Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

ФАКУЛЬТЕТ ДИСТАНЦИОННОГО ОБУЧЕНИЯ (ФДО)

Т. О. Перемитина

МАТЕМАТИЧЕСКАЯ ЛОГИКА И ТЕОРИЯ АЛГОРИТМОВ

Учебное пособие

Томск
2016

УДК 510.6 + 510.51
ББК 22.12я73
П 270

Рецензенты:

Т. А. Ципилева, канд. техн. наук, доцент кафедры автоматизации обработки информации ТУСУР;

О. С. Токарева, канд. техн. наук, доцент кафедры вычислительной техники Института кибернетики Томского политехнического университета

Перемитина Т. О.

П 270 Математическая логика и теория алгоритмов : учебное пособие / Т. О. Перемитина. – Томск : ФДО, ТУСУР, 2016. –132 с.

В учебном пособии представлены разделы, традиционно изучаемые в курсе математической логики: алгебра высказываний, булева алгебра и логика предикатов. Дается введение в теорию алгоритмов и вычислимых функций. Пособие позволяет освоить основные положения дисциплины, а также получить практические навыки по использованию методов математической логики и теории алгоритмов для решения практических задач и их программной реализации. Пособие предназначено для самостоятельной работы студентов при изучении дисциплины «Математическая логика и теория алгоритмов».

ОГЛАВЛЕНИЕ

Введение

В научном познании, практической деятельности и повседневной жизни нам постоянно приходится убеждать своих собеседников и оппонентов в правильности и обоснованности своих утверждений, гипотез и мнений. Хотя на убеждение людей могут влиять также их эмоции, настроения, склонности и даже предубеждения, все же наибольшей убедительностью обладают доводы, опирающиеся на разум и факты [4].

Логике принадлежит центральная роль в обосновании правильности наших рассуждений, так как именно соблюдение ее правил предохраняет нас от ошибочных выводов. Логика была создана Аристотелем как наука, позволяющая различать правильные определения и умозаключения от неправильных и тем самым вскрывать ошибки в рассуждениях и публичных речах ораторов. Однако в дальнейшем логика стала утрачивать свои связи с ораторским искусством и риторикой [13].

Тенденция к символизации и формализации логики привела ее со временем к новому мощному подъему, завершившемуся возникновением математической (символической) логики. В отличие от традиционной (аристотелевской) логики, математическая логика смогла предложить точные и эффективные методы формального анализа, опирающиеся на концепции, методы и технику математики. Эти методы во многом способствовали возникновению теории алгоритмов, приемов математического моделирования и программирования для решения сложных задач техники, экономики, торговли и транспорта и тем самым развертыванию «компьютерной революции» в мире.

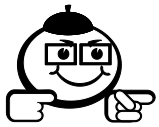
Соглашения, принятые в книге

Для улучшения восприятия материала в данной книге используются пиктограммы и специальное выделение важной информации.

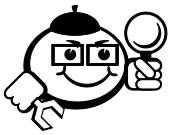
Эта пиктограмма означает определение или новое понятие.

Эта пиктограмма означает «Внимание». Здесь выделена важная информация, требующая акцента на ней. Автор может поделиться с читателем опытом, чтобы помочь избежать некоторых ошибок.

Эта пиктограмма означает теорему.



Эта пиктограмма означает лемму.



Пример

Эта пиктограмма означает пример. В данном блоке автор может привести практический пример для пояснения и разбора основных моментов, отраженных в теоретическом материале.

Контрольные вопросы по главе

1 Алгебра высказываний

1.1 Аксиоматический метод в математике

Аксиома *– основное положение рассматриваемой теории, принимаемое без доказательств.*

Аксиома является истинным исходным положением теории. Такой способ построения научной теории в виде системы аксиом (постулатов) и правил вывода (аксиоматики) позволяет путем дедукции, т. е. по правилам логики, получать утверждения данной теории.

Аксиоматический метод не является достижением двадцатого столетия [5]. В начале XX в. благодаря главным образом работам немецкого математика Д. Гильберта (1862–1943) окончательно сформировались принципиальные положения данного метода и было осознано его значение для математики. Первые идеи, связанные с этим методом, восходят к титанам античной мысли Платону и Аристотелю (IV в. до н. э.). Внутри математической науки взгляд на аксиомы претерпевал самые решительные изменения. Процесс шел постепенно, но качественный скачок в нем произошел после того, как в 20–30-е гг. XIX в. великим русским математиком Н. И. Лобачевским (1792–1856) и независимо от него молодым венгром Яношем Бояи (1802–1860), а также великим немецким ученым К. Ф. Гауссом (1777–1855) были внесены изменения в представления о природе пространства, т. е. возникakf неевклидова геометрия. Суть открытия состояла в том, что вместо пятого постулата Евклида о параллельных в систему аксиом было включено утверждение, являющееся его отрицанием, и затем на базе полученной системы аксиом была построена непротиворечивая геометрическая теория, названная Н. И. Лобачевским «воображаемой геометрией».

Важным этапом в процессе эволюции взглядов на аксиомы явилось построение во второй половине XIX в. нескольких моделей геометрии Лобачевского. Оказалось, что терминам, входящим в аксиомы, и самим аксиомам можно придавать разный смысл, а не только тот наглядный, который имел в виду Евклид.

Такое развитие взглядов на природу аксиом и аксиоматический метод привело к следующей концепции *аксиоматической теории*. Выбирается ряд

первоначальных понятий, которые не определяются и используются без объяснения их смысла. Вместе с тем все другие понятия, которые будут использоваться, должны быть строго определены через первоначальные неопределяемые понятия и через понятия, смысл которых был определен ранее.

Высказывание, определяющее таким способом значение понятия, называется определением, а само понятие, смысл которого определен, носит название определяемого понятия. Евклид сделал попытку строго определить все первоначальные понятия геометрии: точки, прямые, плоскости и т. д. Но эти понятия также должны определяться через свои понятия, которые, в свою очередь, опираются на следующие понятия, и так до бесконечности. Таким образом, первоначальные понятия аксиоматической теории не определяются. Одной из основных причин развития математической логики является широкое распространение аксиоматического метода в построении различных математических теорий, в первую очередь геометрии, а затем арифметики, теории групп и т. д. [15].

Аксиоматическая система *– совокупность основных и производных понятий, аксиом и теорем.*

К системе аксиом предъявляется одно главное требование – она должна быть непротиворечивой. Из непротиворечивой системы аксиом нельзя логическим путем вывести два противоречащих друг другу утверждения.

1.2 Краткие сведения из истории

Возникновение логики как науки имело две предпосылки. Во-первых, это зарождение и первоначальное развитие наук. Этот процесс получает развитие в Древней Греции с VI в. до н. э. Зарождение науки требовало исследования природы мышления как средства познания. Во-вторых, возникновение логики было связано с развитием ораторского искусства. Логика должна была объяснить, как должна строиться речь и какими свойствами она должна обладать. Поэтому неслучайно, что именно Греция стала родиной такой науки, как логика. Основателем логики принято считать древнегреческого философа Аристотеля, который изложил свои идеи в работе «Органон». Согласно Аристотелю, «мышление – это не конструирование или создание умом некой новой сущно-

сти, но, скорее, уподобление в акте мышления чему-то, находящемуся вовне» [5].

Классическая логика Аристотеля – суждение истинное, если соответствует действительности (фактам). Логика как наука в трудах Аристотеля связана с судебной и политической практикой.

В Средние века (VI–XIV вв.) логика была в значительной мере подчинена богословию. В этот период теоретический поиск в логике развернулся вокруг проблемы объяснения общих понятий – универсалий. При этом на всем протяжении Средних веков систематическая разработка формальной логики почти не выходила за пределы силлогистики. Основателем арабо язычной логики считается сирийский математик Аль-Фараби. Его логика направлена на анализ научного мышления. Аль-Фараби выделял в логике две ступени: одна охватывала представления и понятия, другая – теорию суждений, выводов и доказательств. В эпоху Возрождения логика переживала настоящий кризис. Она расценивалась в качестве логики «искусственного мышления», основанного на вере, которому противопоставлялось естественное мышление, базирующееся на интуиции и воображении.

Новый, более высокий этап в развитии логики начинается с XVII в. Его начало было связано с появлением работы Ф. Бэкона «Новый Органон». В этом труде автор стремился разработать приемы исследования самой природы. Он положил начало созданию механизмов установления причинно-следственных связей в объективной реальности. Таким образом, Ф. Бэкон стал родоначальником индуктивной логики, в которой нашли отражение процессы получения новых общих знаний на основе данных, полученных путем эмпирических исследований.

Растущие успехи в развитии математики выдвинули две фундаментальные проблемы: применение логики для разработки математических теорий. Попытку решения этих проблем впервые предпринял Готфрид Лейбниц. В 1666 г., применив аппарат алгебры, он придал новый импульс логическим исследованиям.

В XIX в. была создана символическая логика. Символическая логика изучает символические абстракции, которые фиксируют формальную структуру логического вывода. В алгебраическом духе прогресс периодически возобновлялся, достигнув кульминационных точек в 1847–1877 гг. в работах Дж. Буля, О. де Моргана, Ч. С. Пирса и Э. Шредера.

Дж. Буль предложил логику рассуждений безотносительно к содержанию определить символическим языком формальной логики, утверждениям присвоить абстрактные значения True (истина) или False (ложь). Причем следует отметить, что при изучении проблемы взаимодействия логики и алгебры приоритет всегда отдавался алгебре. Более того, указанные ученые стремились скорее не синтезировать эти науки, а полностью подчинить логику математике.

И только Г. Фреге в 1879 г. отказался от алгебраических аналогий и разработал оригинальный символический и понятийный аппарат, пригодный для использования в универсальной и эффективной логической теории. Только отойдя от полного подражания алгебре, Г. Фреге выяснил истинную природу центрального понятия алгебры и логики – переменной. Обнаружилось родство между переменной и неопределенным местоимением.

Продолжением развития символической логики занимались Б. Рассел и А. Н. Уайтхед. Новая логика позволила с большой точностью описать формы суждений и отношения между ними. На целый ряд философских вопросов, в частности касавшихся природы математики, были сразу даны новые и четкие ответы, и стало казаться, что с помощью формальной логики можно будет найти окончательное решение философских проблем. В современной науке значение символической логики очень велико. Она находит приложение в кибернетике, нейрофизиологии, лингвистике. Символическая логика является современным этапом в развитии формальной логики. Она изучает процессы рассуждения и доказательства посредством его отображения в логических системах (исчислениях). Таким образом, по своему предмету эта наука является логикой, а по методу – математикой [5] .

Непосредственным результатом революции, произошедшей в логике в конце XIX – начале XX в., было возникновение логической теории, получившей название *математическая логика*.

Математическая логика *– естественнонаучная дисциплина, изучающая математические доказательства и вопросы оснований математики.*

Характерным для этой дисциплины является использование формальных языков с точным синтаксисом и четкой семантикой, однозначно определяющими понимание формул. Современная математическая логика – это та же самая

логика Аристотеля, но только громоздкие словесные выводы заменены в ней математической символикой. Эта дисциплина изучает вопросы применения математических методов для решения логических задач и построения логических схем, которые лежат в основе работы любого компьютера. Для построения основных разделов современной математической логики существуют два подхода, образующих два варианта формальной логики: логику (алгебру) высказываний и логические исчисления. В данном пособии рассматривается только первый из этих подходов.

1.3 Высказывания и логические операции

Алгебра высказываний *– раздел математической логики, занимающийся построением и преобразованием высказываний с помощью логических операций, а также изучающий свойства и отношения между ними.*

Основным (неопределяемым) понятием математической логики является понятие «высказывание».

Высказывание *– повествовательное предложение, для которого имеет смысл говорить о его истинности или ложности.*

Логическими значениями высказываний являются «истина» и «ложь».

Для формализации работы с высказываниями их обозначают символами алфавита, например: *A, B, C..., P, Q, R..., X, Y, Z...*, и называют логическими переменными.

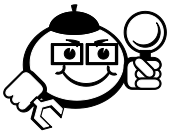
Логическая переменная *– это переменная, обозначающая любое высказывание, которая может принимать логические значения «истина» или «ложь».*

Истинные высказывания обозначают символом *1* (*Истина*, *True*), а ложные – *0* (*Ложь*, *False*).

Введем функцию λ, заданную на совокупности всех высказываний и принимающую значения в двухэлементном множестве $\{0,1\}$ по следующему правилу:

$$\lambda(A)=\begin{cases}1, \textit{ если высказывание } A \textit{ истинно},\\ 0, \textit{ если высказывание } A \textit{ ложно}.\end{cases}$$

Функция λ называется *функцией истинности*, а значение $\lambda(A)$ – логическим значением или значением истинности высказывания A.



Пример 1.1

A: = «1 Кбайт = 1024 байт»; $\lambda(A)=1$.

B: «Париж – столица Англии»; $\lambda(B)=0$.

C: «Карась не рыба»; $\lambda(C)=0$.

D: «Число 6 делится на 2 и на 3»; $\lambda(D)=1$.

E: «Если две параллельные плоскости пересекаются третьей, то прямые пересечения параллельны»; $\lambda(E)=1$.

Не всякое повествовательное предложение является высказыванием, например высказывание «$x>3$», так как неизвестно значение переменной x. Если переменную x заменить каким-либо значением, то выражение станет высказыванием.

Вопросительные и восклицательные предложения не являются высказываниями. Очевидно, предложение «Да здравствуют наши спортсмены!» высказыванием не является.

Высказывание, представляющее собой одно утверждение, принято называть простым или элементарным. Примерами элементарных высказываний могут служить высказывания A и B.

Элементарное высказывание *– высказывание, представляющее собой одно утверждение.*

Высказывания, которые получаются из элементарных с помощью грамматических связок «не», «и», «или», «если…, то…», «тогда и только тогда», принято называть сложными или составными.

Сложное высказывание *– высказывание, образованное из элементарных высказываний с помощью грамматических связок.*

Так, высказывание *C* получается из простого высказывания «Карась – рыба» с помощью отрицания «не», высказывание *D* образовано из элементарных высказываний «Число 6 делится на 2», «Число 6 делится на 3», соединенных союзом «и». Высказывание *E* получается из простых высказываний «две параллельные плоскости пересекаются третьей», «прямые пересечения параллельны» с помощью грамматической связки «если…, то…». Аналогично сложные высказывания могут быть получены из простых высказываний с помощью грамматических связок «или», «тогда и только тогда». Истинность или ложность сложных высказываний зависит от истинности или ложности элементарных высказываний.

В алгебре логики все высказывания рассматриваются только с точки зрения их логического значения. Считается, что каждое высказывание либо истинно, либо ложно и ни одно высказывание не может быть одновременно истинным и ложным. Истинность или ложность высказывания определяется не алгеброй высказываний, а конкретными науками, практикой, наблюдениями. Например, высказывание «сумма углов треугольника равна 180 градусам» истинно в геометрии Евклида, но ложно в геометрии Лобачевского [1].

