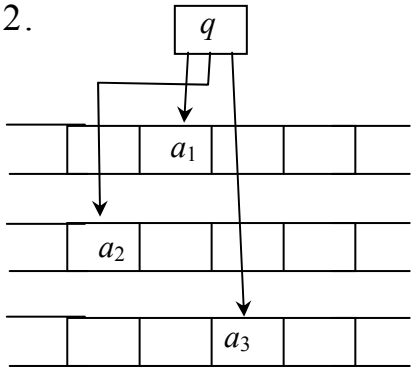


Рис. 6.21

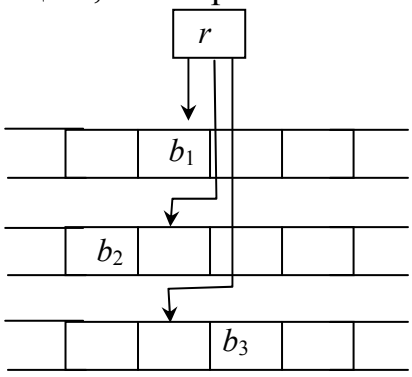


Рис. 6.22

Для многоленточных машин естественным образом определяется правильная вычислимость функции. Мы не будем полностью приводить это определение. Отметим только, что для вычисления k-местной функции на m-ленточной машине Тьюринга первый аргумент записывается на первой ленте, второй – на второй и т. д. Лента с номером k+1 отводится под значение функции. Машина может содержать еще ленты, т. е. m может быть больше, чем k+1. Эти ленты часто называют *рабочими лентами*.

Пример. Напишем программу двухленочной машины Тьюринга, вычисляющей функцию $y = 2x$. Аргумент будет записан на первой ленте в единичном коде, вторая лента пуста. Машина начинает работу, обозревая на первой ленте самую левую единицу (если $x \neq 0$; если же $x = 0$ машина сразу переходит в заключительное состояние). Машина будет считывать аргумент, двигаясь вправо и дублировать его на второй ленте. Найдя на первой ленте первую (справа от аргумента) пустую ячейку, машина по первой ленте будет двигаться влево, затирая единицы и дублируя их на второй ленте. По второй ленте машина, по-прежнему, двигается вправо. Найдя на первой ленте первую (слева от аргумента) пустую ячейку, машина остановится в заключительном состоянии q. Приведем список команд такой машины:

$(q_0, 1, \lambda) \to (q_1, 1R, 1R)$,
$(q_0, \lambda, \lambda) \to (q, \lambda, \lambda)$,
$(q_1, 1, \lambda) \to (q_1, 1R, 1R)$,
$(q_1, \lambda, \lambda) \to (q_2, \lambda L, \lambda)$,
$(q_2, 1, \lambda) \to (q_2, \lambda L, 1R)$,
$(q_2, \lambda, \lambda) \to (q, \lambda, \lambda)$. □

Теорема 6.10. Функция вычислима на одноленточной машине Тьюринга тогда и только тогда, когда она вычислима на многоленточной машине.

Доказательство этого утверждения предоставляется читателю.

§8. Рекурсивные функции

Вернемся к обсуждению понятия алгоритма, проведенному в §1. Там в качестве формального аналога понятия алгоритма было предложено понятие машины Тьюринга. Если дана машина Тьюринга M, на ленте которой записано входное слово w, M находится в начальном внутреннем состоянии и обозревает первую ячейку ленты, то машина M начинает работать или, как еще

иногда говорят, “производить вычисления”. В этом смысле на машину Тьюринга смотрят как на *модель вычислений*. Машина Тьюринга не единственная модель вычислений. В тридцатых-сороковых годах прошлого столетия был предложен ряд таких моделей. Мы в этом параграфе рассмотрим одну из них – модель рекурсивных функций.

Обозначим буквой ***F*** множество всех функций натурального аргумента. Напомним, что функция натурального аргумента не обязательно всюду определена.

Определение. Функции $O(x) = 0$, $s(x) = x+1$, $I_n^m(x_1, \ldots, x_n) = x_m$, где $1 \le m \le n$, назовем *простейшими*.

Введем на ***F*** три оператора: суперпозиции, примитивной рекурсии и минимизации.

Определение. Функция $f(x_1, \ldots, x_n)$ называется *суперпозицией* функций $g(x_1, \ldots, x_k)$, $h_1(x_1, \ldots, x_n)$, …, $h_к(x_1, \ldots, x_n)$, если

$$f(x_1, \ldots, x_n) = g(h(x_1, \ldots, x_n), \ldots, h_k(x_1, \ldots, x_n)).$$

Например, функция $f(x_1, x_2) = x_1^2 + x_1x_2$ – суперпозиция функций $g(x_1, x_2) = x_1+x_2$, $h_1(x_1, x_2) = x_1^2$, $h_2(x_1, x_2) = x_1 \cdot x_2$, поскольку $f(x_1, x_2) = g(h_1(x_1, x_2), h_2(x_1, x_2))$. Заметим, что $h_1(x_1, x_2) = h_2(x_1, I_2^1(x_1, x_2))$. Функция I_m^n, как мы видим, позволяет вводить фиктивные аргументы. Легко видеть, что суперпозиция всюду определенных функций будет всюду определенной.

Определение. Функция $f(x_1, \ldots, x_n, y)$ получается *примитивной рекурсией* из $g(x_1, \ldots, x_n)$ и $h(x_1, \ldots, x_n, y, z)$, если выполняются следующие условия:

1) $f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n)$,

2) $f(x_1, \ldots, x_n, y+1) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$.

Например, функция $f(x, y) = x+y$ получается из функций $g(x) = x$, $h(x, y, z) = z+1$ примитивной рекурсией. Действительно, $f(x, 0) = x+0 = x = g(x)$, $f(x, y+1) = x+(y+1) = (x+y)+1 = f(x, y)+1 = h(x, y, f(x, y))$. Нетрудно показать, что примитивная рекурсия всюду определенных функций будет всюду определена.

Определение. Функция $f(x_1, \ldots, x_n)$ получается из функций $g(x_1, \ldots, x_n)$ *минимизацией*, если $f(x_1, \ldots, x_n)$ равна наименьшему значению u такому, что

$$g(x_1, \ldots, x_{n-1}, u) = x_n.$$

Пусть $g(x_1, x_2) = x_1+x_2$. Тогда если $f(x_1, x_2)$ получается из $g(x_1, x_2)$ минимизацией, т.е. $f(x_1, x_2) = min\{u \mid x_1+u = x_2\}$, то $f(x_1, x_2) = u = x_2-x_1$. Этот пример, в частности показывает, что минимизацией из всюду определенной функции можно получить не всюду определенную.

Введем три класса функций.

Определение. Функция называется *примитивно-рекурсивной*, если она получается из простейших функций с помощью конечного числа применений операторов суперпозиции и примитивной рекурсии.

Функция $f(x, y) = x+y$ является примитивно-рекурсивной. Действительно, $f(x, 0) = x = I_1^1(x)$, $f(x, y+1) = f(x, y)+1 = s(f(x, y))$. Приведем еще в качестве примеров умножение $g(x, y) = xy$ и возведение в степень $h(x, y) = x^y$:

$$g(x, 0) = 0 = 0(x),\ g(x, y+1) = x(y+1) = x+xy = f(x, g((x, y))) \text{ и}$$

$$h(x, 0) = 1 = s(0),\ \ h(x, y+1) = x^{y+1} = xx^y = g(x, h(x, y)).$$

Примитивно-рекурсивными являются также функции $m(x, y) = |x-y|$, $g(x, y)$ – частное от деления y на x, $r(x, y)$ – остаток от деления y на x (см., например, упоминавшуюся книгу А.И.Мальцева из списка литературы).

Определение. Функция называется *рекурсивной* (в другой терминологии *частично-рекурсивной*), если она получается из простейших с помощью суперпозиции, примитивной рекурсии и минимизации.

Всякая примитивно-рекурсивная функция является рекурсивной. Выше было показано, что разность $d(x, y) = y-x$ получается минимизацией из суммы $f(x, y) = x+y$. Следовательно, $d(x, y)$ – рекурсивная функция, которая не является примитивно-рекурсивной (поскольку не всюду определена).

Определение. Всюду определенные рекурсивные функции называются *общерекурсивными*.

Поскольку примитивно-рекурсивные функции всюду определены, всякая примитивно-рекурсивная функция является общерекурсивной. Обратное утверждение несправедливо, т.е. существует общерекурсивная функция, которая не является примитивно-рекурсивной. С доказательством этого утверждения можно познакомиться по книге А.И.Мальцева.

Рассмотренные понятия представляют собой другой подход формализации понятия вычислимости, представляют другую модель вычислений. В этой модели, как мы видели, фиксируются некоторые базовые функции (простейшие функции) и способы порождения новых функций из данных (операторы над функциями). Естественно поставить вопрос о сравнении “вычислительных возможностей” моделей: как соотносятся классы вычислимых функций и рекурсивных функций? Ответ содержится в следующем утверждении.

Теорема 6.11. Класс функций, вычислимых на машинах Тьюринга совпадает с классом рекурсивных функций.

Доказательство теоремы здесь не приводится. Его можно найти в книгах А.И.Мальцева или О.П.Кузнецова и Г.М.Адельсона-Вельского, приведенных в списке литературы.

§9. Нормальные алгоритмы

Этот параграф посвящен изложению еще одного варианта формализации понятия алгоритм, который был предложен отечественным математиком А. А. Марковым в конце 1940-х – начале 1950-х годов. Нормальные алгоритмы (или алгорифмы, как называл их автор) являются определенными правилами преобразования слов в некотором алфавите. Дадим необходимые определения.

Определение. Пусть A – некоторый алфавит, α и β – слова над A. Выражения вида

$$\alpha \to \beta \text{ и } \alpha \to \cdot\beta$$

называются *продукциями*. Первая продукция называется *простой*, вторая – *заключительной*. Слово α называется *левой частью* продукции, слово β – *правой частью*.

Условимся, что запись $\{x\}$ означает символ x или его отсутствие.

Пусть γ – слово над A. Действие продукции $\alpha \to \{\cdot\}\beta$ на слово γ состоит в следующем. Находится первое вхождение слова α в γ (если такое вхождение имеется) и это вхождение слова α заменяется на слово β. Полученное в результате замены слово называется *результатом применения продукции* $\alpha \to \{\cdot\}\beta$ на слово γ. Если γ не содержит подслова α, то говорят, что продукция $\alpha \to \{\cdot\}\beta$ *неприменима* к слову γ.

Пример 1. В следующей таблице приведены примеры действия продукций на слова.

Исходное слово	Продукция	Результат
таблица	блиц → релк	тарелка
ааваава	ава → с	асава
парта	арт → р	пара
парта	т → ε	пара
та	ε → пар	парта

Определение. *Схемой* нормального алгоритма называется упорядоченная последовательность продукций.

Определение. Пусть Σ – схема нормального алгоритма. *Нормальным алгоритмом* (*алгоритмом Маркова*) *в алфавите A*

называется следующее правило построения последовательности слов. Пусть γ – слово над A. Полагаем $\gamma_0 = \gamma$. Пусть для некоторого $i \geq 0$ слово γ_i уже построено и процесс построения последовательности слов еще не завершился. Рассмотрим два случая:

1) в схеме нет продукций, левые части которых являются подсловами слова γ. Тогда γ_{i+1} полагаем равным γ_i и процесс построения считаем завершившимся.

2) в схеме существуют продукции, левые части которых являются подсловами слова γ. Тогда их них берется первая продукция и применяется к слову γ. Если эта продукция является простой, то процесс построения последовательности продолжается. Если же продукция является заключительной, то этот процесс завершается.

Если процесс построения завершается, то говорят, что алгоритм *применим* к слову γ. В этом случае последний член последовательности называется *результатом применения алгоритма* к слову γ.

Пример 2. Пусть $A = \{a, b\}$. Рассмотрим схему

$$\Sigma = (ab \to ba,\ b \to \cdot\varepsilon,\ a \to a).$$

Алгоритм, определяемый этой схемой, работает следующим образом. Если слово содержит буквы a и b, то он ставит все буквы b перед буквами a, т. е. выполняется первая продукция до тех пор, пока это возможно. После этого выполняется вторая продукция, алгоритм зачеркивает первую букву b и останавливается. Если слово содержит только b, то алгоритм выполняет вторую продукцию (так как первая неприменима), т. е. зачеркивает первую букву b и останавливается. К пустому слову и словам, содержащим только букву a, алгоритм не применим. □

Пример 3. Пусть $A = \{1\}$. Рассмотрим нормальный алгоритм в алфавите A со следующей схемой:

$$(11 \to \varepsilon,\ 1 \to \cdot\varepsilon,\ \varepsilon \to \cdot 1).$$

Этот алгоритм последовательно стирает по две единицы, пока не останется одна или ни одной. Если остается одна единица, то алгоритм стирает и ее и останавливается. Если же не остается ни одной, то он печатает единицу и останавливается. Этот алгоритм вычисляет функцию натурального аргумента

$$f(n) = \begin{cases} 1, \text{ если } n \text{ четно}, \\ \varepsilon, \text{ если } n \text{ нечетно}. \end{cases}$$

Здесь натуральное число n представлено в единичном коде. □

Определение. Пусть A – некоторый алфавит и f: $A^* \to A^*$. Функция f называется *нормально вычислимой*, если существует расширение B алфавита и нормальный алгоритм в алфавите B, который каждое слово v из области определения функции f перерабатывает в слово $f(v)$.

Пример 4. Пусть $A = \{0, 1\}$ и $f(x) = x + 1$, где аргумент x представлен в двоичном коде. В качестве алфавита B возьмем $\{0, 1, a, b\}$. Рассмотрим следующую схему

$$\Sigma = (0b \to \cdot 1,\ 1b \to b0,\ b \to \cdot 1,$$
$$a0 \to 0a,\ a1 \to 1a,\ 0a \to 0b,\ 1a \to 1b,\ \varepsilon \to a).$$

Применим алгоритм с данной схемой к слову $x = 10011$. Все продукции, кроме последней, к слову x неприменимы, поэтому применяется эта продукция и получается слово $a10011$. Применение четвертой и пятой продукций приведет к слову $10011a$. Теперь применима только предпоследняя продукция, получается слово $10011b$. Затем с помощью второй продукции «осуществляется поиск» самого правого нуля. К слову $100b11$ применяется первая продукция и алгоритм работу завершает. В итоге имеем равенство $f(10011) = 10111$. □

Пример 5. Составим схему нормального алгоритма, вычисляющего сложение ненулевых чисел в единичном коде. Пусть $A = \{1, *, a\}$ и

$$\Sigma = (a1 \to 1a,\ a* \to 1a,\ 1a \to \cdot\varepsilon,\ \varepsilon \to a).$$

Убедимся в том, что нормальный алгоритм с такой схемой слово 1^x*1^y перерабатывает в слово 1^{x+y}. Первой применяется последняя продукция схемы, так как к исходному слову первые три продукции неприменимы. Эта продукция в начале слова записывает букву a. Многократное применение первой продукции дает слово 1^xa*1^y. Затем применяется вторая продукция (поскольку первая неприменима), которая символ $*$ заменяет на 1. Применение третьей продукции завершает работу алгоритма. □

Напомним, что в § 3 мы ввели понятие словарной функции, вычислимой на машине Тьюринга.

Теорема 6. 12. Функция нормально вычислима тогда и только тогда, когда она вычислима на машине Тьюринга.

Доказывать эту теорему здесь не будем. С доказательством можно ознакомиться по книге [МН].

Кроме машин Тьюринга и Минского, рекурсивных функций и нормальных алгоритмов в литературе описаны и другие модели вычислений: операторные алгоритмы, машины Поста, машины с неограниченными регистрами и т.д. С этими моделями

можно познакомиться по книгам А.И.Мальцева или Н. Катленда.