В алгебре высказываний грамматические связки сложных высказываний рассматриваются как логические операции, имеющие название и обозначение. В таблице 1.1 приведены логические операции и их обозначения.

Для задания логических операций используются *таблицы истинности*, в которых перечисляются все возможные комбинации значений логических переменных и результаты выполнения соответствующих логических операций. Совокупность значений логических переменных называют набором, например: $A = 0$, $B = 1$ образуют набор 01. Наборы логических переменных имеют строгую очередность и записываются всегда в едином виде.

Таблица 1.1 – Логические операции и их обозначения

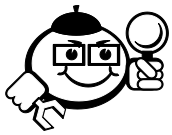
Приоритет логической операции	**Операция**	**Речевые обороты**	**Обозначение**
1.	Отрицание, (инверсия, логическое НЕ)	не A; неверно, что A	$\neg A$ $\overline{A}$
2.	Конъюнкция, (логическое умножение, логическое И)	A и B	$A \& B$ $A \wedge B$ $A \cdot B$
3.	Дизъюнкция, (логическое сложение, логическое ИЛИ)	A или B	$A \vee B$ $A + B$
4.	Импликация (следование)	если A то, B	$A \to B$ (A – посылка, B – следствие)
5.	Эквиваленция (эквивалентность, равнозначность)	A тогда и только тогда, когда B	$A \leftrightarrow B$ $A \equiv B$ $A \sim B$

Отрицанием *(инверсией) высказывания A называется высказывание* $\neg A$*, истинное тогда и только тогда, когда A ложно*.

Таблица истинности операции «отрицание» приведена в таблице 1.2.

Таблица 1.2 – Таблица истинности операции «отрицание»

A	$\neg A$
0	1
1	0



Пример 1.2

Определим результат логической операции отрицания элементарного высказывания: A: «Париж – столица Англии».

Решение.

$\lambda(A)=0$,

$\neg A$: «Неверно, что Париж – столица Англии»;

$\lambda(\neg A)=1$.

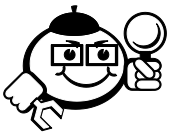
Операция отрицания выполняется над одной логической переменной, поэтому является одноместной или унарной операцией. Все остальные логические операции, перечисленные в таблице 1.1, выполняются над двумя логическими переменными и называются двуместными или бинарными.

Конъюнкцией *двух высказываний A и B называется высказывание* $A \wedge B$*, истинное тогда и только тогда, когда истинны оба высказывания A и B.*

Таблица истинности операции «конъюнкция» приведена в таблице 1.3.

Таблица 1.3 – Таблица истинности операции «конъюнкция»

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1



Пример 1.3

Определим результат логической операции конъюнкции следующих элементарных высказываний:

A: «Число 6 делится на 2»;

B: «Число 6 делится на 3».

Решение.

$\lambda(A)=1;\ \lambda(B)=1;$

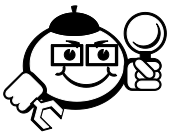
$A \wedge B$: «Число 6 делится на 2 и на 3», $\lambda(A \wedge B)=1 \wedge 1=1.$

Дизъюнкцией *двух высказываний A и B называется высказывание* $A \vee B$*, ложное в том и только в том случае, когда оба высказывания A и B ложны.*

Таблица истинности операции «дизъюнкция» приведена в таблице 1.4.

Таблица 1.4 – Таблица истинности операции «дизъюнкция»

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1



Пример 1.4

Определим результат логической операции дизъюнкции следующих элементарных высказываний:

A: «Число 7 четное»;

B: «Число 7 нечетное».

Решение.

$\lambda(A) = 0; \quad \lambda(B) = 1.$

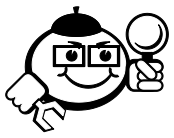
$A \vee B$: «Число 7 четное или нечетное»; $\lambda(A \vee B) = 0 \vee 1 = 1.$

Импликацией *двух высказываний A и B называется высказывание* $A \to B$*, ложное тогда и только тогда, когда A истинно, а B – ложно.*

Логические переменные в операции «импликация» имеют специальные названия: A – посылка, B – заключение. Таблица истинности этой операции приведена в таблице 1.5.

Таблица 1.5 – Таблица истинности операции «импликация»

A	B	$A \to B$
0	0	1
0	1	1
1	0	0
1	1	1



Пример 1.5

Определим результат логической импликации следующих элементарных высказываний:

A: «Число 7 нечетное»; $\lambda(A)=1.$

B: «Число 7 не кратно 2»; $\lambda(B)=1.$

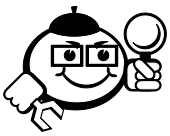
$A \to B$: «Если число 7 нечетное, то число 7 не кратно 2»; $\lambda(A \to B)=1 \to 1=1.$

Эквиваленцией *двух высказываний A и B называется высказывание* $A \leftrightarrow B$*, истинное тогда и только тогда, когда истинностные значения A и B одинаковы.*

Таблица истинности операции «эквиваленция» приведена в таблице 1.6.

Таблица 1.6 – Таблица истинности операции «эквиваленция»

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1



Пример 1.6

Определим результат логической операции эквиваленции следующих элементарных высказываний:

A: «Число 4 четное»;

B: «Число 4 кратно 2».

Решение.

$\lambda(A)=1;\ \lambda(B)=1.$

$A \leftrightarrow B$: «Число 4 четное тогда и только тогда, когда оно кратно 2» $\lambda(A \leftrightarrow B)=1 \leftrightarrow 1=1.$

Все логические операции легко задать с помощью сводной таблицы истинности, указав значение результата операции в зависимости от значений логических переменных и операций над ними (табл. 1.7).

Таблица 1.7 – Сводная таблица истинности логических операций

Логические переменные		**Логические операции**				
		Отрицание	**Конъюнкция**	**Дизъюнкция**	**Импликация**	**Эквиваленция**
A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

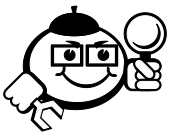
Число строк (наборов логических переменных) в таблице истинности зависит от числа N логических переменных и равно 2^N.

1.4 Формулы алгебры высказываний

Сложное логическое высказывание можно представить в виде логической формулы, состоящего из логических констант «истина», «ложь», логических переменных, знаков логических операций и скобок. Таким образом, алфавит формул алгебры высказыванием содержит:

1) логические константы: 1, 0;
2) логические переменные: $A, B, C...$;
3) логические операции: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$;
4) скобки, меняющие приоритет логических операций.

Формулы логики высказываний принимают значение «истина» или «ложь» в зависимости от логических значений входящих в них элементарных высказываний.



Пример 1.7

Определим логическое значение сложных формул $(A \wedge B) \rightarrow C$ и $C \leftrightarrow (A \vee B)$, если даны:

A: «Число 24 делится на 2»;

B: «Число 24 делится на 3»;

C: «Число 24 делится на 6».

Решение.

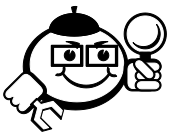
$$\lambda(A) = 1;\ \lambda(B) = 1;\ \lambda(C) = 1.$$

$(A \wedge B) \rightarrow C$: «Если число 24 делится на 2 и делится на 3, то оно делится на 6»; $\lambda\big((A \wedge B) \rightarrow C\big) = (1 \wedge 1) \rightarrow 1 = 1 \rightarrow 1 = 1.$

$C \leftrightarrow (A \vee B)$: «Число 24 делится на 6 тогда и только тогда, когда число 24 делится на 2 или оно делится на 3»; $\lambda\big(C \leftrightarrow (A \vee B)\big) = 1 \leftrightarrow (1 \vee 1) = 1 \leftrightarrow 1 = 1.$

Для каждой формулы должна существовать конечная последовательность всех ее подформул. Так, в формуле $C \leftrightarrow (A \vee B)$ содержится 4 подформулы: $A, B, C, A \vee B$.

Для формул алгебры высказываний можно построить таблицу истинности и узнать, как зависит логическое значение формулы от логических значений входящих в нее переменных.



Пример 1.8

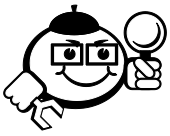
Выпишем все подформулы формулы

$$\left(\left(A_0 \to A_1\right) \& \left(A_1 \to A_2\right)\right) \to \overline{A_0}\,.$$

Решение.

Выписываем все логические переменные $A_0; A_1; A_2$, затем отрицания логических переменных $\overline{A_0}$ и далее логические выражения $(A_0 \to A_1)$; $(A_1 \to A_2)$; $(A_0 \to A_1) \& (A_1 \to A_2)$.

В формуле $F = \left(\left(A_0 \to A_1\right) \& \left(A_1 \to A_2\right)\right) \to \overline{A_0}$ содержится 7 подформул.



Пример 1.9

Построим таблицу истинности для формулы $C \leftrightarrow (A \vee B)$.

Решение.

Шаг 1: определяем число входящих в формулу логических переменных, в данном случае переменных 3, следовательно наборов логических переменных (строк в таблице истинности) будет $2^3 = 8$.

Шаг 2: определяем приоритет логических операций. Первой логической операцией будет операция конъюнкции переменных *A* и *B*. Вторым действием будет операция эквиваленции переменной *C* и результата предыдущего действия.

$$\overbrace{}^{2}$$
$$\overbrace{}^{1}$$
$$F = C \leftrightarrow (A \vee B)$$

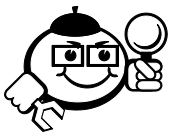
Шаг 3: заполняем таблицу истинности. Рассмотрим заполнение первой строки таблицы истинности. Первому набору логических переменных соответствуют значения $A = 0, B = 0, C = 0$. Первое действие возвращает значение $\lambda(A \vee B) = 0 \vee 0 = 0$, записываем его в соответствующий столбец (действие 1). Второе действие зависит от результата первого действия и возвращает значение $\lambda(C \leftrightarrow 0) = 0 \leftrightarrow 0 = 1$.

Номер набора логических переменных	**Подформулы**				
	A	***B***	***C***	***1***	***F***
1	0	0	0	0	1
2	0	0	1	0	0
3	0	1	0	1	0
4	0	1	1	1	1
5	1	0	0	1	0
6	1	0	1	1	1
7	1	1	0	1	0
8	1	1	1	1	1

Из таблицы истинности видим, что $\lambda\big(C \leftrightarrow (A \vee B)\big) = 1$ на наборах 1, 4, 6, 8 и ложна на наборах 2, 3, 5 и 7.

Формулы в логике высказываний разделяют на несколько видов.

Тождественно истинные формулы (тавтологии) *– формулы, принимающие значение «истина» на всех наборах логических переменных.*



Пример 1.10

Докажем, что $F = A \vee \neg A$ является тождественно истинной формулой.

Решение.

$$\overbrace{}^{2}_{1}$$

Построим таблицу истинности для формулы $F = A \vee \neg A$.

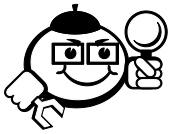
A	$\neg A$	F
0	1	1
1	0	1

Из таблицы истинности видно, что F принимает значение «истина» на всех наборах логических переменных. Следовательно, данная формула относится к классу тождественно истинных формул.

В данном примере всего одна логическая переменная, поэтому доказательство можно оформить и без построения таблицы истинности.

$$F = \begin{cases} \lambda(A) = 0: \ 0 \vee \bar{0} = 0 \vee 1 = 1. \\ \lambda(A) = 1: \ 1 \vee \bar{1} = 1 \vee 0 = 1. \end{cases}$$

Тождественно ложные формулы (противоречия) *– формулы, принимающие значение «ложь» на всех наборах логических переменных.*



Пример 1.11

Докажем, что $F = A \wedge \neg A$ является тождественно ложной формулой.

Решение.

$$\overbrace{}^{2}_{1}$$

Построим таблицу истинности для формулы $F = A \wedge \neg A$.

A	$\neg A$	F
0	1	0
1	0	0

Из таблицы истинности видно, что F принимает значение «ложь» на всех наборах логических переменных. Следовательно, данная формула относится к классу тождественно истинных формул.

Выполнимые формулы *– формулы, принимающие значение «истина» хотя бы на одном наборе логических переменных.*

Примером выполнимой формулы являются формула из примера 1.9.

1.5 Логическая равносильность формул

Равносильные формулы *– формулы, принимающие одинаковые истинностные значения при любых наборах своих логических переменных.*

Равносильность формул в алгебре высказываний обозначается знаком тождественного равенства $F_1(A,B,C...) \equiv F_2(A,B,C...)$.

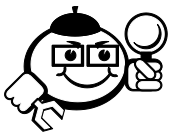
Определение равносильности формул можно записать символически:

$$F_1 \equiv F_2 \Leftrightarrow \lambda\left(F_1(A,B,C...)\right) = \lambda\left(F_2(A,B,C...)\right),$$

для любых высказываний $A,B,C...$.

Стандартный метод установления равносильности двух формул:

1) по каждой формуле строится таблица истинности;
2) полученные таблицы сравниваются по каждому набору значений переменных.



Пример 1.12

Проверим равносильность формул $F_1 = \neg A \vee B$, $F_2 = A \to B$.

Решение.

Построим таблицу истинности для формул $F_1 = \overbrace{\neg A}^{1} \vee B$ $F_2 = \overbrace{A \to B}^{2}$:

A	B	$\neg A$	F_1
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

A	B	F_2
0	0	1
0	1	1
1	0	0
1	1	1

Сравнивая два последних столбца таблиц истинности, убеждаемся, что они полностью совпадают, т. е. $F_1 \equiv F_2$.

Рассмотрим алгоритм проверки двух формул на равносильность. Пусть формулы заданы следующей таблицей истинности:

A	B	C	$F_1(A, B, C)$	$F_2(A, B, C)$
0	0	0	α_0	β_0
0	0	1	α_1	β_1
0	1	0	α_2	β_2
0	1	1	α_3	β_3
1	0	0	α_4	β_4
1	0	1	α_5	β_5
1	1	0	α_6	β_6
1	1	1	α_7	β_7

На рисунке 1.1 приведена блок-схема алгоритма проверки формул на равносильность.

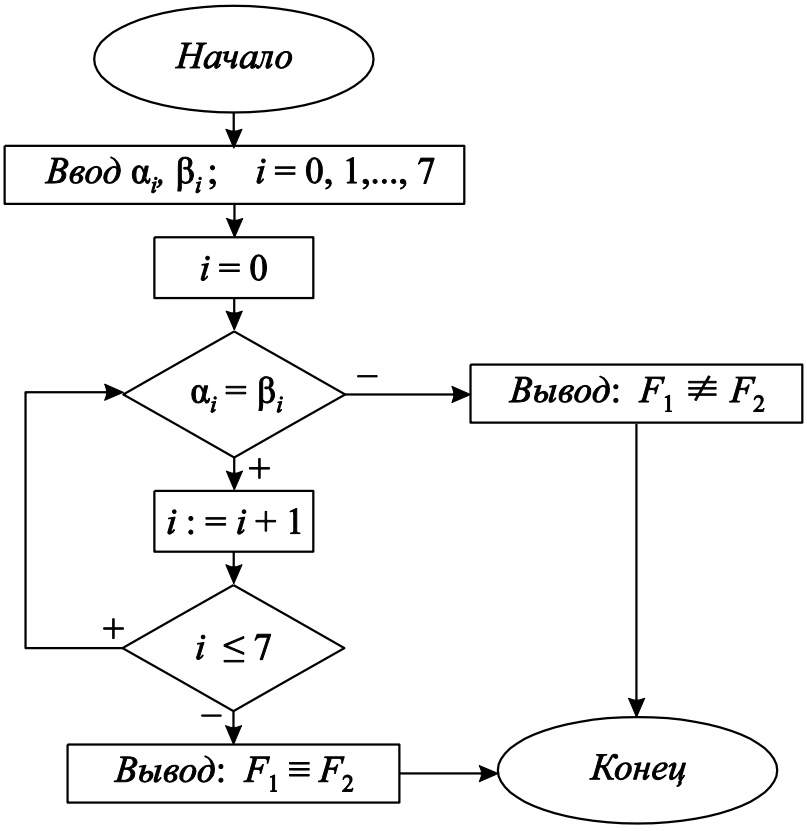


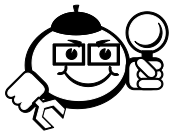
Рис. 1.1 – Блок-схема алгоритма проверки формул на равносильность

Не следует думать, что в обе формулы F_1 и F_2 непременно входят одни и те же переменные. Некоторые из переменных $A, B, C...$ могут отсутствовать

в любой из них. В таком случае для проверки равносильности формул удобнее воспользоваться признаком равносильности формул:

Две формулы F_1 и F_2 алгебры высказываний равносильны тогда и только тогда, когда формула $F_1 \leftrightarrow F_2$ является тождественно истинной формулой.

$$F_1 \equiv F_2 \Leftrightarrow \lambda(F_1 \leftrightarrow F_2) \equiv 1.$$



Пример 1.13

Проверим равносильность формул $F_1 = \neg A$, $F_2 = \neg A \wedge (B \vee \neg A)$.

Решение.

Построим таблицу истинности для формулы

$$\overbrace{F_1 \leftrightarrow F_2}^{4} = \overbrace{\neg A}^{3} \leftrightarrow (\overbrace{\neg A}^{2} \wedge (\overbrace{B \vee \neg A}^{1})).$$

A	B	$\neg A$	2	3	4
0	0	1	1	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	1	0	1	0	1

Согласно признаку равносильности $F_1 \equiv F_2$, так как на всех наборах логических переменных $\lambda(F_1 \leftrightarrow F_2) = 1$.

Отношение равносильности на множестве формул логики высказываний является отношением эквивалентности, так как обладает свойствами рефлексивности ($F \equiv F$ для любой формулы F), симметричности (если $F_1 \equiv F_2$, то $F_2 \equiv F_1$), транзитивности (если $F_1 \equiv F_2$ и $F_2 \equiv F_3$, то $F_1 \equiv F_3$). С помощью этого

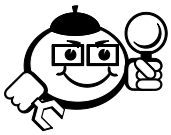
отношения все множество формул логики высказываний разбивается на классы равносильных формул.

Основные равносильности (табл. 1.8) – законы логики высказываний – доказываются с помощью таблиц истинности.

Таблица 1.8 – Основные равносильности логики высказываний

№	Формула	Название
1	$A \vee \neg A \equiv 1$	Закон исключенного третьего
2	$A \wedge \neg A \equiv 0$	Закон противоречия
3	$A \wedge B \equiv B \wedge A,\ A \vee B \equiv B \vee A$	Законы коммутативности
4	$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ $(A \vee B) \vee C \equiv A \vee (B \vee C)$	Законы ассоциативности
5	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$	Законы дистрибутивности
6	$\neg\neg A \equiv A$	Закон двойного отрицания
7	$A \wedge A \equiv A,\ \ A \vee A \equiv A$	Закон идемпотентности
8	$\neg(A \vee B) \equiv \neg A \wedge \neg B$ $\neg(A \wedge B) \equiv \neg A \vee \neg B$	Законы де Моргана
9	$A \vee (A \wedge B) \equiv A,\ A \wedge (A \vee B) \equiv A$	Законы поглощения
10	$(A \wedge B) \vee (A \wedge \neg B) \equiv A$ $(A \vee B) \wedge (A \vee \neg B) \equiv A$	Законы склеивания
11	$A \to B \equiv \neg A \vee B$	Замена импликации
12	$A \leftrightarrow B \equiv (A \wedge B) \vee (\neg A \wedge \neg B)$	Замена эквиваленции

Обратите внимание на сходство основных равносильностей логики высказываний и законов алгебры множеств. Это объясняется тем, что алгебра множеств и алгебра высказываний изоморфны, т. е. одинаково организованы с точки зрения математики, обе они являются булевыми алгебрами [14].



Пример 1.14

Пользуясь основными равносильностями логики высказываний (табл. 1.8), покажем, что формулы $F_1 = \neg(A \to B) \vee (B \to \neg A)$ и $F_2 = \neg(A \wedge B)$ равносильны.

Решение.

Преобразуем первую формулу:

$$F_1 \overset{11}{\equiv} \neg(\neg A \vee B) \vee (\neg B \vee \neg A) \overset{8}{\equiv} ((\neg\neg A) \wedge \neg B) \vee (\neg B \vee \neg A) \overset{6,4}{\equiv}$$

$$\equiv ((A \wedge \neg B) \vee \neg B) \vee \neg A \overset{9}{\equiv} \neg B \vee \neg A \overset{8}{\equiv} \neg(B \wedge A) \overset{3}{\equiv} \neg(A \wedge B) = F_2.$$

Формула F_2 получена из формулы F_1 цепочкой равносильных преобразований (над знаком равносильности указан номер применяемого закона из табл. 1.8), следовательно, $F_1 \equiv F_2$.

. . . . . .

1.6 Нормальные формы записи формул алгебры высказываний

Для каждой формулы алгебры высказываний можно указать равносильную ей формулу, содержащую из логических связок лишь отрицание, конъюнкцию и дизъюнкцию.

. . . . . .

Элементарная дизъюнкция (ЭД) *– дизъюнкция k логических переменных или их отрицаний* $(k \geq 1)$.

. . . . . .

Например: A, $A \vee \neg B$, $\neg A \vee B \vee C$ – элементарные дизъюнкции.

. . . . . .

Элементарная конъюнкция (ЭК) *– конъюнкция k логических переменных или их отрицаний* $(k \geq 1)$.

. . . . . .

Например: A, $A \vee \neg B$, $\neg A \vee B \vee C$ – элементарные дизъюнкции.

. . . . . .

Дизъюнктивная нормальная форма (ДНФ) *– форма записи алгебры высказываний, представленная в виде дизъюнкции элементарных конъюнкций.*

. . . . . .

Примеры формул в виде ДНФ: A; $(A \wedge C) \vee \neg B$; $(\overline{A} \wedge \overline{B}) \vee (B \wedge D)$.

Конъюнктивная нормальная форма (КНФ) *– форма записи алгебры высказываний, представленная в виде конъюнкции элементарных дизъюнкций.*

Примеры формул в виде КНФ: A; $(A \vee C) \wedge \neg B$; $(\overline{A} \vee \overline{B}) \wedge (B \vee D)$.

Любую формулу логики высказываний можно привести равносильными преобразованиями к ДНФ и КНФ.

Рассмотрим построение ДНФ для данной формулы F.

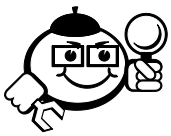
Шаг 1. Построить $F_1 \equiv F$ такую, что F_1 содержит только связки $\neg, \wedge, \vee$. Воспользоваться равносильностями 11, 12 (табл. 1.8).

Шаг 2. Преобразовать F_1 к $F_2 \equiv F_1$ такой, что в F_2 знак $\neg$ стоит только перед логическими переменными. Воспользоваться законом де Моргана.

Шаг 3. Преобразовать F_2 к $F_3 \equiv F_2$, пользуясь первым дистрибутивным законом $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$; при необходимости использовать законы идемпотентности, коммутативности, ассоциативности.

Формула $F_3 \equiv F$ в силу транзитивности отношения $\equiv$ и записана в ДНФ.

Для приведения формулы F к КНФ необходимо и достаточно выполнить *шаг 1* и *шаг 2* так же, как для построения ДНФ, а затем применить второй дистрибутивный закон $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ и сгруппировать в скобки.



Пример 1.15

Приведем к ДНФ формулу.

Шаг 1. Применяем закон замены импликации: $F = \neg(X \,\&\, Y) \vee \neg Y = F_1$.

Шаг 2. Применяем закон де Моргана: $F_1 = \neg X \vee \neg Y \vee \neg Y = F_2$.

Шаг 3. Применяем закон идемпотентности: $F_2 = \neg X \vee \neg Y = F_3$.

ДНФ формулы F имеет вид $F_3 = \neg X \vee \neg Y$.

Нормальные формы записи формул позволяют установить, является ли данная формула тождественно истинной (ложной) или нет.

Теорема 1.1. Формула является тавтологией тогда и только тогда, когда ее можно привести к КНФ, в каждую элементарную дизъюнкцию которой одновременно входят переменная и ее отрицание.

Докажем достаточность. Пусть формула $F = F(X_1, X_2, ..., X_n)$ тавтология. Тогда в силу определения КНФ и операции конъюнкции каждая ЭД записанной КНФ формулы $F_1 \equiv F$ также является тавтологией. Проведем доказательство методом от противного.

Предположим, что найдется элементарная дизъюнкция D, такая, что $D \equiv 1$, но ни одна из переменных списка $(X_1, X_2, ..., X_n)$ не входит в D вместе со своим отрицанием. Рассмотрим оценку списка переменных, в которой положим $X_i \equiv 1$, если переменная X_i входит в D со знаком отрицания, и $X_i \equiv Л$ в противном случае $\left(i = \overline{1,n}\right)$. На этой оценке $D = 0$, т. е. получили противоречие, значит, наше предположение неверно.

ДНФ (КНФ) представляет формулу неоднозначно. Частным случаем ДНФ является совершенная ДНФ (СДНФ) – однозначное представление формул логики высказываний.

Пусть формула F зависит от списка переменных $(X_1, X_2, ..., X_n)$. Полная элементарная конъюнкция (ПЭК) – элементарная конъюнкция, содержащая все логические переменные этого списка (или их отрицания) ровно по одному разу, причем в том порядке, в котором они перечислены в этом списке.

ДНФ называется совершенной относительно списка переменных $(X_1, X_2, ..., X_n)$, если все ее ЭК полные относительно этого списка переменных и различные.

Совершенная ДНФ (СДНФ) *– ДНФ, в которой нет одинаковых элементарных конъюнкций и все конъюнкции состоят из одно-*

го и того же набора переменных, в который каждая переменная входит только один раз (возможно с отрицанием).

. . . . . .

Пример записи формулы с СДНФ: $\left(A\&B\&\overline{C}\right)\vee\left(A\&B\&C\right)$.

. . . . . .

Совершенная КНФ (СКНФ) *– КНФ, в которой нет одинаковых элементарных дизъюнкций и все дизъюнкции состоят из одного и того же набора переменных, в который каждая переменная входит только один раз (возможно с отрицанием).*

. . . . . .

Пример записи формулы с СКНФ: $\left(A\vee B\vee\overline{C}\right)\&\left(A\vee B\vee C\right)$.

. . . . . .

*Теорема 1.2 (о существовании СДНФ).*Пусть формула F_1 не является тождественно ложной и зависит от списка переменных $\left(X_1, X_2, ..., X_n\right)$. Тогда существует равносильная ей формула F_2, находящаяся в СДНФ относительно этого списка переменных.

. . . . . .

. . . . . .

Доказательство. Известно, что для формулы F_1 существует равносильная ей F, находящаяся в ДНФ. Формула F, так же, как и F_1, зависит от списка $\left(X_1, X_2, ..., X_n\right)$ (в процессе приведения к ДНФ новые переменные не появляются). Рассмотрим элементарные конъюнкции формулы F.

Пусть в некоторую ЭК одновременно входят переменная X_i и ее отрицание $\neg X_i$. Если это единственная элементарная конъюнкция формулы F, то F при всех оценках списка переменных должна принимать значение «ложь», что невозможно (по условию теоремы $F_1 \neq 0$). Следовательно, формула F может быть приведена к виду $F=\left(X_i\,\&\neg X_i\,\&K\right)\vee D$, где K – остальные члены элементарной конъюнкции, D – остальные члены ДНФ. Но $F=\left(X_i\,\&\neg X_i\,\&K\right)\vee D\equiv D$, т. е. ЭК, содержащую переменную

и ее отрицание можно отбросить, при этом всегда останутся неотброшенными какие-либо ЭК.

Пусть в некоторой ЭК формулы *F* несколько раз встречается переменная X_i (или ее отрицание $\neg X_i$). В силу идемпотентности можно оставить только одно вхождение X_i ($\neg X_i$).

В результате этих преобразований каждая ЭК будет содержать каждую переменную X_i (либо ее отрицание) не более одного раза.

При этом возможны следующие варианты:

1) ЭК содержит один раз X_i и ни разу $\neg X_i$;
2) ЭК содержит один раз $\neg X_i$ и ни разу X_i;
3) ЭК не содержит ни $\neg X_i$, ни X_i.

В последнем случае применим закон склеивания (формула 10 табл. 1.8): $ЭК \equiv (ЭК \,\&\, X) \vee (ЭК \,\&\, \neg X)$. Это преобразование будет выполняться до тех пор, пока каждая ЭК не станет полной относительно списка переменных формулы *F*. Теперь, используя коммутативность конъюнкции, расположим переменные в каждой ЭК в порядке возрастания индексов и сравним все ЭК между собой. Если получилось несколько одинаковых ЭК, применяя закон идемпотентности, оставим только по одному представителю каждой ЭК. Полученная формула F_2 и есть СДНФ формулы F_1. Действительно: $F_2 \equiv F_1$ и является дизъюнкцией полных ЭК, причем все ЭК – различные. Теорема доказана.

. . . . . .

Доказательство теоремы дает нам способ построения СДНФ для всякой формулы, не являющейся противоречием.

. . . . . .

Теорема 1.3 (о существовании СКНФ). Пусть формула $F_1 \neq И$ и зависит от списка переменных $(X_1, X_2, ..., X_n)$. Тогда существует равносильная ей формула F_2, находящаяся в СКНФ относительно этого списка переменных.

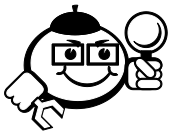
. . . . . .