Задачи

1. Программа машины Тьюринга состоит из следующих команд:

$$q_0a \to q_1mR,\ q_0\lambda \to q_3\lambda,\ q_0b \to q_2bR,$$
$$q_1b \to q_0bR,\ q_2m \to q_1mL,\ q_2a \to q_1bR,$$

где q_0 – начальное, q_3 – заключительное состояния, λ – пустой символ;

а) написать четыре последовательных конфигурации, если начальной является конфигурация q_0abba;

б) определить, остановится ли машина, начиная работу в конфигурации q_0abba;

в) определить, сколько тактов сделает машина до остановки, начиная работу в конфигурации q_0baba.

2. По заданной машине Тьюринга M и начальной конфигурации C найти заключительную конфигурацию, если она существует (q_0 – начальное, q_3 – заключительное состояния):

1)

	a	b	λ	m
q_0	q_1mR	q_1mR		
q_1	R	λR	q_2L	
q_2	L		aL	q_3a

а) $C_1 = q_0a^2baba$, б) $C_2 = q_0ba\lambda b$, в) $C_3 = q_0ba\lambda_2$.

2)

	a	b	λ
q_0	$q_1\lambda R$	$q_1\lambda R$	R
q_1	$q_2\lambda R$	q_3	
q_2	q_3	q_0R	

а) $C_1 = q_0a^2ba$, б) $C_2 = q_0ba^2b$, в) $C_3 = a^3b$.

3. Написать программу машины Тьюринга, переводящей конфигурацию C_0 в конфигурацию C_1:

а) $C_0 = q_0a^nb$, $C_1 = q_1a^nba^n$ ($n \geq 1$);

б) $C_0 = q_0a^nb^n$, $C_1 = q_1(ab)^n$ ($n \geq 1$);

в) $C_0 = \lambda a^n q_0 b^m$, $C_1 = \lambda b^m q_1 a^n$ (m, $n \geq 1$), где λ – пустой символ.

4. Нарисовать диаграммы машин Тьюринга из задач 1 и 2.

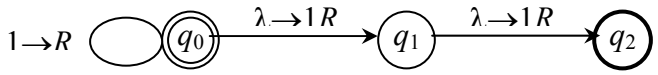
5. Показать, что машина Тьюринга, программа которой представлена в таблице,

	a	*b*	*m*	*	λ
q_1	q_2mR	q_3mR		q_3	
q_2	R	R		R	q_4aL
q_3	R	R		R	q_5bL
q_4	L	L	q_1aR	L	
q_5	L	L	q_1bR	L	

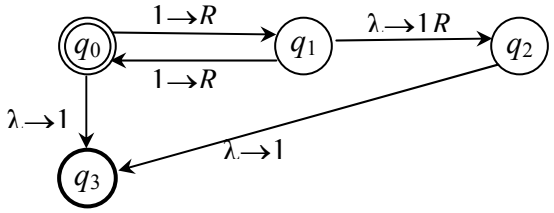
осуществляет копирование, т.е. перезапись слова $d*$ в слово $d*d$, где $d\in\{a, b\}^*$ и $d\neq\varepsilon$. Состояние q_1 – начальное, q_6 – заключительное, λ – пустой символ. Нарисовать диаграмму этой машины.

6. Какую одноместную функцию вычисляет машина Тьюринга со следующей диаграммой (аргумент задается в единичном коде):

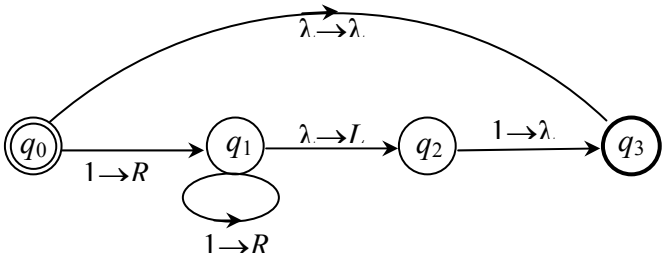
а)



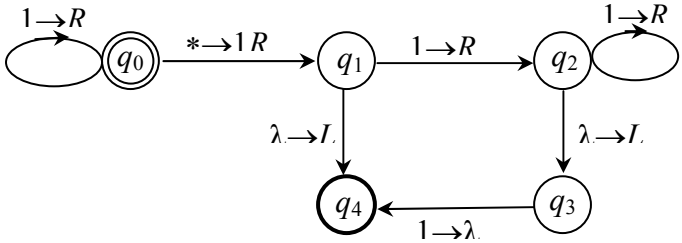
б)



в)



7. Какую двухместную функцию вычисляет машина Тьюринга со следующей диаграммой (аргументы задаются в единичном коде):



8. Написать программу машины Тьюринга, проверяющей правильную расстановку скобок [и].

9. Начертить диаграмму машины Тьюринга, правильно вычисляющей разность $x - y$.

10. Написать программы трехленточных машин Тьюринга, вычисляющих следующие функции (положение головок на лентах в момент остановки не имеет значения):

а) $x + y$, б) $min(x, y)$,
в) $|x - y|$, г) $x - y$,
д) $x \div y$, е) $x \cdot y$.

11. Доказать примитивную рекурсивность следующих функций:

а) $sign(x) = \begin{cases} 0, \text{ если } x = 0, \\ 1, \text{ если } x \neq 0; \end{cases}$ б) $x \div 1 = \begin{cases} 0, \text{ если } x = 0, \\ x - 1, \text{ если } x > 0; \end{cases}$

в) $x \div y = \begin{cases} 0, \text{ если } x < y, \\ x - y, \text{ если } x \geq y; \end{cases}$ г) $|x - y|$;
д) $min(x, y)$;
е) $max(x, y)$.

12. Составить схему нормального алгоритма, вычисляющего функцию $f(x) = x + 1$ в единичном коде.

Ответ: $(a1 \to \cdot 1,\ a \to \cdot 1,\ \varepsilon \to a)$.

13. Составить схему нормального алгоритма, вычисляющего функцию $f(x) = 2x$ в единичном коде.

Ответ: $(a1 \to 11a,\ a \to \cdot\varepsilon,\ \varepsilon \to a)$.

14. Составить схему нормального алгоритма, осуществляющего перевод из единичного кода ненулевого натурального числа в двоичный код. Для некоторого упрощения задачи можно считать, что входное слово имеет вид 1^x*.

Ответ: $(a1 \to 1b,\ b1 \to a,\ b* \to *0,\ a* \to *1,\ *1 \to \cdot 1,\ 1 \to a1)$.

Глава 7. Сложность алгоритмов

В предыдущей главе мы познакомились с задачами, для которых алгоритма решения не существует (поэтому такие задачи называются более солидно – проблемы). В этой главе мы также будем рассматривать различные задачи, причем очень много. В отличие от задач шестой главы, алгоритмы решения у этих задач есть. Все они могут быть решены полным перебором всех вариантов возможных ответов (множество вариантов будет конечным). Однако, как мы увидим позже, полный перебор практически неосуществим даже в простых случаях.

В первом параграфе этой главы будет приведен большой список задач. В этом списке есть примеры задач, для которых известны достаточно простые алгоритмы решения, другие же задачи решаются сложными алгоритмами. Основная идея последних алгоритмов состоит в том, чтобы найти более или менее эффективный способ ограничения перебора. На пути ограничения перебора есть некоторый прогресс. Однако на данный момент для таких задач не известны алгоритмы их решения в реальное время.

§ 1. Примеры задач

Как уже было сказано, в этом параграфе будет приведено много примеров задач. В основе каждой из них лежит некоторая математическая структура, чаще всего конечный граф. Для этих задач нет явной формулы для вычисления решения (ответа), решение (ответ) в принципе может быть получено некоторым перебором вариантов. Такие задачи часто называют *комбинаторными*. Комбинаторных задач известно очень много. Мы приведем те, которые представляют значительный практический интерес и которые используются в последующих параграфах.

Термин *задача* (или *массовая задача*) будет означать некоторый общий вопрос, на который надо дать ответ. Как правило, задача имеет несколько *параметров* (или *атрибутов*). Мы будем предполагать, что при формулировке (постановке) задачи выполняются следующие требования:

1) приводится условие задачи с указанием параметров и областей изменения параметров;

2) формулируются условия, которым должно удовлетворять решение (ответ) задачи (решений может быть несколько).

Индивидуальная задача получается из массовой задачи, если параметрам приданы конкретные значения из своих областей изменения.

В качестве примера приведем задачу о раскраске графа. Напомним ее формулировку. Дан обыкновенный граф G и число k. Определить, можно ли раскрасить вершины графа G k красками так, чтобы смежные вершины были раскрашены разными красками. Часто в случае положительного ответа требуется дополнительно привести хотя бы один вариант раскраски. Здесь параметрами являются обыкновенный граф G и число k. Условие, которому должно удовлетворять решение: смежные вершины раскрашиваются разными красками. Сформулированная задача является массовой. На рисунке 7.1 приведен пример двух индивидуальных задач.

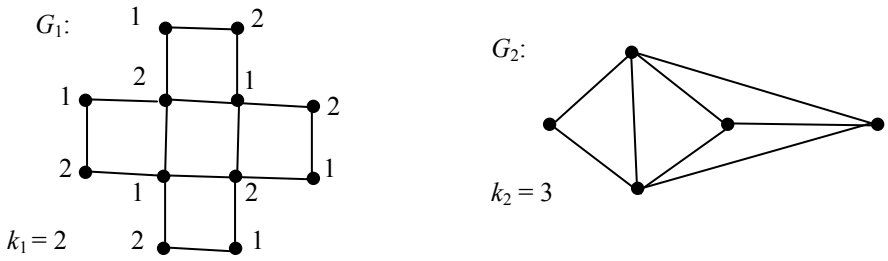


Рис. 7.1

Для первой задачи ответ – «да». Один из вариантов необходимой раскраски обозначен цифрами у вершин графа. В случае второй задачи ответ – «нет».

Список задач начнем с задачи, практическая важность которой не должна вызывать сомнений.

1. «*Сортировка*». Дана последовательность элементов a_1, a_2, …, a_n линейно упорядоченного множества. Найти перестановку индексов π: $\{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ такую, что

$$a_{\pi(1)} \le a_{\pi(2)} \le \ldots \le a_{\pi(n)}.$$

2. «*Эйлеров цикл*». Дан обыкновенный граф G. Определить, существует ли в графе G цикл, проходящий по всем ребрам графа (в точности по одному разу). [В случае положительного ответа выдать один такой цикл].

3. «*Эйлеров контур*». Дан ориентированный граф G. Определить, существует ли в графе G контур, проходящий по всем

дугам графа (в точности по одному разу). [В случае положительного ответа выдать один такой контур].

4. «*Каркас наименьшего веса*». Дан обыкновенный связный граф G, каждому ребру которого поставлено в соответствие положительное действительное число – *вес* ребра. (Такие графы называются *взвешенными*.) Найти подграф T этого графа, удовлетворяющий условиям:

1) T содержит все вершины графа G, т. е. получается из G удалением некоторых ребер;

2) T – связный граф;

3) суммарный вес ребер графа T является наименьшим (среди всех подграфов, удовлетворяющих первым двум условиям).

Заметим, что в силу того, что G имеет конечное множество вершин и ребер, такой подграф T всегда существует (но, как правило, не один). Он называется *каркасом* (или *остовом*) графа G.

5. «*Транзитивное замыкание*». Дан ориентированный граф. Найти транзитивное замыкание этого графа.

Напомним, что *транзитивным замыканием* орграфа G называется орграф H с тем же множеством вершин, что и у графа G, удовлетворяющий условию:

в графе H из вершины x в вершину y ($x \neq y$) идет дуга тогда и только тогда, когда в графе G существует путь из x в y.

Прежде, чем формулировать следующую задачу, напомним ряд определений. *Паросочетанием* обыкновенного графа называется множество ребер, никакие два различных ребра которого не имеют общей инцидентной вершины. Паросочетание называется *максимальным*, если при добавлении нового ребра расширенное множество ребер перестает быть паросочетанием. Паросочетание называется *наибольшим*, если оно имеет наибольшее количество ребер среди всех паросочетаний данного графа.

6. «*Паросочетание*». Дан обыкновенный граф G и натуральное число k. Определить, существует ли в G паросочетание мощности k. [В случае положительного ответа выдать одно такое паросочетание.] У задачи есть варианты оптимизационного характера: «*максимальное паросочетание*» – выдать хотя бы одно максимальное (по включению) паросочетание, «*наибольшее паросочетание*» – выдать хотя бы одно наибольшее (по мощности) паросочетание.

Как и в случае предыдущей задачи, перед постановкой очередной задачи приведем ряд определений.

Сетью называется тройка (V, E, c), где $G = (V, E)$ – ориентированный граф, c – функция из множества дуг E в множество неотрицательных действительных чисел $\boldsymbol{R}^*$. Если e – дуга графа G, то величина $c(e)$ называется *весом* дуги e.

Степенью исхода $\rho^-(v)$ вершины v называется сумма весов дуг, выходящих из v. *Степенью захода* $\rho^+(v)$ вершины v – сумма весов дуг, заходящих в v. Вершина v называется *источником*, если $\rho^+(v) = 0$, $\rho^-(v) > 0$. Вершина v называется *стоком*, если $\rho^+(v) > 0$, $\rho^-(v) = 0$.

Пусть (V, E, c) – сеть, имеющая один источник a и один сток b. Функция φ: $E \to \boldsymbol{R}^*$ называется потоком через сеть, если выполнены условия:

1) $\varphi(e) \le c(e)$ для любой дуги e;

2) в сети (V, E, φ) для любой вершины v, кроме a и b, выполняется равенство $\rho^+(v) = \rho^-(v)$.

Обратим внимание на то, что во втором условии вместо исходных весов дуг взяты значения функции φ.

Пусть φ – поток через сеть (V, E, c), a и b – источник и сток соответственно. Величина $\rho^-(a)$ в сети (V, E, φ) (или что то же самое величина $\rho^+(b)$) называется *величиной потока*. Поток называется *максимальным*, если он имеет наибольшую величину.

7. «*Максимальный поток*». Дана сеть. Найти максимальный поток через эту сеть.

Имеется ввиду, что надо найти хотя бы один максимальный поток, так как таких потоков может быть несколько.

8. «*Планарность*». Дан обыкновенный граф. Определить, будет ли он планарным.

Напомним, что граф называется *планарным*, если его можно расположить на плоскости без пересечений ребер.

9. «*Выполнимость*». Дана формула F логики высказываний, имеющая конъюнктивную нормальную форму. Определить, выполнима ли формула F, т. е. существует ли интерпретация φ такая, что $\varphi(F) = 1$. [В случае положительного ответа выдать такую интерпретацию.]

10. «*Независимое множество*». Дан обыкновенный граф и натуральное число k. Определить, существует ли подмножество вершин мощности k, никакие две вершины которого несмежны. [В случае положительного ответа выдать такое множество.]

Как и задача «паросочетание» эта задача имеет варианты: «*максимальное* (по включению) *независимое множество*» и «*наибольшее* (по мощности) *независимое множество*».

11. «*Полное множество* (или *клика*)». Дан обыкновенный граф и натуральное число k. Определить, существует ли подмножество вершин мощности k, любые две различные вершины которого смежны. [В случае положительного ответа выдать такое множество.]

Варианты задачи: «*максимальное* (по включению) *полное множество*» и «*наибольшее* (по мощности) *полное множество*».

Прежде, чем сформулировать очередную задачу, приведем определение. Множество вершин U обыкновенного графа $G = (V, E)$ называется *доминирующим*, если для любой вершины $u \in V \backslash U$ существует смежная с ней вершина $v \in U$.

12. «*Вершинное покрытие*». Дан обыкновенный граф G и натуральное число k. Определить, имеется ли в графе G вершинное покрытие мощности k. [В случае положительного ответа выдать вершинное покрытие мощности k.]

Варианты задачи: «*минимальное вершинное покрытие*» и «*наименьшее вершинное покрытие*».

Подмножество U множества вершин графа G называется *вершинным покрытием* этого графа, если любое ребро графа инцидентно хотя бы одной вершине из U.

Формулировку следующей задачи начнем с определения. Пусть S – непустое семейство подмножеств непустого множества M. Подсемейство S' семейства S называется *покрытием множества* M, если объединение множеств, принадлежащих S', равно M.

13. «*Покрытие множествами*». Дано непустое семейство S подмножеств непустого множества M и натуральное число k. Определить, существует ли подсемейство S' мощности k семейства S, которое покрывает M. [В случае положительного ответа выдать такое подсемейство.]

Имеется варианты: «*минимальное покрытие множествами*» и «*наименьшее покрытие множествами*».

14. «*Гамильтонов цикл*». Дан обыкновенный граф G. Определить, имеет ли G гамильтонов цикл, т. е. цикл, проходящий по каждой вершине в точности по одному разу. [В случае положительного ответа выдать такой цикл.]

Следующая задача является обобщением задачи «*гамильтонов цикл*».

15. «*Коммивояжер*». Дан обыкновенный взвешенный граф G, т. е. обыкновенный граф, каждой дуге которого поставлено в соответствие положительное действительное число – вес дуги.

Найти в графе G гамильтонов цикл наименьшего суммарного веса (если граф G имеет гамильтонов цикл.

Довольно часто задача «коммивояжер» формулируется в более общем смысле: найти цикл, проходящий по всем вершинам и имеющий наименьший суммарный вес (т. е. снимается условие, что цикл проходил через вершину в точности по одному разу).

16. «*Гамильтонов контур*». Дан ориентированный граф G. Определить, имеет ли G гамильтонов контур, т. е. контур, проходящий по каждой вершине в точности по одному разу. [В случае положительного ответа выдать такой контур.]

Задача, которой мы завершаем список, фактически сформулирована выше. Но для полноты картины повторим ее формулировку.

17. «*Раскраска*». Дан обыкновенный граф G и натуральное число k. Определить, можно ли граф G правильно раскрасить k красками. [В случае положительного ответа выдать такую раскраску.]

Задача имеет и оптимизационный вариант: найти наименьшее число красок, которыми можно правильно раскрасить данный граф. Это вариант называется «*хроматическое число*». Хроматическое число графа G будем обозначать через $\chi(G)$.

§ 2. Задачи и языки

Во введении к данной главе уже отмечалось, что значительная часть главы будет посвящена изучению практически значимых задач, для которых в настоящее время не известны алгоритмы, решающие эти задачи в реальное время. В этой фразе термины «задача», «алгоритм», «реальное время» понимаются в интуитивном смысле. Для второго термина у нас есть формальный аналог – машина Тьюринга. В этом параграфе мы дадим формальный аналог первого термина, формальный аналог последнего термина будет приведен в следующем параграфе.

Проанализируем примеры задач из первого параграфа с точки зрения структуры их ответов. У многих задач предполагается простой ответ: «да» или «нет». Это, например, задачи «*планарность*», «*полное множество*», «*независимое множество*», «*раскраска*», если в формулировках этих задач убрать то, что записано в квадратных скобках. Такие задачи мы будем называть *задачами распознавания*.

Если же в задаче, кроме ответа «да» или «нет», требуется в случае «да» выдать одно из решений, то такие задачи будем называть *задачами распознавания с ответом*. В качестве примеров таких задач можно привести те же, что и в предыдущем абзаце, если в формулировках этих задач оставить то, что записано в квадратных скобках.

Формулировки некоторых задач содержат условия оптимизационного характера. В этих задачах требуется найти максимальное, наибольшее, минимальное или наименьшее в каком-то смысле решение. Таковы, например, задачи «*каркас наименьшего веса*», «*максимальное независимое множество*», «*наибольшее независимое множество*», «*хроматическое число*». Задачи такого типа будем называть *оптимизационными*.

Сказанное в предыдущих абзацах показывает, что формализация понятия «задача» – довольно сложное дело. В литературе, посвященной анализу сложности алгоритмов, обычно рассматриваются только задачи распознавания. Это вызвано, во-первых, тем, что такие задачи допускают довольно простую формализацию (что будет сделано в следующем абзаце). Кроме того, одна из целей этой главы – дать представление о методах оценки трудности решения задачи. Интуитивно понятно, и это во-вторых, что задача распознавания не труднее соответствующей оптимизационной задачи и, тем более, задачи распознавания с ответом. Предположим, например, что у нас есть алгоритм решения задачи «*раскраска*» в оптимизационной форме, т. е. алгоритм вычисления хроматического числа данного графа. Тогда, чтобы решить задачу «*раскраска*» как задачу распознавания, т. е. узнать, можно ли данный граф G правильно раскрасить k красками, достаточно сравнить два числа k и $\chi(G)$. Следовательно, если задача распознавания относится к трудно решаемым в каком-то смысле задачам, то и соответствующие оптимизационные задачи и задачи распознавания с ответом надо тоже отнести к трудно решаемым.

Задачи распознавания мы будем сводить к *задачам распознавания принадлежности к языку*. Пусть L – язык над алфавитом Σ. Задача распознавания принадлежности к языку L состоит в том, чтобы по данной цепочке $w \in \Sigma^*$ определить, принадлежит ли она языку L. В случае задачи о раскраске в k цветов графа G, сводимость к языку может быть проведена следующим образом. Вершины графа G будем именовать натуральными числами, а ребра – парами чисел. Числа будем записывать в десятичном коде. Например, задача о раскраске графа, изображен-

ного на рис. 7.2, в четыре цвета может быть представлена цепочкой

4; (1; 2); (1; 3); (2; 3); (2; 4); (2; 5); (3; 4); (3; 5)

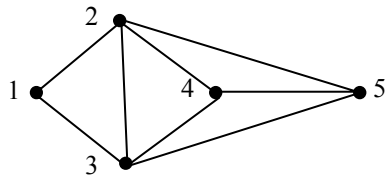


Рис. 7.2

в алфавите $\Sigma = \{0, 1, \ldots, 9, ;, (,)\}$. Язык L состоит из всех таких цепочек, для которых задача «раскраска» имеет положительное решение. В частности, написанная выше цепочка принадлежит языку L. Подчеркнем, что цепочки из $\Sigma^* \backslash L$ либо вообще не представляют индивидуальную задачу о раскраске, либо представляют эту задачу с отрицательным решением.

В только что приведенном примере мы применили некоторую «схему кодирования». Схема кодирования единственной не является. Ту же задачу «раскраска» можно свести к задаче распознавания принадлежности другому языку L_1. Для этого вершины графа пронумеруем натуральными числами и граф G представим матрицей смежности M. Тогда индивидуальной задаче о раскраске в k красок можно поставить в соответствие следующую цепочку в алфавите $\Sigma_1 = \{0, 1, *\}$. Сначала записывается число k в двоичном коде, затем ставится разделитель $*$, а после этого по строкам записывается матрица M, строки матрицы разделяются символом $*$. Так, индивидуальная задача о раскраске в 3 цвета графа, изображенного на рис. 7.3, будет представлена цепочкой

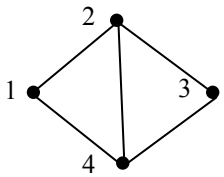


Рис. 7.3

$$101*0101*1011*0101*1110.$$

Приведенные две кодировки задачи о раскраске мы, следуя [ГД], будем называть «*разумными*». Определять, что такое «разумность» кодировки мы здесь не будем (по поводу определения «разумности» кодировки см. [ГД, стр. 36]), а будем предполагать выполнимость следующих неформальных условий:

1) кодировка естественным образом вытекает из условия задачи,

2) подобные способы задания математических структур, используемых в задаче, ранее использовались на практике.

И при той, и при другой кодировке длина получившейся цепочки не превосходит cn^2, где n – число вершин графа, а – c некоторая константа. Кроме того, у этих кодировок есть более важное общее свойство. Все известные на данный момент алгоритмы при обеих кодировках решают задачу о раскраске обыкновенного графа за время, не меньшее, чем $d2^l$ (о том, что же

такое «время решения задачи» – чуть позже), где l – длина цепочки, кодирующей индивидуальную задачу, d – некоторая положительная константа.

Приведем пример «неразумной» кодировки. Пусть граф G имеет n вершин и задается матрицей смежности $M = (m_{ij})$. Задачу о раскраске в k цветов закодируем в алфавите $\Sigma = \{0, 1, *\}$ следующим образом:

<число k в двоичном коде>$*m_{11}$<2^n символов *>

m_{12}<2^n символов *> m_{13}<2^n символов *> и т. д.

Для такой кодировки нетрудно придумать алгоритм, решающий ее за линейное время, т. е. за время cl, где l – длина цепочки, кодирующей индивидуальную задачу, c – некоторая константа.

§ 3. Класс *P-time*

В начале этого параграфа мы определимся с «вычислительным устройством», которое будет решать задачи, и выберем числовые характеристики этого устройства.

В качестве вычислительного устройства возьмем машину Тьюринга, но уточним ее определение следующим образом. Во-первых, будем считать, что зафиксировано непустое подмножество A входного алфавита Σ, которое не содержит пустой символ. Во-вторых, машина будет иметь два заключительных состояния q_y и q_n («*yes*» и «*no*») и будет останавливаться, только приходя в одно из них. Начинать работу машина Тьюринга будет так же, как и раньше, т. е. находясь в начальном состоянии и обозревая первую непустую ячейку ленты. Пусть $w \in \Sigma^*$. Будем говорить, что машина *начинает работу на цепочке* w, если цепочка w записана на входной ленте, остальные ячейки пусты, и машина начинает работу так, как сказано выше. Будем говорить также, что машина *заканчивает работу на цепочке* w, если она приходит в одно из заключительных состояний и цепочка входной ленты, начинающаяся первым непустым символом и заканчивающаяся последним непустым символом, есть цепочка w.

Напомним, что мы рассматриваем задачи распознавания, которые сведены к задачам распознавания принадлежности к языку. Надо определиться, в каком смысле машина распознает язык. В принципе, имеются два варианта определения.

Определение (вариант 1). Машина Тьюринга M *распознает язык* L над Σ, если для любой цепочки $w \in \Sigma^*$, на которой M начинает работу, выполняется одно из двух условий:

1) машина заканчивает работу в состоянии q_y, если $w \in L$;

2) машина заканчивает работу в состоянии q_n, если $w \notin L$.

Определение (вариант 2). Машина Тьюринга M *распознает язык* L над Σ, если она, начиная работу на цепочке $w \in \Sigma^*$, заканчивает ее в состоянии q_y тогда и только тогда, когда $w \in L$.

Подчеркнем, что во втором варианте определения если $w \notin L$, то машина, либо останавливается в состоянии q_n, либо работает бесконечно.

Ясно, что в первом варианте определения распознаваемыми будут рекурсивные языки, во втором – рекурсивно перечислимые. По причинам, которые будут указаны позднее, в *качестве основного варианта выберем второй вариант.*

К понятию распознаваемости языка машиной Тьюринга можно подойти и несколько по-другому.

Определение. Цепочка w *распознается* машиной M, если M, начиная работу на w, останавливается в состоянии q_y (состояние входной ленты на момент остановки значения не имеет).

Используя только что приведенное определение, второй вариант определения распознаваемости языка можно сформулировать и так: языком, распознаваемым машиной Тьюринга, называется множество всех цепочек, распознаваемых этой машиной.

Введем важную числовую характеристику машины Тьюринга, которая называется временной сложностью. В содержательном плане временная сложность машины M – это функция (мы будем ее обозначать через T_M) такая, что всякая распознаваемая машиной M цепочка длины n может быть распознана ею не более, чем за $T_M(n)$ тактов работы (такт работы – выполнение одной команды). Приведенную фразу можно, в принципе, взять за определение временной сложности. Однако эта фраза нуждается в уточнении. Во-первых, получается, что если – временная сложность, то и всякая большая, чем T_M, функция также является временной сложностью. Такая ситуация не всегда удобна. Во-вторых, несложно привести пример языка L, который не содержит цепочек некоторой длины m. Тогда если машина M распознает язык L в приведенном выше смысле, то значение функции $T_M(m)$ не определено. Нам будет удобно в этом случае считать, что $T_M(m) = 1$.

Введем обозначение: $t_M(x)$ – число тактов работы машины M на цепочке x до остановки. Если M не останавливается, начиная работу на x, то $t_M(x) = \infty$.

Учитывая сказанное в предыдущем абзаце, приведем

Определение. *Временной сложностью* машины Тьюринга M называется функция T_M: $\boldsymbol{N} \to \boldsymbol{N}$, определяемая равенством:

$$T_M(n) = max\{\{1\} \cup \{t_M(w) \mid w \in L \text{ и } |w| = n\}.$$

Используя это определение, введем основное понятие этого параграфа.

Определение. Язык L называется *полиномиальным*, если существует машина Тьюринга M, распознающая язык L, и полином p, такие, что $T_M(n) \le p(n)$ для всех $n \in \boldsymbol{N}$.

Полиномиальные языки мы будем содержательно воспринимать, как легко распознаваемые языки.

Вернемся к тому моменту, когда мы доопределили функцию T_M равенством $T_M(n) = 1$ в случае, когда язык L не содержит цепочек длины n. Рассмотрим в качестве L множество четных чисел (представленных, скажем, в десятичном коде). Если не доопределять функцию T_M указанным равенством, то язык L не будет полиномиальным, так как неравенство $T_M(n) \le p(n)$ не будет выполняться для нечетных чисел. Это противоречит содержательному восприятию множества четных чисел, как легко распознаваемого языка.

Еще раз напомним, что задачу распознавания мы свели к задаче распознавания принадлежности к языку. *Размером индивидуальной задачи* мы будем называть длину соответствующей цепочки языка. Задачу будем называть *полиномиальной*, если полиномиальным является соответствующий язык.

Из списка задач первого параграфа задачи 1 – 8 являются полиномиальными. Укажем оценки временной сложности этих задач и ссылки, где можно ознакомиться с соответствующим доказательством. Пусть $f(n)$ – всюду определенная функция из $\boldsymbol{N}$ в $\boldsymbol{N}$.Здесь и далее мы будем пользоваться обозначением $O(f(n))$ для класса функций g: $\boldsymbol{N} \to \boldsymbol{N}$ таких, что

$$g(n) \le cf(n)$$

для всех натуральных чисел n и некоторой положительной константы c.

1. «*Сортировка*», $O(nlogn)$, [АБР, стр. 156].
2. «*Эйлеров цикл*», $O(n^2)$, [АБР, стр. 179].
3. «*Эйлеров контур*», $O(n^2)$, [АБР, стр. 179].
4. «*Каркас наименьшего веса*», $O(n^2logn)$, [АБР, стр. 184].
5. «*Транзитивное замыкание*», $O(n^3)$, [АБР, стр. 191].
6. «*Максимальное паросочетание*», $O(n^{5/2})$, [АБР, стр. 243].
7. «*Максимальный поток*», $O(n^5)$, [АБР, стр. 226].
8. «*Планарность*», $O(n)$, [Л, стр. 202].

Введенная выше числовая характеристика работы машины M (функция $T_M(n)$) не является единственной числовой характеристикой работы машины M.

Во-первых, временная сложность была определена нами как *временная сложность в худшем случае*. (Вспомним, что в определении функции $T_M(n)$ берется максимум по $t_M(w)$.) Кроме нее, изучается *временная сложность в среднем*. Она определяется следующим образом

$$T^*_M(n) = max[\{1\} \cup (t_M(w_1) + t_M(w_2) + \ldots + t_M(w_p)/p],$$

где w_1, w_2, …, w_p – все (попарно различные) цепочки языка L длины n. На практике чаще всего используется временная сложность в худшем случае (т. е. та, которую мы назвали просто временная сложность). Однако возможны ситуации, когда полезной оказывается временная сложность в среднем. Допустим, что пользователю надо решить большое количество индивидуальных задач, принадлежащих одной и той же массовой задаче и имеющих одну и ту же длину. В этом случае временная сложность в среднем дает большую информацию о суммарном времени решения всех задач, чем временная сложность в худшем случае. Мы будем рассматривать только функцию $T_M(n)$. О временной сложности в среднем можно прочитать в [АХУ].