Алгоритм получения СДНФ по таблице истинности:

Шаг 1: Отметить те строки таблицы истинности, в которых формула принимает значение «истина» ($\lambda(F) = 1$).

Шаг 2: Выписать для каждой отмеченной строки конъюнкцию всех переменных следующим образом: если значение некоторой переменной в данной строке равно «истине», то в конъюнкцию включают саму эту переменную, иначе – ее отрицание.

Шаг 3: Все полученные конъюнкции связать в дизъюнкцию.



Пример 1.16

Построим СДНФ формулы $F = \neg A \leftrightarrow \neg B$.

Строим таблицу истинности:

A	B	$\neg A$	$\neg B$	F	$ЭК$
0	0	1	1	1	$\neg A \wedge \neg B$
0	1	1	0	0	–
1	0	0	1	0	–
1	1	0	0	1	$A \wedge B$

Формула принимает значение «истина» на двух наборах логических переменных (выделены в таблице цветом). Для выделенных наборов записываем элементарные конъюнкции (*шаг 2* алгоритма получения СДНФ). На первом наборе $F = \neg A \leftrightarrow \neg B = 1$, причем $A = B = 0$, следовательно в ЭК обе логические переменные входят в ЭК с отрицаниями. На четвертом наборе $F = \neg A \leftrightarrow \neg B = 1$, причем $A = B = 1$, следовательно в ЭК обе логические переменные входят в ЭК без отрицаний.

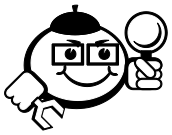
Все полученные конъюнкции связываем операцией дизъюнкция и получаем СДНФ: $F = \left(\neg A \wedge \neg B\right) \vee \left(A \wedge B\right)$.

Алгоритм получения СКНФ по таблице истинности:

Шаг 1: Отметить те строки таблицы истинности, в которых формула принимает значение «ложь» ($\lambda(F) = 0$).

Шаг 2: Выписать для каждой отмеченной строки дизъюнкцию всех переменных следующим образом: если значение некоторой переменной в данной строке равно «ложь», то в дизъюнкцию включают саму эту переменную, иначе – ее отрицание.

Шаг 3: Все полученные дизъюнкции связать в конъюнкцию.



Пример 1.17

Построим СКНФ формулы $F = \neg A \leftrightarrow \neg B$.

Строим таблицу истинности:

A	B	$\neg A$	$\neg B$	F	*ЭД*
0	0	1	1	1	–
0	1	1	0	0	$A \vee \neg B$
1	0	0	1	0	$\neg A \vee B$
1	1	0	0	1	–

Формула принимает значение «ложь» на двух наборах логических переменных (выделены в таблице цветом). Для выделенных наборов записываем элементарные дизъюнкции (*шаг 2* алгоритма получения СКНФ). На втором наборе $F = \neg A \leftrightarrow \neg B = 0$, причем $A = 0$, $B = 1$; следовательно в ЭД переменная A входит без отрицания, а переменная B с отрицанием. На третьем наборе $F = \neg A \leftrightarrow \neg B = 0$, причем $A = 1$, $B = 0$; следовательно в ЭД переменная A входит с отрицанием, а переменная B без отрицания.

Все полученные ЭД связываем операцией конъюнкция и получаем СКНФ: $F = (A \vee \neg B) \wedge (\neg A \vee B)$.

Представить формулу в СДНФ (СКНФ) можно единственным образом.

Теорема 1.4 (о единственности представлении в СДНФ (СКНФ)). Если формулы F_1 и F_2 являются совершенными дизъюнктивными (конъюнктивными) нормальными формами формулы F относительно списка $(X_1, X_2, ..., X_n)$, то они могут отличаться лишь порядком ЭК (ЭД).

Теорема 1.5 (критерий равносильности). Две формулы F_1 и F_2, зависящие от списка $(X_1, X_2, ..., X_n)$ и не являющиеся тождественно ложными (истинными), равносильны тогда и только тогда,

когда они приводятся к СДНФ (СКНФ), отличающимся лишь порядком ЭК (ЭД).

. . . . . .

1.7 Логическое следование формул

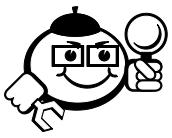
Одна из основных задач математической логики состоит в том, чтобы исследовать, какие следствия вытекают из данных утверждений, например, какие теоремы в математике следуют из принятой системы аксиом. Интуитивно возможно выводить заключения, не обращаясь к логической символике и даже ясно не сознавая те логические правила, которыми неявно пользуемся. Однако в более трудных случаях интуитивных возможностей оказывается недостаточно, в особенности когда приходится проверять рассуждения и анализировать ошибки [12].

Когда говорят, что из одного или нескольких высказываний $P_1, P_2, ..., P_n$ следует предложение D, то подразумевают следующее: всякий раз, когда окажутся истинными все высказывания $P_1, P_2, ..., P_n$, истинным будет и высказывание D. Логическое следование принято обозначать $P \vdash D$, где формула P является посылкой, а D – заключением логического следствия.

Логическое следование $P_1, P_2, ..., P_n, D$ можно записать в виде схемы:

$$\begin{array}{l} P_1 \\ \ldots\ldots \\ Pn \\ \hline D \end{array}$$

.



Пример 1.18

«Если студент знает материал, то может привести доказательство (утверждение P_1). Если студент понимает материал, то может решить задачу (утверждение P_2). На экзамене студент не может привести доказательство или не может решить задачу (утверждение P_3). Следовательно, он не знает или не понимает материал (заключение D)».

Здесь утверждение $P_1 = X \to Y$ является сложным высказыванием, состоящим из двух элементарных: $X =$ «студент знает материал»; $Y =$ «может привести доказательство».

Утверждение $P_2 = Z \to H$ состоит также из двух элементарных высказываний: $Z =$ «студент понимает материал»; $H =$ «может решить задачу».

Утверждение P_3 состоит из двух элементарных высказываний – отрицаний высказываний *Y* и *H*:

- $\neg Y =$«не может привести доказательство».
- $\neg H =$«не сможет решить задачу».

Заключение *D* представлено двумя элементарными высказываниями – отрицаний высказываний *X* и *Z*:

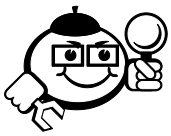
- $\neg X =$«студент не знает материал»;
- $\neg Z =$«студент не понимает материал»;

Схема данного логического следования:

$$\begin{array}{|l} P_1 = X \to Y \\ P_2 = Z \to H \\ P_3 = \neg Y \vee \neg H \quad . \\ \hline D = \neg X \vee \neg Z \end{array}$$

Задача математической логики (в частности, алгебры высказываний) в вопросах логического следования состоит в том, чтобы указать такие формы высказываний $P_1, P_2, ..., P_n, D$, когда последнее высказывание непременно было бы следствием n первых, независимо от конкретного содержания всех этих высказываний.

Таким образом, теория логического следования (в рамках алгебры высказываний) изучает закономерности образования формул $P_1, P_2, ..., P_n, D$, по которым первые *n* из них связаны с последней отношением логического следования.



Пример 1.19

Рассмотрим суждение вида: «Если Иван приложит все усилия к учебе, то он может с отличием окончить университет (утверждение P_1). Если Иван

окончит университет с отличием, то у него есть шанс найти хорошо оплачиваемую работу (утверждение P_2). Следовательно, если Иван приложит усилия к учебе, то у него выше шанс найти хорошо оплачиваемую работу (заключение D)».

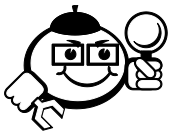
Утверждение P_1 можно записать как $X \to Y$, утверждение P_2: $Y \to Z$, а заключение D: $X \to Z$. Где $X =$ «Иван приложит все усилия к учебе», $Y =$ «Иван с отличием окончит университет», $Z =$ «есть шанс найти хорошо оплачиваемую работу».

Умозаключение в таком виде можно записать:

1) «Если высказывание $A \to B$ верно и высказывание $B \to C$ верно, то верно и высказывание $A \to C$».
2) «Если $\lambda(A \to B) = 1$ и $\lambda(B \to C) = 1$, то $\lambda(A \to C) = 1$».

Рассмотрим определение понятия логического следования:

Из формул $P(X_1, X_2, \ldots, X_n)$ ***логически следует*** *формула* $D(X_1, X_2, \ldots, X_n)$, *если для любого набора логических переменных* $X_1, X_2, \ldots, X_n$ *всякий раз, когда* $P(X_1, X_2, \ldots, X_n) = 1$, *то на этом же наборе* $D(X_1, X_2, \ldots, X_n) = 1$.



Пример 1.20

Проверим, следует ли формула $D = \neg X$ из формулы $P = \neg X \wedge Y$, т. е. $\neg X \,\&\, Y \vdash \neg X$ или $\left| \begin{array}{l} P = \neg X \wedge Y \\ \hline D = \neg X \end{array} \right.$.

Согласно определению логического следования, необходимо построить таблицу истинности и проверить для наборов, где $P = 1$, выполняется ли что на этих же наборах и $D = 1$.

X	Y	$P = \neg X \,\&\, Y$	$D = \neg X$
1	1	0	0
1	0	0	0
0	1	1	1

0	0	0	1

Из таблицы истинности видно, что посылка $P = \neg X \wedge Y$ принимает значение «истина» на одном наборе логических переменных при $X = 0$, $Y = 1$. Заключение $D = \neg X$ на этом наборе также принимает значение «истина». Следовательно, данное логическое следование выполняется и по определению $P \vdash D$.

. . . . . .

В случаях, когда в логических рассуждениях используется не одна посылка P, а несколько $P_1, P_2, \ldots, P_n$, рассуждение будет логически правильным ($P_1, P_2, \ldots, P_n \vdash D$), если $P_1 \wedge P_2 \wedge \ldots \wedge P_n \vdash D$ – из конъюнкции посылок логически следует заключение.

Алгоритм проверки логического следования по определению:

Шаг 1: записать все посылки и заключения в виде формул логики высказываний.

Шаг 2: построить схему логического рассуждения.

Шаг 3: составить конъюнкцию формализованных посылок $P_1 \wedge P_2 \wedge \ldots \ldots \wedge P_n$.

Шаг 4: проверить по таблице истинности, следует ли заключение D из формулы $P_1 \wedge P_2 \wedge \ldots \wedge P_n$.

Рассмотрим алгоритм проверки логического следования в виде блок-схемы. Даны три посылки и заключение, зависящие от двух логических переменных (табл. 1.9).

Таблица 1.9 – Таблица истинности посылок и заключения логического следования

X	Y	P_1	P_2	P_3	D
0	0	α_0	β_0	γ_0	δ_0
0	1	α_1	β_1	γ_1	δ_1
1	0	α_2	β_2	γ_2	δ_2
1	1	α_3	β_3	γ_3	δ_3

Алгоритм действует следующим образом (рис. 1.2): просматривает последовательно по строкам таблицы значений формул P_1, P_2, P_3, D. Если хотя

бы один элемент нулевой строки α_0, β_0, γ_0 равен 0, то без просмотра значения δ_0 формулы D в этой строке (т. е. числа δ_0) происходит переход к просмотру следующей строки α_1, β_1, γ_1.

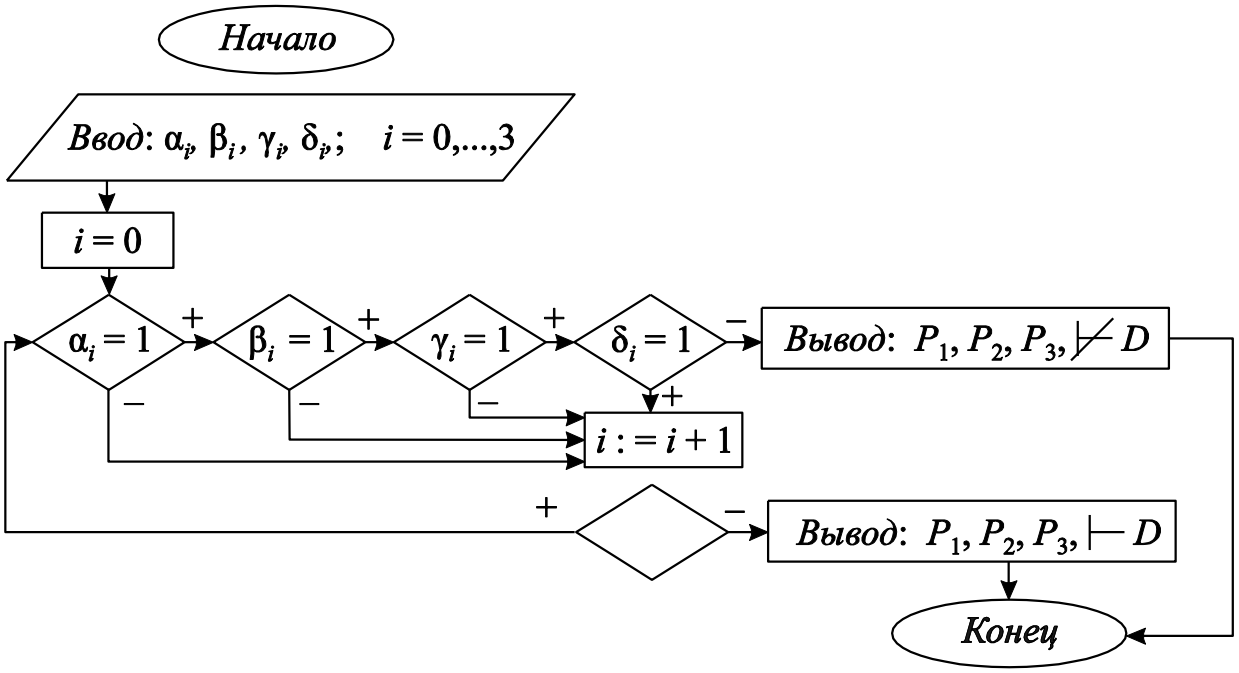
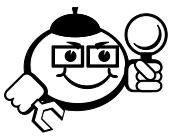


Рис. 1.2 – Блок-схема алгоритма проверки логического следования формул

Если все элементы α_0, β_0, γ_0 нулевой строки равны 1, то просматривается значение δ_0 формулы D в этой строке. При $\delta_0 = 0$ выдается результат: формула D не является логическим следствием формул P_1, P_2, P_3. При $\delta_0 = 1$ происходит переход к просмотру следующей строки α_1, β_1, γ_1. И так далее. Если после просмотра последней строки α_3 , β_3 , δ_3, γ_3 должен произойти переход к просмотру следующей строки, то это означает, что определение логического следования выполнено и формула D является логическим следствием формул P_1, P_2, P_3.



Пример 1.21

Проверим по определению правильность логического следования $X \to Y,\ X \vdash Y$.

Решение.