Во-вторых, кроме временной сложности рассматривается так называемая емкостная сложность. Формальных определений здесь давать не будем (так как емкостную сложность в дальнейшем не будем рассматривать). Скажем только, что емкостная сложность – число просмотренных машиной ячеек в процессе работы. Изучаются емкостная сложность в среднем и емкостная сложность в худшем случае. Подробнее об этом можно прочитать в [АХУ].

§ 4. Полиномиальная сводимость

Начнем с определения полиномиальной сводимости одного языка к другому. Для этого нам понадобятся машины Тьюринга, которые останавливаются на любой цепочке из Σ^*.

Определение. Пусть Σ_1 и Σ_2 – непустые подмножества входного алфавита машины M, не содержащие пустой символ, а f – всюду определенная функция из Σ^*_1 в Σ^*_2. Мы будем говорить, что машина M *вычисляет функцию* f, если M, начиная работу на цепочке $w \in \Sigma^*_1$, заканчивает ее на цепочке $f(w) \in \Sigma^*_2$.

Другими словами, $f(w)$ – цепочка на входной ленте на момент остановки машины M, начинающаяся первым непустым символом и заканчивающаяся последним непустым символом.

Определение. Пусть Σ_1 и Σ_2 – непустые подмножества входного алфавита машины M, не содержащие пустой символ, L_1 – язык над Σ_1, L_2 – язык над Σ_2. Будем говорить, что M *полиномиально сводит* язык L_1 к языку L_2, если для любой цепочки $w \in \Sigma_1^*$ выполняются условия:

1) $w \in L_1$ тогда и только тогда, когда $f(w) \in L_2$;

2) существует полином $p(n)$ такой, что выполняется неравенство

$$max[\{1\}\} \cup \{t_M(w) \mid w \in \Sigma_1^* \text{ и } |w| = n\} \le p(n).$$

Напомним, что – $t_M(w)$ число тактов работы машины M на цепочке w до остановки и что M останавливается на любой цепочке из Σ_1^*.

Разумеется, и здесь (т. е. для машин, вычисляющих словарные функции) можно было бы ввести временную сложность работы машины, но нетрудно привести примеры, показывающие, мы получили бы функцию, отличающуюся от уже введенной функции $T_M(n)$.

Естественно, мы будем говорить, что одна задача полиномиально сводится к другой, если язык, соответствующий первой задаче, полиномиально сводится к языку, соответствующему второй задаче.

Приведем примеры полиномиально сводимых задач. Начнем с фактически очевидного примера.

Пример 1. Задача «*полное множество*» полиномиально сводится к задаче «*независимое множество*».

Пусть дан обыкновенный граф G и число k. Рассмотрим дополнение H графа G. Напомним, что дополнением обыкновенного графа называется граф с тем же множеством вершин, что и исходный граф. Две (различные) вершины смежны в дополнении тогда и только тогда, когда они не смежны в данном графе. Легко видеть, что множество вершин W является полным в графе G тогда и только тогда, когда это же множество является независимым в графе H. Рис 7.4 иллюстрирует это утверждение для $k = 3$, на рисунке пунктиром обведено множество W. осталось отметить, что H можно получить из G за время $O(n^2)$, где n – число вершин графа G. □

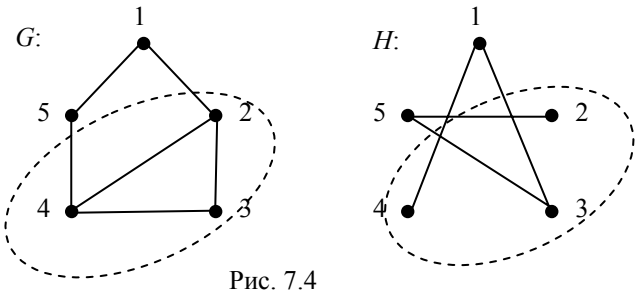


Рис. 7.4

Определение. Два языка называются *полиномиально эквивалентными*, если они полиномиально сводятся один к другому.

Понятие полиномиальной эквивалентности известным нам способом перенесем на задачи. Из рассмотрения предыдущего примера следует, что задачи «*полное множество*» и «*независимое множество*» полиномиально эквивалентны.

Теорема 7.1. Задача «*выполнимость*» полиномиально сводима к задаче «*полное множество*» («*клика*»).

Доказательство. Дана формула F, имеющая конъюнктивную нормальную форму:

$$F = D_1 \,\&\, D_2 \,\&\, \ldots \,\&\, D_p,$$

где D_1, D_2, …, D_p – элементарные дизъюнкции (или дизъюнкты). Рассмотрим обыкновенный граф G, вершинами которого являются пары (D, L), где D – дизъюнкт формулы F, а L – литерал этого дизъюнкта. Две вершины графа (D, L) и (D', L') смежны (соединены ребром), если $D \neq D'$ и множество $\{L, L'\}$ выполнимо. Последнее означает, что $\{L, L'\} \neq \{X, \neg X\}$ для любой атомарной формулы X.

Длиной формулы F (индивидуальной задачи) будем считать сумму числа литералов, входящих в дизъюнкты формулы F. Тогда ясно, что граф G можно получить за время, полиномиально зависящее от длины формулы F.

Докажем, что граф G содержит полное множество из p вершин тогда и только тогда, когда формула F выполнима.

Пусть U – полное множество графа G, содержащее p вершин. Тогда

$$U = \{(D_1, L_1), (D_2, L_2), \ldots, (D_p, L_p)\},$$

для некоторых литералов $L_1, L_2, \ldots, L_p$. Рассмотрим множество литералов $M = \{L_1, L_2, \ldots, L_p\}$. По построению графа G любое двухэлементное подмножество этого множества выполнимо. Убедимся в том, что все множество U выполнимо. Действительно, множество $\{L_1, L_2\}$ выполнимо, так как вершины (D_1, L_1) и (D_2, L_2) смежны. Определим интерпретацию φ так, что $\varphi(L_1) = 1$ и $\varphi(L_2) = 1$. (Подчеркнем, что φ определена только на атомарных формулах из L_1 и L_2.) Если $\varphi(L_3) = 0$, то одно из множеств $\{L_1, L_3\}$ или $\{L_2, L_3\}$ состоит из атомарной формулы и ее отрицания. Это противоречит тому, что вершины (D_1, L_1), (D_2, L_2) и (D_3, L_3) попарно смежны. Следовательно, $\varphi(L_3) = 1$ или $\varphi(L_3)$ не определено. В последнем случае доопределяем $\varphi(L_3)$ так, что бы выполнялось равенство $\varphi(L_3) = 1$. Расширим указанным способом интерпретацию φ на все множество M. Получим интерпретацию φ, при которой все литералы $L_1, L_2, \ldots, L_p$ истинны. Это

означает, что $\varphi(F) = 1$, так как литерал L_i содержится в дизъюнкте D_i для $1 \le i \le p$. Итак, если граф G содержит полное множество из p вершин, то формула F выполнима.

Возьмем теперь интерпретацию φ, для которой выполняется равенство $\varphi(F) = 1$. Тогда $\varphi(D_1) = \varphi(D_2) = \ldots = \varphi(D_p) = 1$. Из равенства $\varphi(D_i) = 1$ следует, что $\varphi(L_i) = 1$ для некоторого литерала L_i дизъюнкта D_i, где $1 \le i \le p$. Следовательно, множество вершин $\{(D_1, L_1), (D_2, L_2), \ldots, (D_p, L_p)\}$ графа G является полным. Из выполнимости формулы F следует существование p-элементного полного множества графа G. □

Приведем пример.

Пример 2. Доказательство теоремы 7.1 проиллюстрируем на примере формулы

$$F = (X_1 \vee \neg X_2 \vee X_3) \,\&\, (\neg X_1 \vee X_2) \,\&\, (\neg X_1 \vee \neg X_3).$$

Пусть $D_1 = X_1 \vee \neg X_2 \vee X_3$, $D_2 = \neg X_1 \vee X_2$ и $D_3 = \neg X_1 \vee \neg X_3$. Граф G, соответствующий этой формуле, изображен на рис. 7.5.

Рассмотрим интерпретацию $\varphi(X_1) = 0$, $\varphi(X_2) = 1$, $\varphi(X_3) = 1$.

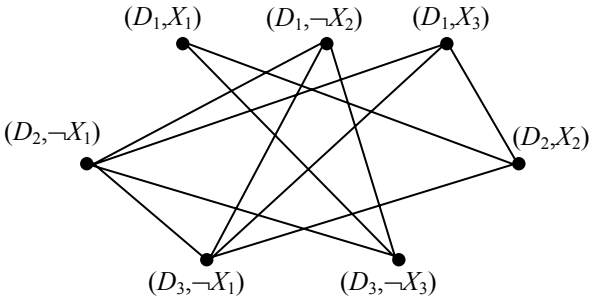


Рис. 7.5

Легко видеть, что $\varphi(F) = 1$, и что $\{(D_1, X_3), (D_2, \neg X_1), (D_3, \neg X_1)\}$ – полное трехэлементное множество графа G. С другой стороны, рассмотрим множество вершин $U = \{(D_1, \neg X_2), (D_2, \neg X_1), (D_3, \neg X_3)\}$ графа G. Множество U является полным, содержит три элемента и определяет интерпретацию ψ: $\psi(X_1) = \psi(X_2) = \psi(X_3) = 0$, при которой выполняется равенство $\psi(F) = 1$. □

Теорема 7.2. Задача «*полное множество*» («*клика*») полиномиально сводится к задаче «*вершинное покрытие*».

Доказательство. Пусть дан обыкновенный граф $G = (V, E)$. Через $H = (V, F)$ обозначим дополнение графа G (в классе обыкновенных графов). Напомним, что две вершины x и y в графе H соединены ребром тогда и только тогда, когда эти вершины в графе G ребром не соединены. Подчеркнем, что графы G и H имеют одно и то же множество вершин.

Докажем, что непустое собственное подмножество U множества V является полным в графе G тогда и только тогда, когда множество $V \backslash U$ является вершинным покрытием графа H.

Предположим, что множество U является полным в графе G, т. е. любые две различные вершины множества U в графе G соединены ребром. Возьмем произвольное ребро (z, w) графа H. Обе вершины z и w принадлежать U не могут по определению дополнения. Следовательно, хотя бы одна из них принадлежит множеству $V\backslash U$. Мы доказали, что если U – полное множество графа G, то $V\backslash U$ – вершинное покрытие графа H.

Докажем обратное утверждение. Пусть $V\backslash U$ – вершинное покрытие графа H. Возьмем различные вершины x и y из U. Если они в G несмежны, то по определению дополнения они смежны в H. Но тогда одна из этих вершин принадлежит $V\backslash U$, так как $V\backslash U$ – вершинное покрытие графа H. Следовательно, x и y смежны в H. Мы доказали, что если $V\backslash U$ – вершинное покрытие графа H, то U – полное множество графа G.

Итак, для того, чтобы узнать, имеет ли обыкновенный граф G с n вершинами полное множество мощности k $(0 < k < n)$ достаточно выяснить, имеет ли дополнение графа G вершинное покрытие мощности $n - k$. Мы уже отмечали, что дополнение графа может быть построено за время, не превосходящее cn^2 для некоторой константы c. □

Пример 3. Рисунок 7.6 иллюстрирует доказательство теоремы 7.2.

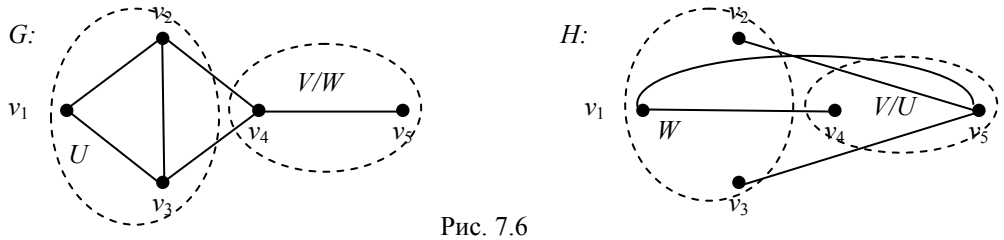


Рис. 7.6

На рисунке изображен граф G и его дополнение H. В качестве U возьмем полное в графе G множество $\{v_1, v_2, v_3\}$. Мы видим, что его дополнение до множества всех вершин, т. е. множество V/U, является вершинным покрытием графа H. Если теперь в графе H взять вершинное покрытие $W = \{v_1, v_2, v_3\}$, то его дополнение до V будет полным множеством в G. □

Теорема 7. 3. Задача «*вершинное покрытие*» полиномиально сводится к задаче «*гамильтонов контур*».

Доказательство. Предположим, что дан обыкновенный граф $G = (V, E)$ и натуральное число k. Построим ориентированный граф $H = (W, F)$. Введем вначале ряд обозначений. Пусть $V = \{v_1, v_2, \ldots, v_n\}$, а e_{ij} – ребро графа G, инцидентное вершинам v_i и v_j. Так как G – неориентированный граф, выполняется равенство $e_{ij} = e_{ji}$. Для каждой вершины графа G упорядочим ребра,

инцидентные этой вершине. Пусть d_{i1}, d_{i1}, …, d_{iri} – список ребер, инцидентных вершине $v_i \in V$.

Граф H построим следующим образом. Множество вершин W будет содержать по четыре вершины для каждого ребра графа G и новые вершины a_1, a_2, …, a_k. Более точно,

$W = \{ a_1, a_2, \ldots, a_k \} \cup$

$\cup \{(v, e, b) \mid v \in V, e \in E, b \in \{0, 1\}$ и e инцидентно $b\}$.

Дуги графа H, т. е. элементы множества F, будут двух типов. Первый тип – дуги, соединяющие четыре вершины, инцидентные ребру e_{ij} так, как показано на рис 7.7. Дуги второго типа получаются следующим образом. Из каждой вершины a_1, a_2, …, a_k проведем дугу во все вершины вида $(v_i, d_{i1}, 0)$. (Напомним, что d_{i1} – первое ребро в списке ребер, инцидентных вершине v_i.) Из вершины $(v_i, d_{ij}, 1)$ проводим дугу в вершину $(v_i, d_{il}, 1)$, если ребро e_{il} непосредственно следует за e_{ij} в списке ребер вершины v_i. Если же ребро e_{ij} является последним в этом списке, то из вершины $(v_i, d_{ij}, 1)$ проводим дуги во все вершины a_1, a_2, …, a_k.

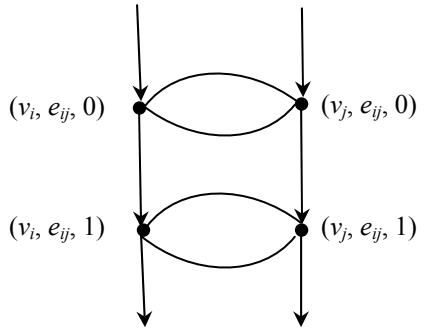


Рис. 7.7

Докажем, что граф G имеет вершинное покрытие мощности k тогда и только тогда, когда граф H содержит гамильтонов контур.

Предположим, что граф G имеет вершинное покрытие мощности k. Без ограничения общности можно считать, что это покрытие образуют вершины v_1, v_2, …, v_k. Рассмотрим следующий контур графа H:

$$a_1 \to (v_1, d_{11}, 0) \to (v_1, d_{11}, 1) \to (v_1, d_{12}, 0) \to \ldots \to (v_1, d_{1r1}, 1) \to$$
$$\to a_2 \to (v_2, d_{21}, 0) \to (v_2, d_{21}, 1) \to (v_2, d_{22}, 0) \to \ldots \to (v_2, d_{2r2}, 1) \to$$
$$\to a_2 \to \ldots \to a_k \to \ldots \to a_1.$$

(Напомним, что d_{1ri} – последнее ребро в списке ребер, инцидентных вершине v_i.) Этот контур проходит по всем вершинам графа G, кроме вершин вида $(v_j, e, 0)$ и $(v_j, e, 1)$, где $j > k$. Пусть построенный контур не проходит через вершины $(v_j, e, 0)$ и $(v_j, e, 1)$. Так как множество $\{v_1, v_2, \ldots, v_k\}$ является вершинным покрытием графа G, существует вершина v_i при $i \le k$, инцидентная ребру e. Тогда в построенном контуре между вершинами $(v_i, e, 0)$ и $(v_i, e, 1)$ поставим вершины $(v_j, e, 0)$ и $(v_j, e, 1)$.

Это можно сделать, так как имеются ребра первого типа (см. рис. 0.0). Расширенный таким способом контур проходит через все вершины графа H, т. е. является гамильтоновым.

Докажем обратное. Предположим, что граф H имеет гамильтонов контур. (Напомним, что гамильтонов контур проходит через каждую вершину в точности по одному разу.) Можно считать, что вершины a_1, a_2, …, a_k при обходе контура встречаются именно в этом порядке, т. е. контур имеет вид:

$$f_0 \to \ldots \to a_1 \to \ldots \to a_2 \to \ldots \ldots \to a_k \to f_0,$$

где – некоторая вершина графа . Этот контур разобьем на k путей π_1, π_2, …, π_k. Путь π_i при $i < k$ начинается в вершине a_i и заканчивается в вершине a_{i+1}. Путь π_k начинается в вершине a_k и заканчивается в вершине a_1. Вершину, в которую ведет дуга контура из a_i обозначим через $(v_i, e, 0)$. Докажем, что множество вершин $U = \{v_1, v_2, \ldots, v_k\}$ является вершинным покрытием графа G. Возьмем ребро e графа G. Пусть оно инцидентно вершинам v_l и v_m , т. е. $e = e_{lm}$. Так как контур проходит через все вершины графа, он пройдет через вершину $(v_l, e, 0)$. Следовательно, эта вершина принадлежит некоторому пути π_i. По построению графа вершинами пути π_i являются либо вершины вида (v_i, e, b), где $v_i \in U$, либо вершины вида (v_i, e, b), где v – вершина, инцидентная ребру e и отличная от v_i. Это означает, что $v_i = v_l$ или $v_i = v_m$. В обоих случаях одна из вершин, инцидентных ребру e принадлежит U. Следовательно, граф G имеет вершинное покрытие мощности k. □

Пример 4. Рисунок 7.8 иллюстрирует доказательство теоремы 7.3. Здесь $k = 2$. Считаем, что ребра, инцидентные вершинам v_1 и

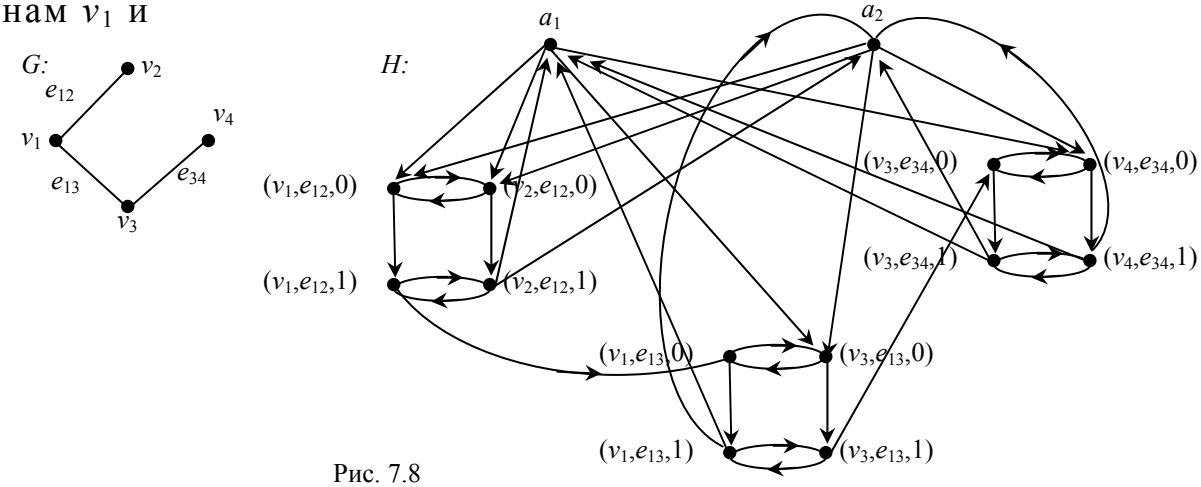


Рис. 7.8

v_3 упорядочены соответственно: (e_{12}, e_{13}) и (e_{13}, e_{34}). (Для вершин v_2 и v_4 этот порядок не указываем, так как этим вершинам инцидентно по одному ребру.)

Граф G имеет вершинное покрытие $\{v_1, v_3\}$ мощности 2, которое определяет гамильтонов цикл графа H: $a_1 \to (v_1, e_{12}, 0) \to \to (v_2, e_{12}, 0) \to (v_2, e_{12}, 1) \to (v_1, e_{12}, 1) \to (v_1, e_{13}, 0) \to (v_1, e_{13}, 1) \to (v_1, e_{13}, 1) \to a_2 \to (v_3, e_{13}, 0) \to (v_3, e_{13}, 1) \to (v_3, e_{34}, 0) \to (v_4, e_{34}, 0) \to (v_4, e_{13}, 1) \to (v_3, e_{34}, 1) \to a_1$. □

Теорема 7. 4. Задача «*гамильтонов контур*» полиномиально сводима к задаче «*гамильтонов цикл*».

Доказательство. Пусть $G = (V, E)$– ориентированный граф. Рассмотрим обыкновенный граф H, множество вершин которого есть множество пар $V \times \{0, 1, 2\}$. Смежными вершинами графа H будут пары вершин трех типов:

1) $(v, 0)$ и $(v, 1)$ для всех $v \in V$;

2) $(v, 1)$ и $(v, 2)$ для всех $v \in V$;

3) $(v, 2)$ и $(w, 0) \Leftrightarrow (v, w) \in E$.

Докажем, что граф G имеет гамильтонов контур тогда и только тогда, когда граф H имеет гамильтонов цикл.

Предположим, последовательность вершин

$$v_1, v_2, \ldots, v_n, v -$$

гамильтонов контур графа G. Тогда, как нетрудно проверить, последовательность вершин

$$(v_1, 0), (v_1, 1), (v_1, 2), (v_2, 0), \ldots, (v_n, 2), (v_1, 0) -$$

гамильтонов цикл графа G.

Докажем обратное. Пусть граф H имеет гамильтонов цикл

$$w_1, w_2, \ldots, w_{3n},$$

где – это пара одного из видов $(v_i, 0)$, $(v_i, 1)$, $(v_i, 2)$и $v_i \in V$. Гамильтонов цикл проходит по всем вершинам графа H, в частности, по вершинам $(v_i, 1)$, где $v_i \in V$. Без ограничения общности можно считать, что цикл по этим вершинам проходит в следующем порядке

$$\ldots, (v_1, 1), \ldots, (v_2, 1), \ldots, (v_n, 1), \ldots .$$

В вершину $(v_i, 1)$ можно «попасть» либо из вершины $(v_i, 0)$, либо из вершины $(v_i, 2)$. Так как цикл можно обходить в любом направлении (граф H неориентированный), предположим, что в вершину $(v_1, 1)$ цикл «приходит» из $(v_1, 0)$. Следующая за $(v_1, 1)$ вершина цикла – вершина $(v_1, 2)$. После этой вершины идет вершина вида $(v', 0)$. Действительно, если цикл из вершины $(v_1, 2)$ идет в вершину вида $(v', 2)$, то в вершину $(v', 1)$ он может прийти только из вершины $(v', 0)$. Но далее можно двигаться только в вершину $(v', 2)$. Получаем противоречие с определением цикла. Следовательно, из вершины $(v_1, 2)$ цикл идет в вер-

шину $(v', 0)$ и затем в $(v', 1)$. это означает, что $v' = v_2$. В итоге мы получаем, что гамильтонов цикл в графе H имеет вид
$(v_1, 0), (v_1, 1), (v_1, 2), (v_2, 0), \ldots, (v_i, 2), (v_{i+1}, 0), \ , (v_n, 2), (v_1, 0)$.

Это означает, что $(v_1, v_2) \in E, \ldots, (v_i, v_{i+1}) \in E, \ldots, (v_n, v_1) \in E$. В таком случае, последовательность вершин

$$v_1, v_2, \ldots, v_n, v_1$$

является гамильтоновым контуром графа G. □

Пример 5. На рисунке 7.9 изображен ориентированный граф G_1 и и обыкновенный граф H_1, полученный из графа G_1 в соответствии с доказательством теоремы 7.4. Легко видеть, что граф G_1 имеет гамильтонов контур, а граф H_1 – гамильтонов цикл. Для аналогичной пары графов G_2 и H_2, изображенных на рисунке 7.10,

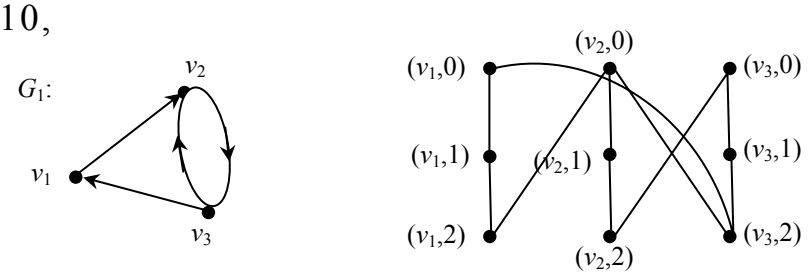


Рис. 7.9

ситуация обратная: граф G_2 не имеет гамильтонова контура, а граф H_2 – гамильтонова цикла.

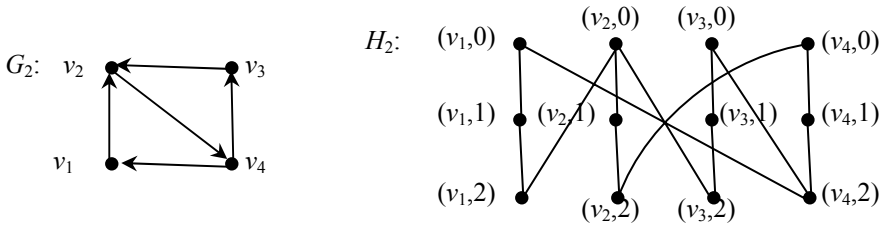


Рис. 7.10

Теорема 7. 5. Задача «*выполнимость*» полиномиально сводима к задаче «3-*выполнимость*».

Доказательство. Возьмем формулу логики высказываний, имеющую конъюнктивную нормальную форму:

$$F = D_1 \ \& \ D_2 \ \& \ \ldots \ \& \ D_p,$$

где $D_1, D_2, \ldots, D_p$ – элементарные дизъюнкции (или дизъюнкты). Каждый из этих дизъюнктов, содержащий более трех литералов, заменим следующим образом. Пусть

$$D_i = L_1 \vee L_2 \vee \ldots L_k -$$

один из таких дизъюнктов. (Здесь $L_1, L_2, \ldots, L_k$ – литералы и $k > 3$.) Заменим дизъюнкт D_i на формулу G_i, имеющую КНФ:

$$G_i = (L_1 \vee L_2 \vee Y_1) \ \& \ (L_3 \vee \neg Y_1 \vee Y_2) \ \& \ \ldots \ \&$$
$$\& \ (L_{k-2} \vee \neg Y_{k-4} \vee Y_{k-3}) \ \& \ (L_{k-1} \vee L_k \vee \neg Y_{k-3}),$$

где Y_1, Y_2, …, Y_{k-3} – новые атомарные формулы, причем для каждого дизъюнкта – свои. Полученную в результате таких замен формулу обозначим через G, т. е. $G = G_1 \,\&\, G_2 \,\&\, \ldots \,\&\, G_p$. (Если дизъюнкт D_i не изменялся, то полагаем $G_i = D_i$.) Ясно, что формула G имеет КНФ, и что каждый ее дизъюнкт содержит не более трех литералов. Нетрудно также подсчитать, что число литералов формулы G не превосходит , где – число литералов формулы F.

Докажем, что формулы F и G одновременно выполнимы или невыполнимы. Предположим, что формула F выполнима, т. е. $\varphi(F) = 1$ для некоторой интерпретации φ. Это означает, что $\varphi(D_i) = 1$ для всех $1 \le i \le p$. Если при построении формулы G дизъюнкт D_i не изменился, то он будет истинным и в формуле G при интерпретации φ. Предположим, что дизъюнкт D_i содержит более трех литералов и был заменен на формулу G_i. Равенство $\varphi(D_i) = 1$ означает, что $\varphi(L_j) = 1$ для некоторого литерала L_j среди L_1, L_2, …, L_k. Расширим интерпретацию φ на множество атомарных формул Y_1, Y_2, …, Y_{k-3} следующим образом:

$$\varphi(Y_1) = \ldots = \varphi(Y_{j-2}) = 1,\ \varphi(Y_{j-1}) = \ldots = \varphi(Y_{k-3}) = 0.$$

Например, если $j = 3$, то $\varphi(Y_1) = 1$, $\varphi(Y_2) = 1$ … $\varphi(Y_{k-3}) = 0$. Нетрудно понять, что после такого расширения выполняется равенство $\varphi(D_i) = 1$. Следовательно, формула G также выполнима.

Докажем обратное утверждение. Предположим, что существует интерпретация ψ такая, что $\psi(G) = 1$. Это означает, что для всех i формула G_i истинна, и следовательно, истинны при интерпретации ψ все дизъюнкты формулы G_i. Пусть

$$\psi(L_1) = \psi(L_2) = \ldots = \psi(L_k) = 0.$$

Рассмотрим последовательно дизъюнкты формулы G_i, (которые, как мы знаем, истинны). Из истинности первого дизъюнкта следует, что $\psi(Y_1) = 1$, истинности второго – $\psi(Y_2) = 1$ и т. д. Из истинности предпоследнего дизъюнкта формулы G следует, что $\psi(Y_{k-3}) = 1$. Но это противоречит равенствам $\psi(L_{k-1}) = \psi(L_k) = 0$ и $\psi(L_{k-1} \vee L_k \vee \neg\, Y_{k-3}) = 1$. Следовательно, хотя бы один из литералов L_1, L_2, …, L_k является истинным при интерпретации ψ, т. е. $\psi(D_i) = 1$. Итак, если дизъюнкт D_i содержал более трех литералов и расширялся при построении формулы G, то он является истинным. Если же он содержал менее четырех литералов, то он в неизменном виде содержится в формуле G, и поэтому тоже является истинным при интерпретации ψ. Это означает, что $\psi(F) = 1$, т. е. что формула F выполнима. □

Пример 6. Приведем несложный пример, иллюстрирующий доказательство теоремы 7. Пусть дана формула

$$F = (X_1 \vee \neg X_2 \vee X_3 \vee X_4 \vee \neg X_5) \,\&\, (\neg X_1 \vee X_3).$$

Тогда формула, построенная в доказательстве этой теоремы, будет иметь вид

$$G = (X_1 \vee \neg X_2 \vee Y_1) \,\&\, (X_3 \vee \neg Y_1 \vee Y_2) \,\&$$
$$\&\, (X_4 \vee \neg X_5 \vee \neg Y_2) \,\&\, (\neg X_1 \vee X_3). \,\square$$

Теорема 7.6. Задача «3-*выполнимость*» полиномиально сводится к задаче «*раскраска*».

Доказательство. Пусть дана формула F, имеющая конъюнктивную нормальную форму:

$$F = D_1 \,\&\, D_2 \,\&\, \ldots \,\&\, D_p,$$

где $D_1, D_2, \ldots, D_p$ – дизъюнкты, каждый из которых содержит не более трех литералов. Предположим далее, что формула F построена из атомарных формул $X_1, X_2, \ldots, X_n$. Можно считать, что $n > 3$. В противном случае к формуле F добавим дизъюнкты, каждый из которых является новой атомарной формулой. При таком расширении формула F сохранит выполнимость (и невыполнимость).

Рассмотрим следующий обыкновенный граф G с $3n + p$ вершинами.