Запишем схему логического следования.

$$\begin{array}{|l} P_1 = X \to Y \\ P_2 = X \\ \hline D = Y \end{array} .$$

Составим конъюнкцию посылок $P = P_1 \wedge P_2 = \overbrace{(\overbrace{X \to Y}^{1}) \wedge X}^{P}$ и построим таблицу истинности.

X	Y	$X \to Y$	P	D
0	0	1	0	0
0	1	1	0	1
1	0	0	0	0
1	1	1	1	1

Из таблицы истинности видно, что посылка $P = P_1 \wedge P_2 = (X \to Y) \wedge X$ принимает значение «истина» на одном наборе логических переменных при $X = 1,\ Y = 1$. Заключение $D = Y$ на этом наборе также принимает значение «истина». Следовательно, данное логическое следование выполняется и по определению $P = P_1 \wedge P_2 \vdash D$.

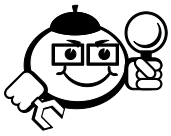
Рассмотренный выше способ проверки логического следования требует построения таблицы истинности и малоэффективен в случаях, когда в рассуждении используется большое количество логических переменных. Поэтому рассмотрим далее ***второй способ проверки логического следования***, основанный на применении законов равносильных преобразований формул алгебры высказываний (табл. 1.8).

Теорема 1.6 (признак логического следования). Формула *D* логически следует из формулы *P* тогда и только тогда, когда формула $P \to D$ является тавтологией.

Необходимость. Дано $P \vdash D$, тогда для всех наборов переменных значение $P = 1$ влечет $D = 1$. Это означает, что для всех наборов переменных $P \to D \equiv 1$, т. к. формула $P \to D$ принимает значение «ложь» только в одном случае: когда $P = 1$, а $D = 0$, но такая ситуация исключена по условию. Следовательно, $P \to D$ – тавтология.

Достаточность. Дано $P \to D$ – тавтология, т. е. $P \to D \equiv 1$. Отсюда по определению операции импликации заключаем: не существует такого набора значений переменных, при котором $P = 1$, а $D = 0$. Значит, $P \vdash D$.

Согласно доказанной теореме проверка правильности логического рассуждения сводится к ответу на вопрос: является ли формула $P \to D$ тавтологией. На этот вопрос можно ответить сведя эту формулу с помощью равносильных преобразований к известной тавтологии.



Пример 1.22

Проверим логическое рассуждение $X \to Y, X \vdash Y$ с помощью признака логического следования. Для этого необходимо применить законы равносильных преобразований логики высказываний (табл. 1.8).

$\overbrace{(X \to Y)}^{11} \wedge X \to Y \equiv$ применим *Закон замены импликации* № 11 (табл. 1.8).

$\equiv \overbrace{[(\neg X \vee Y) \wedge X] \to Y}^{11} \equiv$ *Закон замены импликации*.

$\equiv \overbrace{\neg[(\neg X \vee Y) \,\&\, X]}^{8} \vee Y \equiv$ *Закон де Моргана.*

$\equiv (\overbrace{\neg(\neg X \vee Y)}^{8} \vee \neg X) \vee Y \equiv$ *Закон де Моргана.*

$\equiv (\overbrace{(X \,\&\, \neg Y) \vee \neg X)}^{5} \vee Y \equiv$ *Закон дистрибутивности.*

$\equiv (\overbrace{(X \vee \neg X)}^{1} \wedge (\neg Y \vee \neg X)) \vee Y \overset{1}{\equiv}$ *Закон исключения третьего*.

$\equiv (\overbrace{1 \wedge (\neg Y \vee \neg X)}^{1 \wedge A \equiv A}) \vee Y \equiv$ *свойство операции конъюнкции.*

$\equiv (\overbrace{\neg Y \vee \neg X}^{3}) \vee Y \equiv$ *Закон коммутативности.*

$\equiv \overbrace{(\neg X \vee \neg Y) \vee Y}^{4} \equiv$ *Закон ассоциативности.*

$\equiv \neg X \vee (\neg Y \vee Y) \overset{1}{\equiv}$ *Закон исключения третьего.*

$\equiv \overbrace{\neg X \vee 1}^{1 \vee A \equiv 1} \equiv 1$ *свойство операции дизъюнкции.*

Таким образом, $P \to D \equiv 1$ и согласно признаку логического следования данное рассуждение логически правильно.

. . . . . .

Из примера 1.22 видно, что способ проверки логического следования требует знания законов равносильных преобразований формул логики высказываний (табл. 1.8) и свойств логических операций.

Существует еще один способ проверки правильности логического рассуждения – ***метод*** «***от противного***», который не требует полного перебора значений переменных для построения таблицы истинности и знания законов равносильных преобразований формул логики высказываний.

Для обоснования этого способа сформулируем условие, при котором логическое рассуждение является неправильным.

. . . . . .

Рассуждение является ***неправильным****, если найдется набор значений переменных* $X_1^i, X_2^i, ..., X_n^i$, *такой, что посылка* $P\left(X_1^i, X_2^i, ..., X_n^i\right) = 1$, *а заключение* $D\left(X_1^i, X_2^i, ..., X_n^i\right) = 0$.

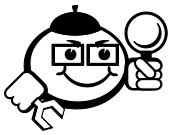
. . . . . .

Метод «от противного» заключается в следующем.

Пусть требуется проверить правильность логического следования формулы D из посылок $P_1, P_2, \ldots, P_n$.

Предположим, что существует набор $X_1^i, X_2^i, ..., X_n^i$, при котором все посылки истинны, а заключение ложно, и попытаемся найти этот набор. Если такой набор будет обнаружен, то наше предположение оправдалось, и рассужде-

ние является логически неправильным. Если в процессе поисков набора придем к противоречию, то наше предположение ошибочно, а рассуждение является логически правильным.



Пример 1.23

Проверим с помощью метода «от противного» логическое следование $X \to Y, X \vdash Y$.

Пусть существует набор X^i, Y^i, при котором посылки истинны, а заключение ложно. Данные условия можно записать в виде системы логических условий:

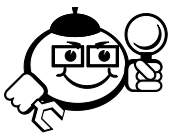
$$\begin{cases} P_1 = X^i \to Y^i = 1 \\ P_2 = X^i = 1 \\ D = Y^i = 0. \end{cases}$$

Если найдется такой набор X^i, Y^i, который удовлетворит всем условиям, то логическое следование не выполняется и рассуждение является неверным.

Из P_2 получаем $X^i = 1$, а из D определяем, что $Y^i = 0$. Осталось проверить еще одно условие системы:

$$\begin{cases} P_1 = X^i \to Y^i = 1 \\ X^i = 1 \\ Y^i = 0. \end{cases}$$

Подставляем найденные логические значения переменных в P_1 и получаем $P_1 = X^i \to Y^i = 1 \to 0 \neq 1$. Найдено противоречие, следовательно, рассуждение $X \to Y, X \vdash Y$ является логически правильным.



Пример 1.24

Проверим при помощью метода «от противного» логическое следование: $X \to Y, X \wedge Y \vdash \neg X$.

Пусть существует набор X^i, Y^i, при котором посылки истинны, а заключение ложно. Данные условия можно записать в виде системы логических условий:

$$\begin{cases} P_1 = X^i \to Y^i = 1 \\ P_2 = X^i \wedge Y^i = 1 \\ D = \neg X^i = 0. \end{cases}$$

По определению конъюнкции двух высказываний из P_2 можно определить, что $X^i = 1$. Из D определяем, что $X^i = 1$. Противоречий в этих двух условиях нет, осталось проверить условие, что $P_1 = 1$:

$$\begin{cases} P_1 = X^i \to Y^i = 1 \\ X^i = 1, Y^i = 1 \\ X^i = 1. \end{cases}$$

Подставляем найденные логические значения переменных в P_1 и получаем $P_1 = X^i \to Y^i = 1 \to 1 = 1$. Это условие тоже выполняется, и противоречий не найдено.

Таким образом, найден набор $X^i = 1, Y^i = 1$, на котором все посылки истинны, а заключение ложно. Следовательно, из посылок $X \to Y, X \wedge Y$ не следует заключение $\neg X$ и данное рассуждение логически неправильно.

. . . . . .

Логически правильное рассуждение можно построить, пользуясь уже готовыми логически правильными схемами рассуждений – правилами вывода.

1. ***Правило отделения (modus ponens):*** $\dfrac{F, F \to G}{G}$.

Это правило означает, что от утверждения об истинности посылки F с помощью другой посылки $F \to G$ переходят к утверждению об истинности следствия G. Данное правило называют также правилом заключения (от посылки $F \to G$ с помощью посылки F отделяется заключение G).

2. ***Правило отрицания (modus fallens):*** $\dfrac{F \to G, \neg G}{\neg F}$.

От отрицания истинности посылки G с помощью посылки $F \to G$ переходят к отрицанию истинности F.

Таким образом, рассмотренные правила вывода позволяют в истинной импликации $F \to G$ из истинности посылки F делать вывод об истинности следствия G, а из ложности следствия G – о ложности посылки F.

Укажем еще некоторые правила вывода, применяемые в рассуждениях. Путь их получения состоит в том, что сначала заменяем в соответствующей

тавтологии каждую логическую переменную произвольной формулой алгебры высказываний, в результате чего на основании признака логического следования снова получаем тавтологию, а затем от нее переходим к соответствующему правилу вывода (умозаключения), которое и записываем в принятой форме.

3. ***Правило введения конъюнкции:*** $\dfrac{F, G}{F \wedge G}$.

4. ***Правило удаления конъюнкции:*** $\dfrac{F \wedge G}{F}$, $\dfrac{F \wedge G}{G}$.

5. ***Правило введения дизъюнкции:*** $\dfrac{F, G}{F}$, $\dfrac{F, G}{G}$.

6. ***Правило контрапозиции:*** $\dfrac{F \to G}{\neg G \to \neg F}$.

7. ***Правило цепного заключения:*** $\dfrac{F \to G, G \to H}{F \to H}$.

8. ***Правило перестановки посылок:*** $\dfrac{F \to (G \to H)}{G \to (F \to H)}$.

9. ***Правила объединения и разъединения посылок:***

$$\frac{F \to (G \to H)}{(F \wedge G) \to H}, \frac{(F \wedge G) \to H}{F \to (G \to H)}.$$

10. ***Правило расширенной контрапозиции:*** $\dfrac{(F \wedge G) \to H}{(F \wedge \neg H) \to \neg G}$.

На правила 1–10 можно смотреть с двух точек зрения. Во-первых, каждое из них представляет собой утверждение следующего типа: формула, записанная в знаменателе, является логическим следствием всех формул, записанных в числителе данного правила. Во-вторых, каждое из этих правил можно рассматривать как правило получения новой тавтологии из уже имеющихся: если все формулы, записанные в числителе, являются тавтологиями, то тавтологией будет и формула, записанная в знаменателе [7].

Доказывая теоремы в математике, мы всякий раз проводим логическое рассуждение $P \vdash D$ (P – условие теоремы, D – заключение), т. е. выясняем, является ли тавтологией формула $P \to D$. При этом доказательство теоремы может быть *прямым*, когда на основе правил вывода из посылки P мы получаем заключение D. Но доказательство может быть и *косвенным*, когда вместо формулы $P \to D$ мы рассматриваем другую, но равносильную ей формулу.

Теорему вида $P \to D \overset{?}{\equiv} 1$ принято называть *прямой* теоремой.

Наряду с ней можно рассматривать теоремы:

- $D \to P$ – обратную;
- $\neg P \to \neg D$ – противоположную;
- $\neg D \to \neg P$ – обратную противоположной.

Построим таблицу истинности для теорем различных видов.

P	D	$P \to D$	$D \to P$	$\neg P \to \neg D$	$\neg D \to \neg P$
0	0	1	1	1	1
0	1	1	0	0	1
1	0	0	1	1	0
1	1	1	1	1	1

Из таблицы истинности видим, что прямая теорема равносильна обратно противоположной:

$$P \to D \equiv \neg D \to \neg P.$$

Эта равносильность имеет специальное название – *закон контрапозиции*. Заметим, что обратная и противоположная теоремы также связаны законом контрапозиции.

Пример 1.25

Вместо доказательства утверждения «Если $m \cdot n$ нечетное число, то m и n нечетны» ($P \to D$) согласно закону контрапозиции можно доказывать утверждение ($\neg D \to \neg P$): «Если хотя бы одно из чисел m или n четно, то $m \cdot n$ четно».

К методам косвенного доказательства относятся доказательства «от противного». Схемы таких доказательств основаны на законах равносильных преобразований (справедливость которых можно проверить по таблице истинности):

$$A \to B \equiv (A \wedge \neg B) \to \neg A;$$

$$A \to B \equiv (A \wedge \neg B) \to B;$$

$$A \to B \equiv \neg(A \to B) \to C \wedge \neg C.$$

Контрольные вопросы по главе 1

1. Является ли предложение «Африка – остров» элементарным высказыванием алгебры высказываний?
2. У какой бинарной операции самый высокий приоритет?
3. Какие скобки в формуле $F = \big((A \wedge B) \to (\neg A \vee B)\big) \wedge A$ можно убрать так, чтобы значение формулы не изменилось?
4. Является ли последовательность символов $F = C \leftrightarrow (A \neg B)$ формулой логики высказываний?
5. Сколько подформул содержит формула $F = (\neg A_1 \wedge A_2) \to (A_3 \vee \neg A_2)$?
6. Равносильны ли формулы $F_1 = (X \to Y) \vee (X \to Z)$ и $F_2 = X \to (Y \vee Z)$?
7. Является ли форма записи формулы логики высказываний $F = C \vee (A \wedge B)$ дизъюнктивной нормальной формой записи (ДНФ)?
8. К какому классу формул логики высказываний относится формула $F = X \to (Y \wedge X)$?
9. Является ли правильным логическое рассуждение $\dfrac{F \wedge G}{G}$?
10. Какой способ проверки логического следования основан на определении неправильного логического рассуждения?

2 Булевы функции

Джордж Буль (1815–1864) – известный английский математик, чьи работы способствовали созданию современной символической логики. Работы Дж. Буля положили начало алгебре логики, или *булевой алгебре*, он первым показал, что существует аналогия между алгебраическими и логическими действиями, так как и те, и другие предполагают только два варианта ответов – истина или ложь, ноль или единица. Дж. Буль ввел систему обозначений и правил, пользуясь которыми стало возможным закодировать любые высказывания, а следом оперировать ими как обычными числами.

2.1 Введение в булеву алгебру

Под *алгеброй* в современной математике понимают множество объектов произвольной природы с определенными на них *операциями* и свойствами этих операций, даваемых в форме *аксиом*.

Рассмотрим основные требования, предъявляемые к алгебре. Операции алгебры должны быть применимы ко всем объектам множества. В результате выполнения операций должны получаться объекты той же природы, что и исходные. В этом случае говорят, что множество объектов *замкнуто* относительно операций. Операций должно быть конечное число и каждая операция должна быть конечноместной [6].