Вершинами графа G будут формулы:

1.1) литералы $X_1, X_2, \ldots, X_n, \neg X_1, \neg X_2, \ldots, \neg X_n$;

1.2) дизъюнкты $D_1, D_2, \ldots, D_p$;

1.3) новые атомарные формулы $Y_1, Y_2, \ldots, Y_n$.

Смежными вершинами графа будут следующие вершины:

2.1) X_i и $\neg X_i$ для всех i;

2.2) X_i и D_k, если X_i не входит в D_k;

$\neg X_i$ и D_k, если $\neg X_i$ не входит в D_k;

2.3) Y_i и Y_j при $i \neq j$;

2.4) Y_i и X_j при $i \neq j$;

Y_i и $\neg X_j$ при $i \neq j$.

Докажем, что граф G можно правильно раскрасить в n+1 цветов тогда и только тогда, когда формула F выполнима.

Предположим, что построенный граф G можно раскрасить в n+1 цветов. Подграф, порожденный вершинами $Y_1, Y_2, \ldots, Y_n$ является полным, поэтому для его раскраски необходимы n цветов. Будем считать, что вершина Y_i раскрашена цветом i. Цвета 1, …, n назовем *основными*, а цвет n+1 – *дополнительным*. Одна из вершин X_j или $\neg X_j$ будет раскрашена так же, как и вершина Y_j, а другая должна быть раскрашена дополнительным цветом. Действительно, вершины X_j и $\neg X_j$ смежны между собой и со

всеми вершинами Y_i при $i \neq j$ (пункты 2.1 т 2.4). Следовательно, для раскраски литералов X_1, X_2, …, X_n, $\neg X_1$, $\neg X_2$, …, $\neg X_n$ требуются все n+1 цветов.

Рассмотрим вершину D_k. Так как дизъюнкт D_k содержит не более трех литералов, а n>3, найдутся литералы X_j и $\neg X_j$, не содержащиеся в D_k. Эти литералы будут смежны с D_k (пункт 2.2). Как показано выше, однин из литералов X_j или $\neg X_j$ раскрашен дополнительным цветом. Это означает, что вершина D_k раскрашена основным цветом.

Предположим, что все литералы, содержащиеся в D_k, раскрашены вспомогательным цветом. Для определенности пусть $D_k = X_1 \vee \neg X_2 \vee X_3$. Тогда литерал $\neg X_1$ будет раскрашен цветом 1, литерал X_2 – цветом 2, литерал $\neg X_3$ – цветом 3, один из литералов X_4 или $\neg X_4$ – цветом 4 и т. д., один из литералов X_n или $\neg X_n$ будет раскрашен цветом n. Следовательно, для раскраски литералов $\neg X_1$, X_2, $\neg X_3$, X_4, $\neg X_4$, …, X_n, $\neg X_n$ потребуются все n основных цветов. Но все эти литералы не содержатся в D_k, и поэтому смежны с D_k (пункт 2.2). Получили противоречие с тем, что D_k раскрашен основным цветом. Это означает, что хотя бы один из литералов, содержащихся в D_k, раскрашен основным цветом.

Рассмотрим следующую интерпретацию

$$\varphi(X_i) = \begin{cases} 1, \text{ если } X_i \text{ раскрашен основным цветом,} \\ 0, \text{ если } X_i \text{ раскрашен вспомогательным цветом.} \end{cases}$$

Пусть – произвольный литерал формулы . В предыдущем абзаце было доказано, что хотя бы один из литералов этого дизъюнкта раскрашен основным цветом. Если это литерал X_i (т. е. атомарная формула), то по построению интерпретации φ имеем равенство $\varphi(D_k) = 1$. Если же основным цветом раскрашен литерал $\neg X_i$, то литерал X_i должен быть, как мы знаем, раскрашен вспомогательным цветом. Поэтому $\varphi(X_i) = 0$ и $\varphi(\neg X_i) = \varphi(D_k) = 1$. Итак, если граф G можно раскрасить n+1 цветами, то формула F выполнима.

Предположим, что формула F выполнима, т. е. найдется интерпретация φ такая, что $\varphi(F) = \varphi(D_1) = \ldots = \varphi(D_p) = 1$. Раскрасим граф G следующим образом. Вершину Y_i раскрасим цветом i. Из двух вершин X_i и $\neg X_i$, ту вершину, для которой значение интерпретации φ равно 0, раскрасим дополнительным цветом, а оставшуюся вершину – цветом i. Так как $\varphi(D_k) = 1$, дизъюнкт D_k содержит литерал L, для которого $\varphi(L) = 1$. Дизъюнкт D_k раскрасим так же, как литерал L. Нетрудно проверить, что все

смежные вершины графа G раскрашены в разные цвета. Следовательно, если формула F выполнима, то граф G можно правильно раскрасить n+1 красками. □

Пример 7. Рассмотрим формулу

$$F = (\neg X_1 \vee \neg X_2 \vee X_3) \,\&\, (X_2 \vee \neg X_4).$$

Для этой формулы $n = 4$, $D_1 = \neg X_1 \vee \neg X_2 \vee X_3$, $D_2 = X_2 \vee \neg X_4$. Граф G, соответствующий этой формуле, изображен на рисунке 7.11. Рассмотрим интерпретацию φ: $\varphi(X_1) = \varphi(X_4) = 0$, $\varphi(X_2) = \varphi(X_3) = 1$. Очевидно, что $\varphi(F) = 1$. Раскрасим граф G следующим образом. Вершину Y_i раскрасим i-ой краской. Вершины $\neg X_1$, X_2, X_3, $\neg X_4$ раскрасим соответственно 1, 2, 3, 4-ой красками. Пятой (дополнительной) краской раскрасим все вершины X_1, $\neg X_2$, $\neg X_3$, X_4. Осталось раскрасить вершины D_1 и D_2 соответственно 1 и 2-ой красками. Легко видеть, что эта раскраска является правильной.

Рассмотрим теперь следующую раскраску h:

$h(Y_i) = i$ для $1 \le i \le 4$,

$h(\neg X_1) = 1$, $h(X_2) = 2$, $h(\neg X_3) = 3$, $h(X_4) = 4$,

$h(X_1) = h(\neg X_2) = h(X_3) = h(\neg X_4) = 5$,

$h(D_1) = 3$, $h(D_2) = 2$.

Эта раскраска является правильной. Она определяет интерпретацию φ: $\varphi(X_2) = \varphi(X_4) = 1$, $\varphi(X_1) = \varphi(X_3) = 0$. Очевидно, что $\varphi(F) = 1$. □

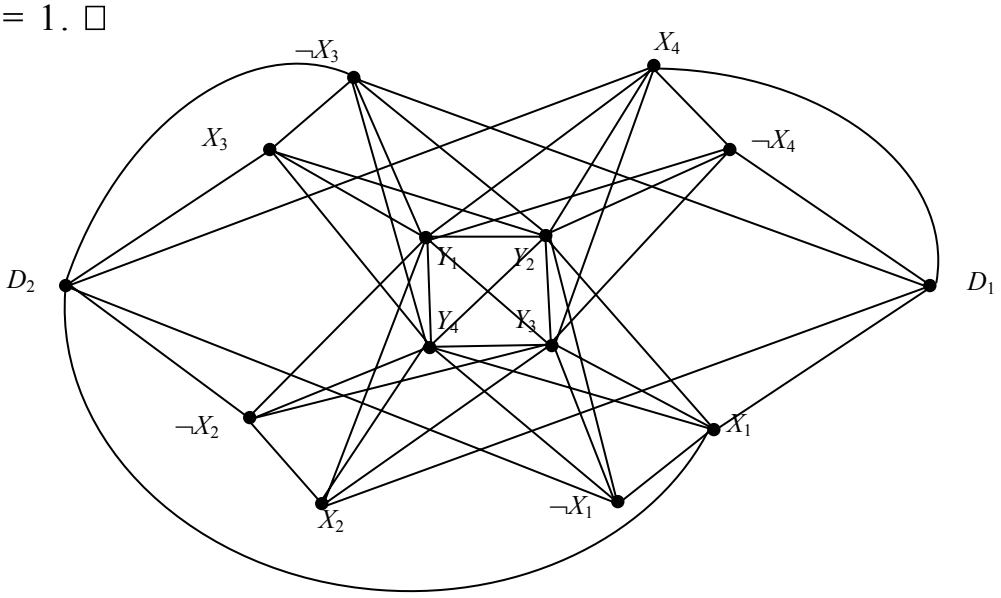


Рис. 7.11

§ 5. Класс *NP-time*

В этом параграфе будут обсуждаться понятия: недетерминированный алгоритм и недетерминированная машина Тьюринга. Первое понятие будем воспринимать на интуитивном уровне, для второго дадим строгое определение.

Выполнение недетерминированного алгоритма, решающего массовую задачу T, будет состоять из двух стадий: *стадии угадывания* и *стадии проверки*. На первой стадии для данной индивидуальной задачи $I \in T$ происходит угадывание некоторой структуры S. На этой стадии алгоритм работает недетерминировано. Затем I и S подаются на стадию проверки, которая выполняется детерминировано. Вторая стадия либо заканчивается ответом «да», либо ответом «нет», либо на второй стадии алгоритм работает бесконечно. Например, в случае задачи «выполнимость» на вход такого алгоритма подается формула F, имеющая конъюнктивную нормальную форму. Предположим, что формула F построена из атомарных формул X_1, X_2, …, X_n. На стадии угадывания строится интерпретация φ, т. е. функция, которая атомарным формулам придает истинностные значения. На второй стадии происходит проверка того, выполняется ли равенство $\varphi(F) = 1$ или не выполняется.

Будем считать, что алгоритм решает массовую задачу T, если выполнены условия:

1) если $I \in T$, то существует структура S, которая угадывается на первой стадии, а вторая стадия, имея на входе I и S, выдает ответ «да»;

2) если $I \notin T$, то для любой структуры S, выданной первой стадией, вторая стадия либо выдает ответ «нет», либо работает бесконечно.

Формальным аналогом понятия недетерминированного алгоритма является понятие недетерминированной машины Тьюринга (сокращенно: НДМТ). Определение такой машины получается из определения «обычной», т. е. детерминированной машины Тьюринга, данного в первом параграфе предыдущей главы, если фразу

«программа – множество команд
с различными левыми частями»

заменить на фразу

«программа – любое множество команд».

Предполагается дополнительно, что уже внесены изменения о двух заключительных состояниях и что машина останавливается только в случае, когда приходит в одно из них. НДМТ на-

чинает работу так же, как и детерминированная машина, т. е. находясь в начальном внутреннем состоянии и обозревая первую непустую ячейку ленты.

Пример 1. Рассмотрим недетерминированную машину Тьюринга M. Пусть $Q = \{q_0, q_1, q_y, q_n\}$, $A = \{a, b, \lambda\}$, λ – пустой символ, $\Sigma = \{a, b\}$. Программа машины M состоит из следующих команд:

$$q_0a \to q_1aR,\ q_0a \to q_na,$$
$$q_0b \to q_yb,\ q_0b \to q_1aR,$$
$$q_0\lambda \to q_0\lambda R,$$
$$q_1a \to q_1aR,\ q_1b \to q_yb,\ q_1\lambda \to q_n\lambda.$$

Если машина находится в состоянии q_0 и обозревает ячейку, в которой записан символ a, то она может, выполнив команду $q_0a \to q_1aR$, перейти в состояние q_1 и сдвинуться на одну ячейку вправо. В этой же ситуации машина может выполнить команду $q_0a \to q_na$ и остановиться. □

Различные команды с совпадающими левыми частями назовем *альтернативными* друг другу (или *альтернативами*). В примере 1 машина содержит две пары альтернатив.

Определение. Пусть A – входной алфавит недетерминированной машины M, Σ – непустое подмножество множества A, не содержащее пустой символ и w – цепочка над Σ. Будем говорить, что машина M *распознает цепочку* w, если она, начиная работу на w, заканчивает ее в состоянии q_y (хотя бы при одном варианте выбора альтернатив на каждом такте).

Выбор альтернатив на каждом такте – это и есть «стадия угадывания» недетерминированного алгоритма, о которой говорилось выше.

Определение. *Языком, распознаваемым машиной* M называется множество всех цепочек, распознаваемых этой машиной.

Другими словами, машина M распознает язык L надΣ , если M, начиная работу на цепочке $w \in \Sigma^*$, заканчивает ее в состоянии q_y (хотя бы при одном варианте выбора альтернатив на каждом такте) тогда и только тогда, когда $w \in L$.

Мы видим, что определение распознаваемости цепочки и языка для НДМТ есть обобщение соответствующих понятий для ДМТ. Именно поэтому для определения распознаваемости цепочки в случае ДМТ был выбран второй вариант (см. § 3).

Пример 2. Убедимся в том, что машина M из примера 1 допускает язык L, цепочки которого содержат хотя бы одну букву b, т. е. $L = \{xby \mid x, y \in \Sigma^*\}$. На пустой цепочке машина работает

бесконечно. Если цепочка w начинается с буквы b, то машина может ее сразу распознать, применив команду $q_0b \to q_yb$. В случае, когда $w = a^nbw'$ и $n > 0$, машина M, применив команды $q_0a \to q_1aR$, $q_1a \to q_1aR$ (n – 1 раз) и $q_1b \to q_yb$ перейдет в распознающее состояние q_y. Если же $w = a^n$ и $n > 0$, то при любом варианте выбора альтернатив на каждом такте в итоге машина переходит в состояние q_n. □

Введем понятие временной сложности недетерминированной машины Тьюринга по аналогии с соответствующим понятием для обычной машины, данным в третьем параграфе.

Пусть Σ - непустое подмножество входного алфавита НДМТ M и $x \in \Sigma^*$. Пусть машина M останавливается на входе x, хотя бы при одном выборе альтернатив на каждом такте. Распространим обозначение $t_M(x)$ на класс недетерминированных машин. Здесь $t_M(x)$ будет обозначать *наименьшее* число тактов, которые может выполнить машина M до остановки, начиная работу на цепочке x.

Определение. *Временной сложностью* НМТ M называется функция T_M: $\boldsymbol{N} \to \boldsymbol{N}$, определяемая равенством

$$T_M(n) = max[\{1\} \cup \{t_M(w) \mid w \in L \text{ и } |w| = n\}].$$

Другими словами, если не существует цепочки, распознаваемой машиной M и имеющей длину n, то $T_M(n) = 1$. Если же такая цепочка w существует, то $T_M(n)$ есть минимально возможное число тактов работы машины $T_M(n)$ на цепочке w до остановки в состоянии q_y. Подчеркнем, что это определение согласовано с аналогичным определением в случае ДМТ.

Разумеется, в предыдущем определении предполагается, что зафиксировано непустое подмножество Σ множества A, не содержащее пустой символ, и что рассматриваются только цепочки над Σ.

Определение. НМТ M называется *полиномиальной*, если существует полином p такой, что неравенство

$$T_M(n) \le p(n)$$

выполняется для всех $n \in \boldsymbol{N}$. Язык называется *недетерминированно полиномиальным*, если существует распознающая этот язык недетерминированная полиномиальная машина.

Класс всех недетерминированно полиномиальных языков обозначим через *NP*.

Формальности с языками пока закончены, перейдем к содержательным задачам. Будем говорить, что задача является *недетерминированно полиномиальной*, если полиномиален соот-

ветствующий язык. Класс всех недетерминированно полиномиальных задач так же, как и класс всех недетерминированно полиномиальных языков, будем обозначать через *NP*.

Теорема 7. 7. Все задачи распознавания из первого параграфа принадлежат классу *NP*.

Эту теорему доказывать не будем. Ограничимся лишь примером.

Пример 3. Рассмотрим задачу «*выполнимость*». Пусть дана формула

$F_1 = (X_1 \vee \neg X_2 \vee X_3)\ \&\ (\neg X_1 \vee X_2 \vee X_3)\ \&\ (\neg X_1 \vee \neg X_2 \vee \neg X_3)$.

На стадии угадывания алгоритм «выдает» интерпретацию

$$\varphi(X_1) = 1,\ \varphi(X_2) = 1,\ \varphi(X_3) = 0.$$

А на стадии проверки он проверяет, что $\varphi(F_1) = 1$.

Рассмотрим теперь формулу

$F_2 = (X_1 \vee X_2)\ \&\ (\neg X_1 \vee X_2)\ \&\ (X_1 \vee \neg X_2)\ \&\ (\neg X_1 \vee \neg X_2)$.

Легко видеть, что формула F_2 невыполнима. Какую бы интерпретацию ψ недетерминированный алгоритм на стадии угадывания ни выдал, на стадии проверки получается, что $\psi(F_2) = 0$. Если недетерминированный алгоритм «снабжен» способом проверки того, выдавалась ли очередная интерпретация ранее, то алгоритм закончит работу и выдаст «нет». Если же в алгоритм такой способ не заложен, то он будет работать бесконечно. □

Ясно, что $\boldsymbol{P} \subseteq \boldsymbol{NP}$, поскольку всякая полиномиальная ДМТ – частный случай полиномиальной НДМТ. Вопрос о том, выполняется ли равенство $\boldsymbol{P} = \boldsymbol{NP}$, является сейчас одним из основных в теории алгоритмов и в дискретной математике в целом. Этот вопрос интересует также и программистов-теоретиков, т. е. тех исследователей, которые работают в области информатики. В литературе этот вопрос получил название «*проблема* $\boldsymbol{P} = \boldsymbol{NP}$». Эта проблема пока не решена.

Из теоретических результатов, характеризующих взаимоотношение детерминированной и недетерминированной вычислимости, приведем следующий результат.

Теорема 7. 8. Если язык L принадлежит классу $\boldsymbol{NP}$, то существует полином $p(n)$ такой, что задача распознавания принадлежности к языку L может быть решена детерминированной машиной с временной сложностью из класса $O(2^{p(n)})$.

Доказывать эту теорему здесь не будем. С ее доказательством можно ознакомиться в книге [ГД, стр. 49-50].

§ 6. *NP*-полнота

Как сказано в § 3, задачи 1 – 8 принадлежат классу ***P***. Для остальных задач в настоящее время не известны полиномиальные алгоритмы, решающие эти задачи. Теорема из предыдущего параграфа утверждает, что задачи 9 – 18 принадлежат классу ***NP***. Оказывается, что все эти задачи обладают следующим удивительным свойством: любая задача из класса ***NP*** полиномиально сводится к каждой из задач 9 - 18. Это свойство называется *NP*-полнотой. Дадим формальное

Определение. Язык L называется *NP*-полным, если $L \in \boldsymbol{NP}$ и любой язык из ***NP*** полиномиально сводится к L.

Следующее утверждение сразу следует из определения.

Лемма. Любые два *NP*-полных языка полиномиально эквивалентны. Если язык L_1 полиномиально сводится к языку L_2, L_1 является *NP*-полным и L_2 принадлежит классу ***NP***, то язык L_2 также является *NP*-полным.

Как обычно, задачу будем называть *NP*-полной, если соответствующий ей язык при некоторой разумной кодировке является *NP*-полным.

Цель параграфа – показать, что задачи 9 – 18 являются *NP*-полными. Начнем с результата Кука, ставшего уже классическим.

Теорема 7. 9. Задача «*выполнимость*» является *NP*-полной.

Доказательство. Возьмем язык L, принадлежащий классу *NP*. Пусть M – недетерминированная машина Тьюринга, распознающая язык L за время, ограниченное полиномом $p(n)$. Будем считать, что $p(n) \geq n + 1$. В противном случае в качестве полинома $p(n)$ можно взять $p(n) + n + 1$. Пусть, далее, $Q = \{q_1, q_2, \ldots, q_d\}$ – внутренний алфавит машины, $A = \{a_1, a_2, \ldots, a_m\}$ – внешний алфавит и Σ – алфавит языка L. Состояние q_1 является начальным, q_d – распознающим (состоянием «*yes*»). Возьмем цепочку w из языка L. Так как машина M распознает язык L, существует последовательность конфигураций этой машины

$$C_0, C_1, \ldots, C_r,$$

где C_0 – начальная конфигурация, т. е. $C_0 = q_1 w$, а C_r – заключительная конфигурация, содержащая состояние q_d. Поскольку машина M распознает язык за полиномиальное время, выполняется неравенство $r \leq p(n)$. В результате выполнения команды длина конфигурации может возрасти разве лишь на единицу. Так как $p(n) \geq n + 1$, каждая конфигурация имеет длину не более, чем $p(n)$. («Плюс один» для учета того, что конфигурация

содержит внутреннее состояние и что нумерация конфигураций начинается с нуля.) Будем считать, что все конфигурации занимают ровно n ячеек. Это предположение не ограничивает общности, так как каждую конфигурацию можно дополнить любым количеством пустых символов. Сделаем еще одно предположение. Для доказательства теоремы сделаем еще ряд предположений. Будем считать, что

– $r = p(n)$,

– ячейки входной ленты занумерованы целыми числами,

– входная цепочка записана на ленте, начиная с первой ячейки,

– машина в процессе работы не выходит влево за первую ячейку.

Пусть $w \in \Sigma^*$ и $|w| = n$. По цепочке w построим формулу F, которая выполнима тогда и только тогда, когда $w \in L$, т. е. w распознается машиной M.

В построении F будут использоваться следующие атомарные формулы (в квадратных скобках указана предполагаемая интерпретация атомарной формулы):

1) $A\langle i, j, t\rangle$, где $1 \le i \le p(n)$, $1 \le j \le m$, $0 \le t \le p(n)$. [В момент времени t (после выполнения такта с номером t) i-ая ячейка ленты содержит символ a_j.] Начальное значение $t = 0$ «представляет» конфигурацию C_0. Всего атомарных формул этого вида $O(p^2(n))$.

2) $Q\langle k, t\rangle$, где $1 \le k \le s$, $0 \le t \le p(n)$. [В момент времени t внутреннее состояние машины равно q_k.] Всего атомарных формул этого вида $O(p(n))$.

3) $H\langle k, t\rangle$, где $1 \le k \le p(n)$, $0 \le t \le p(n)$. [В момент времени t обозревается ячейка с номером k.] Всего атомарных формул этого вида $O(p^2(n))$.

Для сокращения записи формулы F, введем следующее обозначение: формулу

$$(X_1 \vee X_2 \vee \ldots \vee X_r) \,\&$$
$$\& \,[(\neg X_1 \vee \neg X_2) \,\&\, (\neg X_1 \vee \neg X_3) \,\&\, \ldots \,\&\, (\neg X_{r-1} \vee \neg X_r)]$$

будем обозначать через $U(X_1, X_2, \ldots, X_r)$. Эта формула истинна тогда и только тогда, когда истинна в точности одна из атомарных формул $X_1, X_2, \ldots, X_r$. Отметим, что длина формулы $U(X_1, X_2, \ldots, X_r)$ есть $O(r^2)$.

Формула F будет описывать работу машины Тьюринга M при распознавании цепочки w. Она будет конъюнкцией формул $F_1, F_2, \ldots, F_7$. Ниже будут выписаны эти формулы с указанием содержательного смысла и длины каждой из них. Напомним,

что связки & и $\vee$ имеют более высокий приоритет, нежели $\to$ и $\leftrightarrow$.

1. В каждой конфигурации машина обозревает точно одну ячейку входной ленты:

$$F_1 = \&\{U(H\langle 1, t\rangle, H\langle 2, t\rangle, \ldots, H\langle p(n), t\rangle) \mid 0 \le t \le p(n)\}.$$

Длина формулы F_1 имеет порядок $p^3(n)$.

2. В каждый момент времени в каждой клетке записан точно один символ

$$F_2 = \&U(A\langle i, 1, t\rangle, A\langle i, 2, t\rangle, \ldots, A\langle i, m, t\rangle) \mid 0 \le i, t \le p(n)\}.$$

Длина формулы F_2 имеет порядок $p^2(n)$.

3. Каждая конфигурация содержит только одно внутреннее состояние

$$F_3 = \&\{U(Q\langle 1, t\rangle, Q\langle 2, t\rangle, \ldots, Q\langle p(n), t\rangle) \mid 0 \le t \le p(n)\}.$$

Длина формулы F_3 имеет порядок $p(n)$.

4. В момент t можно изменить состояние только одной ячейки

$$F_4 = \&\{A\langle i, j, t\rangle \leftrightarrow A\langle i, j, t+1\rangle \vee H\langle i, t\rangle \mid 1 \le i, t \le p(n), 1 \le i \le m\}.$$

Длина формулы F_4 имеет порядок $p^2(n)$. Заметим, что формула F_4 не имеет КНФ, но приводится к этой форме за линейное (относительно длины формулы F_4) время.

5. Очередная конфигурация получается из предыдущей срабатыванием некоторой команды

$$F_5 = \&\{A\langle i, j, t\rangle \,\&\, H\langle i, t\rangle \,\&\, Q\langle k, t\rangle \to F_5' \mid 0 \le i, t \le p(n), 1 \le j \le m\},$$

где F_5' – дизъюнкция формул $A\langle i, j', t+1\rangle \,\&\, H\langle i', t+1\rangle \,\&\, Q\langle k', t+1\rangle$ для всех команд вида $q_k a_j \to q_{k'} a_{j'} D$ и $i' = i+1$, если $D = R$, $i' = i-1$, если $D = L$, $i' = i$, если – пустой символ, т. е. при выпонении команды обозреваемая ячейка остается прежней. Длина формулы F_5 имеет порядок $p^2(n)$. Как и F_4, формула F_5 не имеет КНФ, но легко к этой форме приводится.

6. Первая конфигурация является начальной

$$F_6 = Q\langle 1, 0\rangle \,\&\, H\langle 1, 0\rangle \,\&\{A\langle i, j_i, 0\rangle \mid 1 \le i \le n\} \,\&\, \&\{C\langle i, 1, 0\rangle \mid n < i \le p(n)\}.$$

Предполагается, что i-ый символ цепочки w есть a_j и что a_1 – пустой символ. Длина формулы F_6 имеет порядок $p(n)$.

7. Последняя конфигурация является заключительной

$$F_7 = Q\langle d, p(n)\rangle.$$

(Здесь d – номер распознающего внутреннего состояния.) Длина формулы F_7 не зависит от n.

Как уже отмечалось, $F = F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_7$. Формулы F_1, F_2, F_3, F_6, F_7 имеют конъюнктивную нормальную форму, формулы

F_4 и F_5 легко (линейное время) к такой форме приводятся. Следовательно, можно считать, что формула F имеет конъюнктивную нормальную форму. Исходя из построения формул F_1, …, F_7, мы видим, что формула F истинна тогда и только тогда, когда w распознается машиной M. Это означает, что задача распознавания принадлежности к языку L сводится к задаче «выполнимость». Длина формулы F имеет порядок $p^3(n)$. Следовательно, сводимость является полиномиальной. □

Следствие. Задачи 9 – 17 из первого параграфа являются *NP*-полными.

Это вытекает из результатов предыдущего параграфа и только что доказанной теоремы.

Итак, значительная часть практически важных задач оказались *NP*-полными. Напомню, что все известные алгоритмы решения задач 9 – 18 имеют временную сложность не менее, чем $c2^n$, где n – длина индивидуальной задачи, c – положительная константа. В силу этого, естественно поставить вопрос о существовании приближенных полиномиальных алгоритмов, решающих эти задачи. В следующем параграфе это будет сделано для оптимизационных задач.

§ 7. Приближенные алгоритмы

Ранее мы уже пользовались в интуитивном смысле термином «оптимизационная задача». Дадим формальное

Определение. *Оптимизационная массовая задача* есть совокупность трех компонент:

1) множества T индивидуальных задач ;

2) функции σ, которая каждой индивидуальной задаче $I \in T$ ставит в соответствие конечное множество $\sigma(I)$ *допустимых решений* этой задачи;

3) функции d, которая каждой индивидуальной задаче I и каждому ее допустимому решению $S \in \sigma(I)$ ставит в соответствие положительное действительное число $d(I, S)$, называемое *величиной решения*.

Определение. Оптимизационная массовая задача называется задачей *минимизации* [*максимизации*], если для любой индивидуальной задачи I требуется найти среди $\sigma(I)$ (хотя бы одно) решение с наименьшей [наибольшей] величиной.

Если рассматривается задача минимизации [максимизации], то для индивидуальной задачи I через $OPT(I)$ будем обозначать наименьшую [наибольшую] величину допустимого решения из $\sigma(I)$.

Определение. Алгоритм A называется *приближенным алгоритмом* решения массовой задачи, если для любой $I \in T$ алгоритм находит некоторое допустимое решение из $\sigma(I)$.

Через $A(I)$ будем обозначать величину допустимого решения, найденного алгоритмом A, а через $OPT(I)$ – величину оптимального решения. Если $A(I) = OPT(I)$ для любой индивидуальной задачи I, то алгоритм A будем называть *точным*.

Приведем ряд примеров приближенных алгоритмов.

Пример 1. Рассмотрим задачу «*хроматическое число*». (Напомним, что эта задача является оптимизационным вариантом задачи «раскраска».) В этом случае допустимое решение – любая правильная раскраска графа, а функция d – число использованных красок. Рассмотрим следующий алгоритм, который в литературе получил название «алгоритма последовательной раскраски». Пусть вершины графа занумерованы: $V = \{v_1, v_2, \ldots, v_n\}$. Вершину v_1 красим первой краской. Если v_1 и v_2 несмежны, то вершину v_2 также красим первой краской. Если же эти вершины смежны, то v_2 красим второй краской. Предположим, что уже раскрашены вершины $v_1, v_2, \ldots, v_i$, на это затрачено p красок и $i < n$. Если вершина v_{i+1} смежна всем указанным вершинам, то ее красим $(p+1)$-ой краской. Предположим, что среди $v_1, v_2, \ldots, v_i$ существуют вершины, несмежные вершине v_{i+1}. Пусть v_k – первая (по номеру) среди таких вершин. Тогда вершину v_{i+1} красим той же краской, что и вершину v_k. Ясно, что в результате получается правильная раскраска графа и что алгоритм раскрашивает граф за время $O(n^2)$.

Рис. 7.12

Как показывает пример графа, изображенного на рис. 7.12, алгоритм последовательной раскраски не является оптимальным. Алгоритму последовательной раскраски понадобится три краски, а хроматическое число графа равно двум.□

Пример 2. В этом примере рассмотрим задачу «*коммивояжер*». Исходный взвешенный граф G будем задавать матрицей весов M порядка n, где n – число вершин графа и вес ребра (u, v) равен $M[u, v]$. Вес $M[u, v]$ ребра (u, v) иногда содержательно удобно воспринимать, как наличие «прямой» дороги от u к v. Если прямой дороги нет, то $M[u, v] = \infty$. (Матрица M является симметрической матрицей с нулевой главной диагональю, так как граф G неориентированный и не имеет петель.) Будем предполагать, что граф G является полным. Известно, что задача остается NP-полной и при этом предположении. Условимся, что в

этом параграфе термин "задача «*коммивояжер*»" будет означать задачу для полного графа G. Множество допустимых решений в этом случае – множество всех гамильтоновых циклов. Функция d – суммарный вес ребер цикла.

В дальнейшем мы будем рассматривать еще одно условие:

$$M[u, w] \le M[u, v] + M[v, w]$$

для любых вершин u, v, w графа G. Это условие часто называется *неравенством треугольника*. Оказывается, что если рассматривать задачу «*коммивояжер*» для графов с неравенством треугольника, то она останется *NP*-полной (см. [АБР, стр. 259]). Этот вариант задачи коммивояжера будем называть «*коммивояжер + НТ*».