*Переменная x называется **булевой**, если она способна принимать только два значения: 0 и 1.*

В качестве примера интерпретации такого рода переменных может выступать обычный настенный выключатель света на два положения. Здесь 1 соответствует положению переключателя вверх и 0 – положению вниз.

*Функция $f\left(x_1, x_2, ..., x_n\right)$ называется **булевой**, если все ее аргументы x_i являются булевыми, а сама функция также может принимать только два значения: 0 и 1.*

$f: E_2^n \to E_2$ где $E_2 \overset{def}{=} \{0,1\}$. Множество булевых функций от n аргументов обозначим P_n, $P_n \overset{def}{=} \{ f \mid f: E_2^n \to E_2 \}$.

При работе с булевыми функциями происходит полное абстрагирование от содержательного смысла, который имелся в виду в алгебре высказываний [7]. Тем не менее, между булевыми функциями и формулами алгебры высказываний можно установить взаимно-однозначное соответствие, если:

- установить взаимно-однозначное соответствие между булевыми переменными и логическими переменными;
- установить связь между булевыми функциями и логическими связками;
- оставить расстановку скобок без изменений.

2.2 Способы задания булевых функций

Способы задания булевых функций не отличаются от способов задания обычных функций анализа [6]. К таковым способам задания стандартно относятся:

1) табличный;
2) графический;
3) аналитический.

Рассмотрим табличный способ задания. Пусть $\omega = f(x_1, x_2, ..., x_n)$ – булева функция n аргументов. Область определения данной функции можно рассматривать и как множество упорядоченных наборов (векторов или двоичных наборов) $D = \{(x_1, x_2, ..., x_n) \mid x_i \in \{0,1\}, i = 1,2,...,n\}$, на каждом из которых функция принимает одно из двух значений $\omega \in \{0,1\}$. Количество таких наборов $(x_1, x_2, ..., x_n)$ согласно правилу прямого произведения равно:

$$|D| = |\underbrace{\{0,1\} \times \{0,1\} \times ... \times \{0,1\}}_{n}| = 2 \cdot 2 \cdot ... \cdot 2 = 2^n.$$

Нетрудно определить и количество всех функций $\omega = f(x_1, x_2, ..., x_n)$. Отдельная функция $\omega = f(x_1, x_2, ..., x_n)$ задана, если определены ее значения $(\omega_1, \omega_2, ..., \omega_{2^n})$ на всех наборах $(x_1, x_2, ..., x_n) \in D$, где $\omega_j \in \{0,1\}$ – значение

функции $\omega = f(x_1, x_2, ..., x_n)$ на j-м наборе $(x_1, x_2, ..., x_n) = (0, 1 ... 1) \in D$, $j = 1, 2, ..., 2^n$. Итак, количество булевых функций $\omega = f(x_1, x_2, ..., x_n)$ совпадает с числом двоичных наборов $\left(\omega_1, \omega_2, ..., \omega_{2^n}\right)$, где $\omega_j \in \{0,1\}$. Согласно правилу прямого произведения число последних равно:

$$\underbrace{|\{0,1\} \times \{0,1\} \times ... \times \{0,1\}|}_{2^n} = 2 \cdot 2 \cdot ... \cdot 2 = 2^{2^n}.$$

В качестве примера рассмотрим табличное представление булевой функции трех аргументов $\omega = f(x, y, z)$, где $\omega, x, y, z \in \{0,1\}$. Область определения функции – это множество двоичных наборов $D = \{(x, y, z) \mid x, y, z \in \{0,1\}\}$. Число наборов равно $|D| = 2^3 = 8$, а количество таких функций равно $2^{|D|} = 2^{2^3} = 256$. Значения функции $f(x, y, z)$ удобно представить в виде таблицы (табл. 2.1), где перечислены всевозможные наборы из нулей и единиц длины 3 и для каждого набора указано значение функции $f_i \in \{0,1\}$ на этом наборе.

Таблица 2.1 – Табличное представление булевой функции $f(x, y, z)$

Число	**Двоичная форма**			**Функция**
№	x	y	z	$\omega = f(x, y, z)$
0	0	0	0	$f_0 = 1 = f(0,0,0)$
1	0	0	1	$f_1 = 1 = f(0,0,1)$
2	0	1	0	$f_2 = 0 = f(0,1,0)$
3	0	1	1	$f_3 = 1 = f(0,1,1)$
4	1	0	0	$f_4 = 0 = f(1,0,0)$
5	1	0	1	$f_5 = 0 = f(1,0,1)$
6	1	1	0	$f_6 = 0 = f(1,1,0)$
7	1	1	1	$f_7 = 1 = f(1,1,1)$

Рассмотрим графическое представление булевой функции трех аргументов $\omega = f(x, y, z)$, заданной таблично (табл. 2.1). Множество наборов области определения функции $D = \{(x, y, z) \mid x, y, z \in \{0,1\}\}$ является множеством координат точек вершин единичного трехмерного куба (рис. 2.1, *в*).

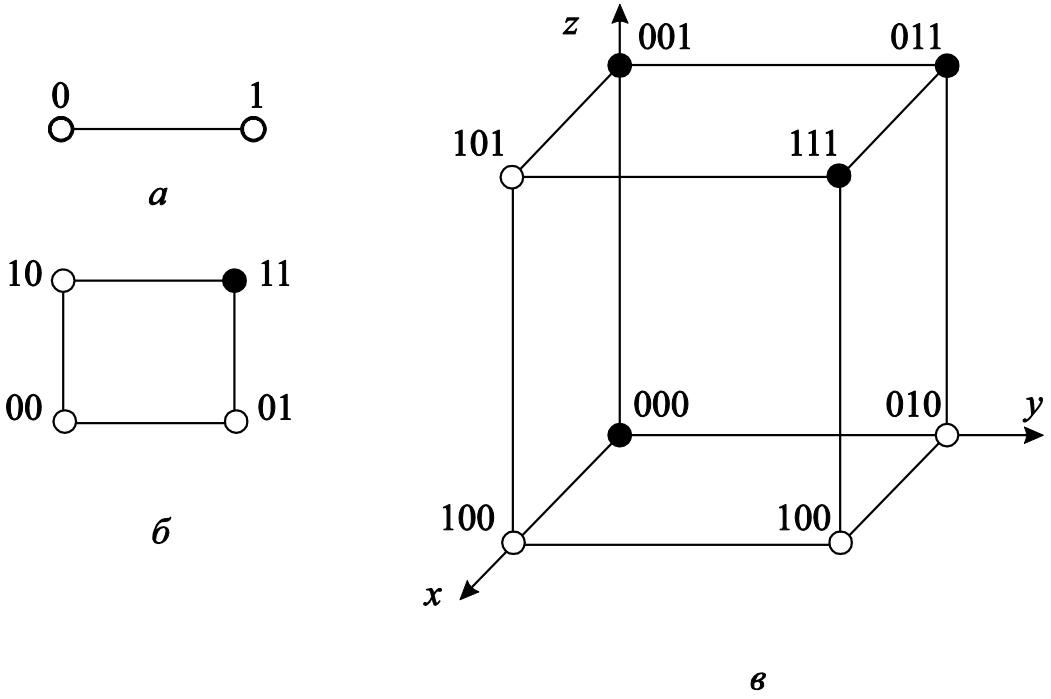


Рис. 2.1 – Графический способ задания булевых функций

На рисунке 2.1 вершины, соответствующие набору булевых переменных, на котором функция принимает значение *1*, обозначаются закрашенными точками. Для функции от одного аргумента $f(x)$ на рисунке 2.1, *а* нет ни одной закрашенной вершины, следовательно, это функция «тождественный нуль» $f(x) = 0$. На рисунке 2.1, *б* представлена булева функция от двух аргументов $f(x, y)$, причем только на одном наборе функция принимает значение «истина» (для этого набора булевых переменных вершина закрашена): $f(1,1) = 1$, следовательно, по определению, это графическое представление булевой функции $f(x, y) = x \wedge y = (0, 0, 0, 1)$.

Соседними *называются два таких двоичных набора одинаковой длины, которые отличаются друг от друга значениями лишь в одном месте.*

Область определения булевой функции n аргументов $\omega = f(x_1, x_2, ..., x_n)$ составляется из наборов координат вершин единичного n-мерного куба.

Рассмотрим аналитический способ задания булевых функций. Приведем таблицы истинности функций одного и двух аргументов. В таблице 2.2 все функции одного аргумента – их число $2^{2^1} = 4$.

Таблица 2.2 – Булевы функции одного аргумента

x	$f(x) = 0$ Тождественный нуль	$f(x) = x$ Повторение x	$f(x) = \bar{x}$ Отрицание x	$f(x) = 1$ Тождественная единица
0	0	0	1	1
1	0	1	0	1

В таблице 2.3 представлены булевы функции от двух аргументов. Для удобства представления БФ таблица «перевернута» – строки таблицы 2.3 – это столбцы, а столбцы – это строки.

В таблице 2.3 функции f_0, f_3, f_5, f_{10}, f_{12}, f_{15} сводятся к булевым функциям одного аргумента (табл. 2.2), а среди остальных принято выделять еще семь функций, которые вместе с отрицанием образуют множество элементарных булевых функций. Это функции:

- f_1 – конъюнкция;
- f_2 – дизъюнкция;
- f_6 – сумма по модулю 2;
- f_8 – стрелка Пирса;
- f_9 – эквивалентность;
- f_{13} – импликация;
- f_{14} – штрих Шеффера.

Таблица 2.3 – Булевы функции двух аргументов

x y	0 0	0 1	1 0	1 1	f
f_0	0	0	0	0	0
f_1	0	0	0	1	xy
f_2	0	0	1	0	$\overline{x \to y}$
f_3	0	0	1	1	x
f_4	0	1	0	0	$\overline{y \to x}$
f_5	0	1	0	1	y
f_6	0	1	1	0	$x \oplus y$
f_7	0	1	1	1	$x \vee y$
f_8	1	0	0	0	$x \downarrow y$
f_9	1	0	0	1	$x \leftrightarrow y$
f_{10}	1	0	1	0	$\overline{y}$
f_{11}	1	0	1	1	$y \to x$
f_{12}	1	1	0	0	$\overline{x}$
f_{13}	1	1	0	1	$x \to y$
f_{14}	1	1	1	0	$x \mid y$
f_{15}	1	1	1	1	1

Функции одного и двух аргументов, представленные в таблицах 2.2–2.3 называются *элементарными.* Символы $\{\neg, \mid, \downarrow, \wedge, \vee, \to, \oplus, \leftrightarrow\}$, участвующие в обозначениях элементарных функций, называются *логическими связками*, или *функциональными символами*.

2.3 Реализация булевых функций формулами

Построение формул выполняется из элементарных булевых функций и носит рекурсивный характер [6]. Пусть X – некоторый фиксированный алфа-

вит переменных, $\sigma = \{\neg, |, \downarrow, \wedge, \vee, \rightarrow, \oplus, \leftrightarrow\}$ – множество функциональных символов (базис) и F – множество булевых функций, соответствующих функциональным символам σ.

Формулой над σ называется всякое выражение вида:

1) x – любая переменная из множества X;

2) $(\neg A), (A|B), (A \downarrow B), (A \wedge B), (A \vee B), (A \rightarrow B), (A \oplus B), (A \leftrightarrow B)$, где A и B – это формулы над σ.

Всякой формуле G однозначно соответствует некоторая функция f_G. Понятие булевой функции f_G, реализуемой формулой G, вводится рекурсивно следующим образом:

1. Формуле $G = x$, где $x \in X$, сопоставляется функция $f_G(x) = x$.
2. Если G равна одной из формул $(\neg A), (A|B), (A \downarrow B), (A \wedge B), (A \vee B), (A \rightarrow B), (A \oplus B), (A \leftrightarrow B)$, где A и B – это формулы над σ, то f_G равно соответствующей элементарной булевой функции $\neg f_A$, $f_A | f_B$, $f_A \downarrow f_B$, $f_A \wedge f_B$, $f_A \vee f_B$, $f_A \rightarrow f_B$, $f_A \oplus f_B$, $f_A \leftrightarrow f_B$. Если $f_G(x_1, x_2, ..., x_n), x_i \in X$, то значение ее на произвольном наборе $(\alpha_1, \alpha_2, ..., \alpha_n), \alpha_i \in \{0,1\}$, совпадает со значением на этом наборе для соответствующей ей элементарной булевой функции.

Таким образом, зная таблицы истинности элементарных функций (функций базиса), можно вычислить и таблицу истинности функции f_G, которую реализует формула G.

Формулы G_1 и G_2 над σ называются *эквивалентными*, если они реализуют равные булевы функции f_{G_1} и f_{G_2}.

В основу рекурсивного правила получения новых формул из элементарных булевых функций положены операции замены переменных и суперпозиции. Рассмотрим их определения и примеры их использования.

Операцией ***замены*** *переменных булевой функции* $f(x_1, x_2, ..., x_n)$ *называется переименование ее переменных или замена их порядка.*

Пусть $\pi = (\pi_1, \pi_2, ..., \pi_n)$ – перестановка, тогда операция замены исходных переменных $(x_1, x_2, ..., x_n)$ на новые $(y_{\pi 1}, y_{\pi 2}, ..., y_{\pi n})$ записывается как:

$$g(y_1, y_2, ..., y_n) = f(x_1, x_2, ..., x_n) = f(y_{\pi 1}, y_{\pi 2}, ..., y_{\pi n}),$$

$$\left| \begin{array}{l} y_{\pi 1} \mapsto x_1 \\ y_{\pi 2} \mapsto x_2 \\ \quad ... \\ y_{\pi n} \mapsto x_n \end{array} \right.$$

где $g(y_1, y_2, ..., y_n)$ – новая булева функция – результат операции замены переменных.

Пример операции замены:

$$(x \leftrightarrow y) \to z = (y \leftrightarrow z) \to \omega,$$

$$\left| \begin{array}{l} y \mapsto x \\ z \mapsto y \\ \omega \mapsto z. \end{array} \right.$$

. . . . . .

Операция ***суперпозиции*** *булевых функций состоит в том, что вместо аргументов данной булевой функции* $f(x_1, x_2, ..., x_n)$ *подставляют некоторые другие булевы функции* $h_1(y_{k11}, y_{k12}, ..., y_{k1m_1})$, $h_2(y_{k21}, y_{k22}, ..., y_{k2m_2})$, ... , $h_r(y_{kr1}, y_{kr2}, ..., y_{krm_r})$.

. . . . . .

Записывают суперпозицию булевых функций следующим образом:

$$f(x_1, x_2, ..., x_n) = h_1(y_{k11}, y_{k12}, ..., y_{k1m_1}), h_2(y_{k21}, y_{k22}, ..., y_{k2m_2}), ..., h_r(y_{kr1}, y_{kr2}, ..., y_{krm_r})$$

$$\left| \begin{array}{l} h_1(y_{k11}, y_{k12}, ..., y_{k1m_1}) \mapsto x_{i_1} \\ h_2(y_{k21}, y_{k22}, ..., y_{k2m_2}) \mapsto x_{i_2} \\ \quad ... \\ h_r(y_{kr1}, y_{kr2}, ..., y_{krm_r}) \mapsto x_{i_n} \end{array} \right.$$

В качестве примера рассмотрим суперпозицию следующих функций:

$$(x \oplus y) = ((x \leftrightarrow z) \oplus (z \to y)) \oplus (x \,|\, (y \wedge x)),$$

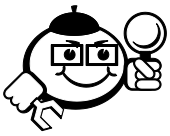
$$\left| \begin{array}{l} (x \leftrightarrow z) \oplus (z \to y) \mapsto x \\ x \,|\, (y \wedge x) \mapsto y. \end{array} \right.$$

Нетрудно заметить, что операция замены переменных является частным случаем операции суперпозиции, если в последней в качестве функций $h_1, h_2, ..., h_r$ положить обычные булевы переменные.

Булева функция $f(x_1, x_2, ..., x_n)$ ***существенно*** *зависит от переменной* x_i, *если существует такой набор значений* $(\alpha_1, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_n)$ *переменных* $(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$, *что:* $f(\alpha_1, ..., \alpha_{i-1}, 0, \alpha_{i+1}, ..., \alpha_n) \neq f(\alpha_1, ..., \alpha_{i-1}, 1, \alpha_{i+1}, ..., \alpha_n)$.

Переменная x_i в этом случае называется *существенной*. В противном случае x_i называется *несущественной*, или *фиктивной*.

Булевы функции f_1 *и* f_2 *называются* ***равными****, если функцию* f_1 *можно получить из* f_2 *способом введения и (или) удаления фиктивных переменных.*



Пример 2.1

Определить существенные и фиктивные переменные булевой функции $f(x, y) = x \vee (x \wedge y)$.

Решение.

Рассмотрим переменную x:

$$\begin{cases} x = 0, 0 \vee (0 \wedge y) = 0; \\ x = 1, 1 \vee (1 \wedge y) = y. \end{cases}$$

$f(0, y) \neq f(1, y)$, следовательно переменная x – существенная.

Рассмотрим переменную y:

$$\begin{cases} y = 0, x \vee (x \wedge 0) = x; \\ y = 1, x \vee (x \wedge 1) = x \vee x = x. \end{cases}$$

Таким образом, $f(x, 0) = f(x, 1)$, следовательно переменная y – фиктивная.

2.4 Минимизация булевых функций

Минимизация булевых функций – представление булевых функций в самом экономичном коротком виде. Известно, что всякой формуле *G* однозначно соответствует некоторая функция f_G. Отсюда следует, что всякая формула алгебры высказываний, равносильная формуле, представляющей некоторую булеву функцию f, будет представлять функцию, равную f.

В частности, одной из таких представляющих формул будет дизъюнктивная нормальная форма (ДНФ) или конъюнктивная нормальная форма (КНФ), а также совершенная дизъюнктивная нормальная форма (СДНФ) или совершенная конъюнктивная нормальная форма (СКНФ). Очевидно, что среди этих форм будут такие, какие содержат меньшее число переменных, чем исходная [16].

Дизъюнктивная нормальная форма называется ***минимальной****, если она содержит наименьшее число вхождений переменных по сравнению со всеми равносильными ей ДНФ.*

Задача минимизации булевой функции состоит в построении такой ДНФ (КНФ) для некоторой заданной функции, которая реализует эту функцию и имеет наименьшее суммарное число операций и символов в формуле, т. е. минимальную сложность. Решение этой задачи важно, когда логические функции реализуются техническими устройствами. Одной из основных целей минимизации является упрощение логических устройств и достижение максимальной экономичности разрабатываемых систем.

Для минимизации булевых функций в целях получения самых простых выражений используется ряд методов, среди которых наибольшее применение находят метод неопределенных коэффициентов, метод карт Карно и метод Квайна – Мак Класки.

Сущность ***метода неопределенных коэффициентов*** заключается в представлении функции в самом общем виде в ДНФ и введении коэффициентов a_i, значения которых (0 или 1) подбираются таким образом, чтобы формула была минимальной. Пусть дана функция трех переменных $f(x_1, x_2, x_3)$, представим формулу этой функции в виде ДНФ общего вида:

$$f(x_1, x_2, ..., x_n) = a_1 x_1 \vee a_2 \overline{x_1} \vee a_3 x_2 \vee a_4 \overline{x_2} \vee a_5 x_3 \vee a_6 \overline{x_3} \vee a_7 x_1 x_2 \vee$$
$$\vee a_8 \overline{x_1} x_2 \vee a_9 x_1 \overline{x_2} \vee a_{10} \overline{x_1}\,\overline{x_2} \vee a_{11} x_1 x_3 \vee a_{12} \overline{x_1} x_3 \vee a_{13} x_1 \overline{x_3} \vee a_{14} \overline{x_1}\,\overline{x_3} \vee$$
$$\vee a_{15} x_2 x_3 \vee a_{16} \overline{x_2} x_3 \vee a_{17} x_2 \overline{x_3} \vee a_{18} \overline{x_2}\,\overline{x_3} \vee a_{19} \overline{x_1} x_2 x_3 \vee a_{20} x_1 \overline{x_2}\,\overline{x_3} \vee$$
$$\vee a_{21} \overline{x_1} x_2 \overline{x_3} \vee a_{22} \overline{x_1}\,\overline{x_2} x_3 \vee a_{23} \overline{x_1} x_2 x_3 \vee a_{24} x_1 \overline{x_2} x_3 \vee a_{25} x_1 x_2 \overline{x_3} \vee a_{26} x_1 x_2 x_3.$$

Подформулы, содержащие k переменных, будем называть подформулами ранга k. Если записать общий вид ДНФ для всех возможных значений аргументов x_1, x_2, x_3, то получим систему 2^n уравнений (т. е. 2^3=8):

0) $f(0,0,0) = a_2 \vee a_4 \vee a_6 \vee a_{10} \vee a_{14} \vee a_{18} \vee a_{19}$;

1) $f(0,0,1) = a_2 \vee a_4 \vee a_5 \vee a_{10} \vee a_{12} \vee a_{16} \vee a_{22}$;

2) $f(0,1,0) = a_2 \vee a_3 \vee a_6 \vee a_8 \vee a_{14} \vee a_{17} \vee a_{21}$;

3) $f(0,1,1) = a_2 \vee a_3 \vee a_5 \vee a_8 \vee a_{12} \vee a_{15} \vee a_{23}$;

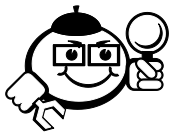
4) $f(1,0,0) = a_1 \vee a_4 \vee a_6 \vee a_9 \vee a_{13} \vee a_{18} \vee a_{20}$;

5) $f(1,0,1) = a_1 \vee a_4 \vee a_5 \vee a_9 \vee a_{11} \vee a_{16} \vee a_{24}$;

6) $f(1,1,0) = a_1 \vee a_3 \vee a_6 \vee a_7 \vee a_{13} \vee a_{17} \vee a_{25}$;

7) $f(1,1,1) = a_1 \vee a_3 \vee a_5 \vee a_7 \vee a_{11} \vee a_{15} \vee a_{26}$.

Для заданной конкретной функции конкретные значения левой части уравнений известны. Если набор x_1, x_2, x_3 такой, что функция на нем принимает значение 0, то все коэффициенты в правой части равны 0. Эти нулевые коэффициенты вычеркиваются и в остальных уравнениях. В каждом оставшемся уравнении приравниваем единице коэффициент, определяющий конъюнкцию наименьшего ранга (ранг – число переменных), а остальные коэффициенты в правой части уравнения приравниваем нулю с учетом ранее сделанных подстановок. После подстановки коэффициентов в ДНФ общего вида получаем результирующее выражение булевой функции.



Пример 2.2

Минимизируем булеву функцию с применением метода неопределенных коэффициентов:

$$f\left(x_1, x_2, x_3\right) = x_1 x_2 x_3 \vee x_1 x_2 \overline{x_3} \vee x_1 \overline{x_2} x_3 \vee x_1 \overline{x_2}\,\overline{x_3} \vee \overline{x_1}\,\overline{x_2}\,\overline{x_3}.$$

Решение.

На первом шаге необходимо вычислить значения функции на всех наборах булевых переменных:

№	x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Подставим полученные логические значения в уравнения:

0) $f(0,0,0) = a_2 \vee a_4 \vee a_6 \vee a_{10} \vee a_{14} \vee a_{18} \vee a_{19} = 1;$

1) $f(0,0,1) = a_2 \vee a_4 \vee a_5 \vee a_{10} \vee a_{12} \vee a_{16} \vee a_{22} = 1;$

2) $f(0,1,0) = a_2 \vee a_3 \vee a_6 \vee a_8 \vee a_{14} \vee a_{17} \vee a_{21} = 1;$

3) ~~$f(0,1,1) = a_2 \vee a_3 \vee a_5 \vee a_8 \vee a_{12} \vee a_{15} \vee a_{23}$~~ $= 0;$

4) $f(1,0,0) = a_1 \vee a_4 \vee a_6 \vee a_9 \vee a_{13} \vee a_{18} \vee a_{20} = 1;$

5) ~~$f(1,0,1) = a_1 \vee a_4 \vee a_5 \vee a_9 \vee a_{11} \vee a_{16} \vee a_{24}$~~ $= 0;$

6) ~~$f(1,1,0) = a_1 \vee a_3 \vee a_6 \vee a_7 \vee a_{13} \vee a_{17} \vee a_{25}$~~ $= 0;$

7) $f(1,1,1) = a_1 \vee a_3 \vee a_5 \vee a_7 \vee a_{11} \vee a_{15} \vee a_{26} = 1.$

Все коэффициенты, входящие в нулевые наборы, должны быть равны нулю, т. е.:

$$a_2 = a_3 = a_4 = a_5 = a_6 = a_8 = a_{10} = a_{12} = a_{14} = a_{15} =$$
$$= a_{16} = a_{17} = a_{21} = a_{22} = a_{23} = 0.$$

Остается рассмотреть единичные наборы:

$$a_1 \vee a_7 \vee a_{11} \vee a_{26} = 1;$$

$$a_1 \vee a_7 \vee a_{13} \vee a_{25} = 1;$$

$$a_1 \vee a_9 \vee a_{11} \vee a_{24} = 1;$$

$$a_1 \vee a_9 \vee a_{13} \vee a_{18} \vee a_{20} = 1;$$

$$a_{18} \vee a_{19} = 1.$$

В первых четырех уравнениях конъюнкция наименьшего ранга – a_1, а в последнем – a_{18}. Таким образом, принимаем $a_1 = 1$, $a_{18} = 1$. Остальные коэффициенты принимаем равными нулю. Подставляя коэффициенты в ДНФ общего вида, получаем минимальную ДНФ заданной функции: $f(x_1, x_2, x_3) = x_1 \vee \overline{x_2}\,\overline{x_3}$.

. . . . . .

Карта Карно – это графическое представление таблицы истинности.

Представляет собой совокупность ячеек, определяемых системой вертикальной и горизонтальных координат.

Каждой ячейке соответствует набор значений входных переменных. Запись в ячейке – это значение функции на соответствующем наборе (рис. 2.2–2.4).

x	y	$f(x,y)$
0	0	$f(0,0)$
0	1	$f(0,1)$
1	0	$f(1,0)$
1	1	$f(1,1)$

а

x \ y	0	1
0	$f(0,0)$	$f(0,1)$
1	$f(1,0)$	$f(1,1)$

б

Рис. 2.2 – Таблица истинности (*а*) и карта Карно (*б*) для булевой функции двух аргументов

x	y	z	$f(x,y,z)$
0	0	0	$f(0,0,0)$
0	0	1	$f(0,0,1)$
0	1	0	$f(0,1,0)$
0	1	1	$f(0,1,1)$
1	0	0	$f(1,0,0)$
1	0	1	$f(1,0,1)$
1	1	0	$f(1,1,0)$
1	1	1	$f(1,1,1)$

а

x \ yz	00	01	11	10
0	$f(0,0,0)$	$f(0,0,1)$	$f(0,1,1)$	$f(0,1,0)$
1	$f(1,0,0)$	$f(1,0,1)$	$f(1,1,1)$	$f(1,1,0)$

б

Рис. 2.3 – Таблица истинности (*а*) и карта Карно (*б*) для булевой функции трех аргументов

xy \ zw	00	01	11	10
00	$f(0,0,0,0)$	$f(0,0,0,1)$	$f(0,0,1,1)$	$f(0,0,1,0)$
01	$f(0,1,0,0)$	$f(0,1,0,1)$	$f(0,1,1,1)$	$f(0,1,1,0)$
11	$f(1,1,0,0)$	$f(1,1,0,1)$	$f(1,1,1,1)$	$f(1,1,1,0)$
10	$f(1,0,0,0)$	$f(1,0,0,1)$	$f(1,0,1,1)$	$f(1,0,1,0)$

Рис. 2.4 – Карта Карно для булевой функции четырех аргументов

Таким образом, имеется взаимно-однозначное соответствие между ячейками карты Карно и строками таблицы истинности. Сущность метода заключается в выделении в карте Карно прямоугольных ячеек (блоков), содержащих одно и то же значение функции. Любой блок может иметь размер $2^a \times 2^b$, где a, b – целые.

Основные правила формирования минимальной ДНФ булевой функции:

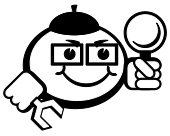
1) каждая ячейка, содержащая единицу, должна войти в какой-то из блоков;
2) результат записывается в виде логической суммы термов, соответствующих сформированным блокам;
3) терм (подформула), описывающий блок, записывается в форме произведения тех переменных, которые на координатной сетке не изменяют свое значение в пределах блока; если координата равна нулю, то соответствующая ей переменная входит в терм с отрицанием, а если координата равна единице, то без отрицания.

Степень сложности булевой функции оценивается числом слагаемых (термов, подформул) и числом букв (булевых переменных, литер). Выражение, которое получено с применением метода карт Карно, на основе термов, сформированных из групп с единичным значением логической функции, при минимальном количестве литер, называется *минимальной* суммой.

При формировании групп для получения минимальных сумм необходимо руководствоваться двумя принципами:

1) группа должна быть как можно больше;
2) число групп должно быть как можно меньше.

Карту Карно следует рассматривать как трехмерную, представляя ее в виде цилиндра (склеивая правую и левую, а также нижнюю и верхнюю границы).



Пример 2.3

Построить минимальную ДНФ булевой функции $f(x, y, z, w) = 1011101110001100$.

Решение.

x	y	z	w	$f(x, y, z, w)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

На рисунке 2.5 показана карта Карно с оптимальным выбором групп ячеек.