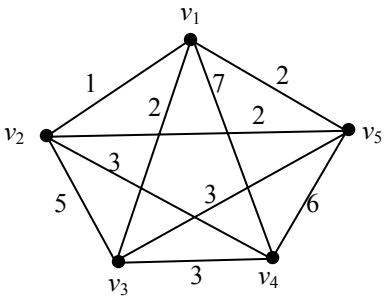


Рис. 7.13

Рассмотрим простой алгоритм построения маршрута коммивояжера, который называется «*ближайший сосед*». Алгоритм состоит в следующем. В качестве первой вершины u_1 цикла берется любая вершина графа G. В качестве второй вершины u_2 выбирается ближайшая к u_1 вершина среди вершин из $V \setminus \{u_1\}$. (Если таких вершин несколько, то берется любая из них.) Третья вершина u_3 – ближайшая к u_2 вершина из $V \setminus \{u_1, u_2\}$ и т. д. Цикл завершает вершина u_1. Например, если в графе, изображенном на рис. 7.13, в качестве первой вершины взять вершину v_1, то алгоритм «выдаст» цикл v_1, v_2, v_5, v_3, v_4, v_1. Его вес будет равен 16. Заметим, что этот цикл не является минимальным. Цикл v_1, v_5, v_2, v_4, v_3, v_1 имеет минимальный вес, равный 12. Отметим, что граф, изображенный на рис. 7.13 неравенству треугольника не удовлетворяет. □

Пример 3. Нетрудно привести примеры графов, для которых алгоритм «*ближайший сосед*» дает решения очень далекие от оптимальных. Это видно из рис 7.13, так как вес ребра (v_4, v_1) можно сделать сколь угодно большим. Можно получить лучшие результаты, применяя более тонкий способ выбора очередной вершины, включаемой в цикл. Один из вариантов такого выбора реализован в алгоритме «*ближайшая вставка*».

Приведем описание этого алгоритма. Пусть T – множество вершин графа. *Расстоянием от вершины v до множества T* назовем величину

$$d(v, T) = min\{M[v, w] \mid w \in T\}.$$

Алгоритм запишем в виде четырех шагов.

Шаг 1. Выбрать произвольную вершину v графа и циклический маршрут, состоящий из одной этой вершины, объявить текущим маршрутом T.

Шаг 2. Если все вершины графа содержатся в текущем маршруте, то выдать маршрут T и завершить работу.

Шаг 3. Среди вершин графа, не входящих в маршрут T, выбрать вершину, для которой расстояние до T является наименьшим. Пусть v – такая вершина, u – вершина из T, для которой $M[v, u] = d(v, T)$. Следующую за u в маршруте T вершину обозначим через w.

Шаг 4. Расширить маршрут T, поставив вершину v между u и w. Перейти к шагу 2.

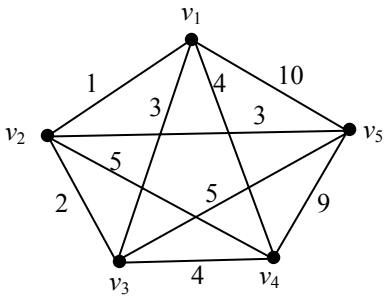


Рис.7.14

Рассмотрим работу этого алгоритма на примере графа, изображенного на рис 7.14. В качестве начальной вершины возьмем v_1. Таблица 7.1 показывает, как изменяется текущий маршрут T в зависимости от числа итераций k (числа проходов) алгоритма.

Найденный алгоритмом «*ближайшая вставка*» маршрут v_1, v_2, v_5, v_3, v_4, v_1 имеет вес 17. Если в качестве начальной вершины взять ту же вершину v_1, то алгоритм ближайший сосед нашел бы маршрут v_1, v_2, v_3, v_4, v_5, v_1, который имеет вес 25. □

Легко видеть, что приведенные в примерах 2 и 3 алгоритмы являются полиномиальными. Они выдают приближенное решение за время $O(n^2)$, где n – число вершин графа.

Таблица 7. 1

Число итераций k	Текущий маршрут T
1	v_1, v_2, v_1
2	v_1, v_2, v_3, v_1
3	v_1, v_2, v_5, v_3, v_1
4	v_1, v_2, v_5, v_3, v_4, v_1

Мы уже отмечали, что алгоритм «ближайший сосед» может дать результаты, сколь угодно далекие от оптимальных. Как показывает следующее утверждение алгоритм «ближайшая вставка» отличается от этого алгоритма «в лучшую сторону».

Теорема 7.10. Пусть G – полный граф, матрица весов которого удовлетворяет неравенству треугольника, A – алгоритм «ближайшая вставка». Тогда

$$A(G) \le 2OPT(G).$$

Доказывать теорему 1 здесь не будем, Ее доказательство можно найти, например, в [АБР, стр. 265]. Отметим, что существуют примеры, показывающие, что приведенная выше оценка для $|A(G)|$ является точной.

Оказывается, что вопрос «***P*** = ***NP***?» является существенным не только для существования точных полиномиальных алгоритмов, но и для существования приближенных полиномиальных алгоритмов. В подтверждение этого приведем теоремы 2 и 3.

Теорема 7 11. Пусть ***P*** ≠ ***NP***. Не существует полиномиального приближенного алгоритма A, решающего задачу «наибольшее независимое множество» такого, что выполняется неравенство

$$OPT(G) \le cA(G)$$

для некоторой неотрицательной константы c и любого обыкновенного графа G.

Доказательство проведем методом от противного. Предположим, что алгоритм A из формулировки теоремы 2 существует. Рассмотрим произвольный граф G. Сначала для данного графа G строится граф $G^* = G_1 \cup G_2 \cup \ldots \cup G_{c+1}$, где $G_1, G_2, \ldots, G_{c+1}$ – графы, изоморфные графу G с попарно непересекающимися множествами вершин.

Ясно, что в графе G найдется независимое множество мощности $[A(G^*)/(c+1)]$, где квадратные скобки обозначают целую часть числа. Для этого надо просмотреть, сколько вершин алгоритм A выбирает в каждом графе $G_1, G_2, \ldots, G_{c+1}$ и среди этих множеств взять наибольшее. Получаем, что

$$A(G) \le [A(G^*)/(c+1)] \le A(G^*)/(c+1) \le OPT(G).$$

Из этих неравенств следует, что

$$(\mathrm{c} + 1)A(G) \le (\mathrm{c} + 1)OPT(G)$$

Используем неравенство из формулировки теоремы:

$$(\mathrm{c} + 1)A(G) \le (\mathrm{c} + 1)OPT(G) \le$$
$$\le (c + 1)(A(G) + c) = (c + 1)A(G) + c(c + 1).$$

В итоге получим, что $(\mathrm{c} + 1)A(G) \le (c + 1)A(G) + c(c + 1)$, т. е. что $c = 0$.

Мы получили, что A – точный полиномиальный алгоритм, решающий задачу «наибольшее независимое множество». Это противоречит неравенству ***P*** ≠ ***NP*** и *NP*-полноте этой задачи. □

Теорема 7. 12. Пусть ***P*** ≠ ***NP***. Не существует полиномиального приближенного алгоритма A, решающего задачу «коммивояжер» такого, что выполняется неравенство

$$A(G) \le c \cdot OPT(G)$$

для некоторой константы c и любого полного взвешенного графа G.

Доказательство. Как и в случае теоремы 2, применим метод от противного. Предположим, что алгоритм A, о котором говорится в формулировке теоремы, существует. Рассмотрим произвольный обыкновенный граф $H = (V, E)$. Пусть $V = \{v_1, v_2, \ldots, v_n\}$. Исходя из графа H, построим граф G с множеством вершин V, что и следующей матрицей весов

$$m_{ij} = \begin{cases} 1, \text{ если ребро } (v_i, v_j) \text{ принадлежит } E; \\ c\cdot|V| \text{ , если } i \neq j \text{ и ребро } (v_i, v_j) \text{ не принадлежит } E; \\ 0, \text{ если } i = j. \end{cases}$$

Предположим, что граф H имеет гамильтонов цикл. Тогда $OPT(G) = |V|$ и по предположению $A(G) \leq c\cdot OPT(G) = c|V|$. Пусть теперь граф H не имеет гамильтонова цикла. Тогда любой маршрут коммивояжера в графе G будет содержать «новое» по сравнению с ребро. Это означает, что в этом случае $A(G) > c|V|$. Итак, граф H имеет гамильтонов цикл тогда и только тогда, когда $A(G) \leq c|V|$. Но алгоритм A является полиномиальным. Следовательно, задача «гамильтонов цикл» принадлежит классу $\boldsymbol{P}$. Получили противоречие с неравенством $\boldsymbol{P} \neq \boldsymbol{NP}$ и NP-полнотой этой задачи.□

Задачи

1. Доказать, что для ДМТ варианты 1 и 2 определения распознаваемости эквивалентны.

2. Доказать, что класс $\boldsymbol{P}$ замкнут относительно следующих теоретико-множественных операций: пересечения, объединения и разности

3. *Доминирующим множеством* обыкновенного графа называется непустое множество вершин такое, что каждое ребро графа инцидентно некоторой вершине из этого множества. Задача «доминирующее множество» формулируется следующим образом: для данного обыкновенного графа G и натурального числа k определить, имеется ли в графе G доминирующее множество мощности k. Доказать, что задача доминирующее множество полиномиально сводима к задаче «покрытие множествами».

4. В § 1 задача «коммивояжер» была определена в двух вариантах. Доказать, что эти варианты полиномиально эквивалентны.

5. Для логики высказываний доказать *NP*-полноту следующих задач.

5.1. Распознать, является ли формула тождественно истинной.

5.2. Распознать, являются две формулы равносильными.

5.3. Распознать, является ли формула логическим следствием конечного множества формул.

5.4. Распознать, является ли конечное множество формул выполнимым.

6. Доказать, что задача «изоморфизм подграфу» *NP*-полна. Эта задача формулируется следующим образом: для данных обыкновенных графов G и H определить, существует ли в графе H подграф, изоморфный графу G.

7. Множество вершин M ориентированного графа *разрезает* контуры, если любой контур графа проходит хотя бы через одну вершину из M. Задача «множество вершин, разрезающих контуры», формулируется следующим образом. Дан ориентированный граф и число k. Определить, существует ли в графе такое множество вершин, разрезающих контуры, мощности k. Доказать, что эта задача *NP*-полна.

Литература

1. [АБВ] Асанов М. О., Баранский В. А., Расин В. В. Дискретная математика: графы, матроиды, алгоритмы. – М.: «РХВ», 2001.

2. [АХУ] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: «Мир», 1979.

3. [ГБ] Гильберт Д., Бернайс П. Основания математики: логические исчисления и формализация арифметики. – М.: «Наука», 1979.

4. [ГД] Гэри М., Джонсон Д. Вычислительные машины и трудно решаемые задачи. М.: «Мир», 1982.

5. [ЕП] Ершов Ю. Л., Палютин Е. А. Математическая логика. – СПб: «Лань», 2005.

6. [И] Игошин В. И. Математическая логика и теория алгоритмов. – М.: «Академия», 2004.

7. [К] Катленд Н. Вычислимость. Введение в теорию рекурсивных функций. – М.: «Мир», 1983.

8. [Кл] Клини. Математическая логика. – М.: «Мир», 1973.

9. [КА-В] Кузнецов О. П., Адельсон-Вельский Г. М. – Дискретная математика для инженера.

10. [Л] Линдон Р. Заметки по логике. М.: «Мир», 1968.

11. [Лор] Лорьер Ж.-Л. Системы искусственного интеллекта. – М.: «Мир», 1991.

12. [М] Мальцев А. И. Алгоритмы и рекурсивные функции. – М.: «Наука», 1986.

13. [МН] Марков А. А., Нагорный Н. М. Теория алгорифмов. – М.: «Наука», 1984.

14. [Мен] Мендельсон Э. Введение в математическую логику. – М.: «Наука», 1984.

15. [Мин] Минский М. Вычисления и автоматы. – М.: «Мир», 1984.

16. [Н] Новиков П. С. Элементы математической логики. – М.: «Наука», 1973.

17. [СО] Судоплатов С. В., Овчинникова Е. В. Математическая логика и теория алгоритмов. – М.: «ИНФРА-М», 2004.

18. [У] Ульман Дж. Основы систем баз данных. М.: «Финансы и статистика», 1983.

19. [Ч] Черч А. Введение в математическую логику. – М.: Мир, 1963.

20. [Я] Яблонский С. В. Введение в дискретную математику. – М.: «Высшая школа», 2006.

Содержание

Введение..... 3

Глава 1. Логика высказываний..... 7
§ 1. Высказывания и операции над ними..... 7
§ 2. Формулы логики высказываний, интерпретация..... 9
§ 3. Равносильность и законы логики высказываний..... 12
§ 4. Логическое следствие..... 16
§ 5. Нормальные формы в логике высказываний..... 18
§ 6. Контактные схемы..... 23
Задачи..... 25
Ответы, указания и решения..... 28

Глава 2. Логика предикатов первого порядка 32
§ 1. Предикаты и операции над ними 32
§ 2. Формулы логики первого порядка 35
§ 3. Интерпретация в логике первого порядка 37
§ 4. Равносильность и законы логики первого порядка 39
§ 5. Логическое следствие 42
§ 6. Нормальные формы 44
§ 7. Невыразимость в логике первого порядка 49
§ 8. Многосортная логика первого порядка 51
Задачи..... 56
Ответы, указания и решения..... 64

Глава 3. Исчисление предикатов 68
§ 1. Об определении некоторых семантических понятий... 68
§ 2. Теоремы о замене (о подстановке)..... 71
§ 3. Аксиоматизация логики предикатов..... 74
§ 4. Теорема о дедукции..... 76
§ 5. Оправданность аксиоматизации..... 80
§ 6. Теорема о непротиворечивости (леммы)..... 82
§ 7. Теорема о непротиворечивости (доказательство теоремы) 87
§ 8. Теоремы о полноте и о компактности 89
§ 9. Независимость аксиом 90
§ 10. Другие аксиоматизации 93
Задачи 95
Решения 96

Глава 4. Метод резолюций 99
§ 1. Метод резолюций в логике высказываний 99
§ 2. Подстановка и унификация 103
§ 3. Метод резолюций в логике первого порядка 108

§ 4. Эрбрановский универсум множества дизъюнктов 113
§ 5. Семантические деревья, теорема Эрбрана 116
§ 6. Полнота метода резолюций в логике первого порядка 120
§ 7. Стратегии метода резолюций 122
§ 8. Применение метода резолюций 124
§ 9. Метод резолюций и логическое программирование 128
Задачи..... 132
Ответы, указания и решения..... 135
Глава 5. Функции *k*-значной логики 138
§ 1. Булевы функции. Способы задания 138
§ 2. Булевы функции. Замкнутость и полнота 144
§ 3. Самодвойственные функции 148
§ 4. Монотонные функции 151
§ 5. Линейные функции 153
§ 6. Критерий полноты классов булевых функций 155
§ 7. Сокращенные ДНФ 158
§ 8. Минимальные ДНФ 163
§ 9. Функции *k*-значной логики. Способы задания 167
§ 10. Функции *k*-значной логики. Замкнутость и полнота . 169
§ 11. Классы сохранения отношений 172
§ 12. Критерий Розенберга 175
Задачи..... 183
Ответы, указания и решения..... 187
Глава 6. Алгоритмы и машины Тьюринга..... 191
§ 1. Понятие машины Тьюринга 192
§ 2. Примеры машин Тьюринга 196
§ 3. Функции, вычислимые на машинах Тьюринга 205
§ 4. Универсальные машины Тьюринга 211
§ 5. Алгоритмически неразрешимые проблемы 214
§ 6. Реурсивные и рекурсивно перечислимые множества .. 217
§ 7. Многоленточные машины 221
§ 8. Рекурсивные функции 222
§ 9. Алгоритмы Маркова 225
Задачи..... 228
Глава 7. Сложность алгоритмов 231
§ 1. Примеры задач 231
§ 2. Задачи и языки 236
§ 3. Класс ***P**-time* 239
§ 4. Полиномиальная сводимость 242
§ 5. Класс ***NP**-time* 255
§ 6. *NP*-полнота 259

§ 7. Приближенные алгоритмы 262
Задачи..... 267
Литература 269