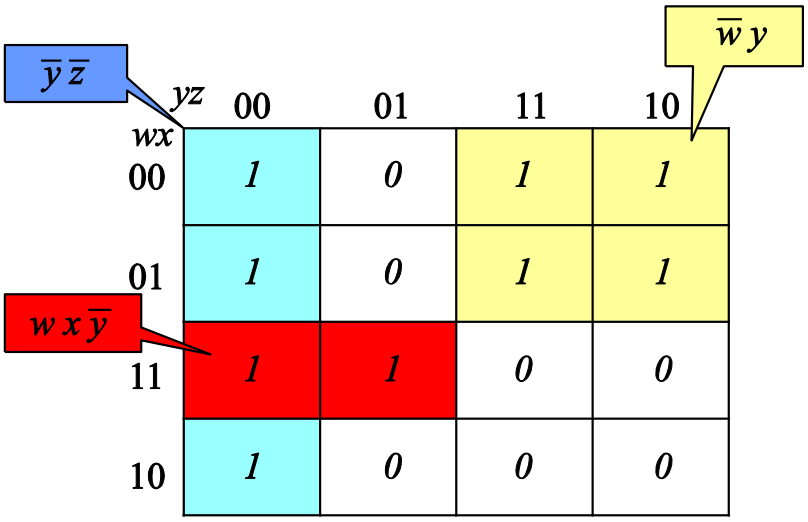


Рис. 2.5 – Карта Карно $f(x, y, z, w)$

В данном примере нельзя ни увеличить размеры групп, ни уменьшить их число. Группа в первом столбце (выделена синим цветом) имеет размер $2^2 \times 2^0 = 4 \times 1$. Квадратная группа справа вверху (светло-коричневого цвета) $2^1 \times 2^1 = 2 \times 2$ и самая маленькая группа из двух ячеек в третьей строке (красного цвета) – $2^1 \times 2^0 = 2 \times 1$. Обратите внимание, что группы могут пересекаться.

Для того чтобы написать выражение по карте Карно, следует воспользоваться «разметкой» координатных осей карты. Нужно для каждой группы отобрать те переменные на координатных осях, значения которых не изменяются в пределах группы. Эти переменные и будут входить в соответствующие термы-произведения. Если переменные равны логическому 0, то они должны входить с отрицанием, если они равны логической 1 – без отрицания.

Для иллюстрации снова обратимся к рисунку 2.5. Квадратная группа расположена в первой и второй строках карты. В этих строках переменная *w* равна логическому 0. Поэтому терм для этой группы должен содержать $\bar{w}$. Поскольку переменная *x* меняет значение при переходе от первой строки ко второй, она не должна входить в терм. Рассмотрев два столбца, в которых лежит группа, мы обнаружим, что переменная *y* имеет постоянное значение, равное логической 1, и, следовательно, должна войти в терм, а *z* изменяет значение и поэтому входить в терм не должна. Итак, установили, что квадратная группа соответствует терму $\bar{w}y$.

Применив аналогичную процедуру к двум другим группам на рисунке 2.5, можно найти соответствующие им термы-произведения. Группа в пер-

вом столбце соответствует терму $\bar{y}\bar{z}$, поскольку и y, и z в этом столбце равны логическому 0, а обе переменные w и x не сохраняют своих значений для этого столбца.

Что же касается самой маленькой группы, то ей соответствует терм $wx\bar{y}$. Таким образом, минимальная сумма для рассматриваемой карты Карно равна: $f(x,y,z,w) = \bar{w}y \vee \bar{y}\bar{z} \vee wx\bar{y}.$

. . . . . .

2.5 Представление булевых функций полиномами Жегалкина

Представление булевых функций в нормальном виде использует три операции: дизъюнкцию, конъюнкцию и отрицание. Булевы функции «дизъюнкция» и «сложение по модулю 2 (неравнозначность)», а также «константа единица», позволяют представлять булевы функции в виде полиномов, по своим алгебраическим свойствам аналогичных обычным алгебраическим полиномам. Такие полиномы называют полиномами Жегалкина (в честь русского математика конца XIX в. И. И. Жегалкина).

Будем говорить, что БФ $f\left(x_1,...,x_n\right)$ представлена в виде полинома Жегалкина, если:

$$f\left(x_1,...,x_n\right) = a_0 \oplus a_1x_1 ... a_nx_n \oplus a_{12}x_1x_2 \oplus ...$$
$$... \oplus a_{n-1,n}x_{n-1}x_n \oplus a_{123}x_1x_2x_3 \oplus ... \oplus a_{1...n}x_1x_2...x_n \ ,$$

причем коэффициенты полинома $a_0,...,a_{1...n} \in \{0,1\}$.

Операция сложения по модулю 2 обладает свойствами:

$x \oplus y = y \oplus x$;

$x \oplus (y \oplus z) = (x \oplus y) \oplus z$;

$x(y \oplus z) = xy \oplus xz$;

$x \oplus 0 = x$;

$x \oplus x = 0$;

$a \oplus b$.

Учитывая эти свойства, а также свойства конъюнкции, будем преобразовывать формулы, содержащие только $\oplus$ и $\wedge$, по обычным алгебраическим за-

конам: переставлять множители и слагаемые, раскрывать скобки, выносить общие множители. Особенность преобразования только одна: если в сумме встречаются два одинаковых слагаемых, их можно опустить.

Теорема 2.1. Любую булеву функцию можно представить полиномом Жегалкина, причем однозначно.

Доказательство теоремы дает способ построения полинома Жегалкина для произвольной БФ, поэтому будем сопровождать наши рассуждения примером.

Шаг 1. Пусть $f(x_1, ..., x_n)$ – произвольная булева функция. Причем, если $f(x_1, ..., x_n) = 0 = a_0$, то полином Жегалкина для нее уже построен (полином нулевой степени). В противном случае ($f \neq 0$) данную булеву функцию можно представить однозначно в СДНФ.

В качестве примера рассмотрим функцию, заданную таблицей истинности:

№	x_1	x_2	x_3	f	ЭК
0	0	0	0	0	
1	0	0	1	1	$\overline{x_1}\,\overline{x_2}x_3$
2	0	1	0	0	
3	0	1	1	1	$\overline{x_1}x_2x_3$
4	1	0	0	0	
5	1	0	1	1	$\overline{x_1}x_2\overline{x_3}$
6	1	1	0	0	
7	1	1	1	0	

СДНФ для данной функции имеет вид:

$$f(x_1, x_2, x_3) = \overline{x_1}\,\overline{x_2}x_3 \vee \overline{x_1}x_2x_3 \vee x_1\overline{x_2}x_3.$$

Шаг 2. Заменим в СДНФ данной булевой функции знак дизъюнкции знаком сложения по модулю 2. Рассмотрим табличное представление этих функций:

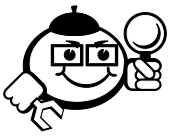
a	b	$a \vee b$	$a \oplus b$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

Значения $a \vee b$ и $a \oplus b$ отличаются только в одном случае: когда аргументы a и b одновременно принимают значение единица. Но в СДНФ знак дизъюнкции соединяет полные ЭК, каждая из которых принимает значение единица только на «своем» наборе переменных (т. е. одновременно разные ПЭК не могут принимать значение единица), поэтому в СДНФ можно знак $\vee$ заменить на $\oplus$:

$$f(x_1, x_2, x_3) = (x_1 \oplus 1)(x_2 \oplus 1)x_3 \oplus (x_1 \oplus 1)x_2x_3 \oplus x_1(x_2 \oplus 1)x_3.$$

Затем преобразуем полученную формулу, раскрывая скобки и приводя подобные:

$$f(x_1, x_2, x_3) = x_1x_2x_3 \oplus x_2x_3 \oplus x_1x_3 \oplus x_3 \oplus x_1x_2x_3 \oplus x_2x_3 \oplus x_1x_2x_3 \oplus x_1x_3 =$$
$$= x_1x_2x_3 \oplus x_3.$$



Пример 2.4

Представить полиномом Жегалкина формулу $F = xy \vee \bar{x}\bar{y} \vee \bar{y}z$.

Решение.

Будем использовать свойства операции сложения по модулю два:

$$F = xy \vee \bar{x}\bar{y} \vee \bar{y}z \equiv \overline{\overline{xy} \,\&\, \overline{\bar{x}\bar{y}} \,\&\, \overline{\bar{y}z}} \equiv$$
$$\equiv (xy \oplus 1)((x \oplus 1)(y \oplus 1) \oplus 1)((y \oplus 1)z \oplus 1) \oplus 1 \equiv$$
$$\equiv (xy \oplus 1)(xy \oplus x \oplus y)(yz \oplus z \oplus 1) \oplus 1 \equiv (x \oplus y)(yz \oplus z \oplus 1) \oplus 1 \equiv$$
$$\equiv xyz \oplus yz \oplus xz \oplus yz \oplus x \oplus y \oplus 1 \equiv xyz \oplus xz \oplus x \oplus y \oplus 1.$$

Получили представление БФ полиномом Жегалкина.

2.6 Функциональная полнота системы булевых функций

Система булевых функций $F = \{f_1, f_2, ..., f_m\}$ *из* P_2 *называется (функционально)* ***полной****, если любая булева функция может быть реализована формулой над этой системой.*

Известно, что в основу рекурсивного правила получения новых формул из элементарных булевых функций положены операции замены переменных и суперпозиции. Тогда определение можно записать так: система булевых функций $F = \{f_1, f_2, ..., f_m\}$ из P_2 называются (функционально) *полной*, если любая булева функция может быть выражена через $f_1, f_2, ..., f_m$ с помощью операций замены переменных и суперпозиции.

Класс (множество) булевых функций называется (функционально) ***замкнутым****, если вместе с функциями из этого класса он содержит и все функции, полученные из них с помощью указанных операций.*

Очевидно, что для доказательства замкнутости класса достаточно проверить его замкнутость лишь относительно элементарных операций суперпозиции.

Рассмотрим некоторые важнейшие замкнутые классы функций из P_2.

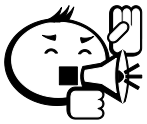
1. *Класс функций, сохраняющих константу 0.*

Функция $f(x_1, ..., x_n)$ *сохраняет константу 0, если* $f(0, ..., 0) = 0$.

Класс функций, сохраняющих константу 0, обозначают через T_0.

Теорема 2.2. Класс функций T_0 замкнут.

Пусть $f_1(x_1, ..., x_n), f_2(x_1, ..., x_n)$ – функции, сохраняющие 0. Покажем, что и суперпозиция их $f_3(x_1, ..., x_n) = f_1(x_1, f_2(x_1, ..., x_n), ..., x_m)$ сохраняет 0. Действительно, $f_3(0, ..., 0) = f_1(0, f_2(0, ..., 0), ..., 0) = f_1(0, ..., 0) = 0$.



Число функций n переменных, сохраняющих 0, равно половине общего числа функций: $|T_0| = 2^{2^n} / 2 = 2^{2^n - 1}$.

2. *Класс функций, сохраняющих константу 1.*

Функция $f(x_1, ..., x_n)$ *сохраняет константу 1, если* $f(1, ..., 1) = 1$.

Класс функций, сохраняющих константу 1, обозначают через T_1.

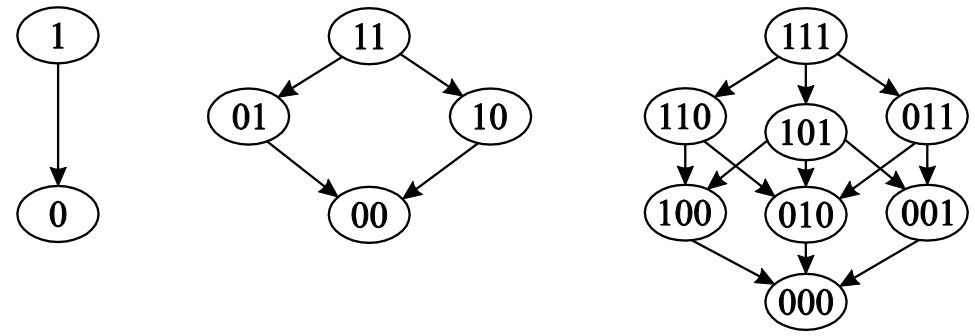
Теорема 2.3. Класс функций T_1 замкнут.

Пусть $f_1(x_1, ..., x_n), f_2(x_1, ..., x_n)$ – функции, сохраняющие 1. Покажем, что и суперпозиция их $f_3(x_1, ..., x_n) = f_1(x_1, f_2(x_1, ..., x_n), ..., x_m)$ сохраняет 1. Действительно, $f_3(1, ..., 1) = f_1(1, f_2(1, ..., 1), ..., 1) = f_1(1, ..., 1) = 1$.

Число функций n переменных, сохраняющих 1, равно половине общего числа функций: $|T_1| = 2^{2^n} / 2 = 2^{2^n - 1}$.

3. *Класс монотонных функций.*

Для определения понятия монотонной функции введем частичный порядок на множестве двоичных слов длины n: будем говорить, что слово $A=(a_1,...,a_n)$ предшествует слову $B=(b_1,...,b_n)$ (обозначим $A\le B$), если $a_i\le b_i$ для всех $i=\overline{1,n}$ $(a_i,b_i\in\{0,1\})$.



а *б* *в*

а) $n=1$, *б*) $n=2$, *в*) $n=3$

Рис. 2.6 – Отношение предшествования

Например, рассмотрим набор слов длиной $n=2$ $A=(00)$, $B=(01)$, $C=(10)$, $D=(11)$. Отношение предшествования связывает тройки слов: $A\le B\le D$, $A\le C\le D$, но B и C несравнимы.

Отношение предшествования можно представить графически (рис. 2.6). Здесь дуги соединяют только «соседние» двоичные слова длины n, т. е. двоичные слова, отличающиеся лишь одним двоичным символом, а предшествующими данному двоичному слову будут все слова, соответствующие вершинам, достижимым из данной.

*Функция $f(x_1,...,x_n)$ называется **монотонной**, если для любых наборов значений $A=(a_1,...,a_n)$ и $B=(b_1,...,b_n)$ списка переменных функций, таких, что $A>B$, выполняется $f(A)\ge f(B)$.*

Класс монотонных функций обозначают через M.

Теорема 2.4. Класс монотонных функций M замкнут.

Пусть $f_1(x_1,...,x_n), f_2(x_1,...,x_n)$ – монотонные функции. Покажем, что и суперпозиция их $f_3(x_1,...,x_n)=f_1(x_1,f_2(x_1,...,x_n),...,x_m)$ есть монотонная функция. Пусть $A>B$ – произвольные наборы. Обозначим наборы, соответствующие суперпозиции функций, через $\tilde{A}=(a_1,f_2(a_1,a_2,...,a_n),...,a_n)$ и $\tilde{B}=(b_1,f_2(b_1,b_2,...,b_n),...,b_n)$. Так как сравнение наборов $A\succ B$ выполняется по координатам и $f_2(A)\geq f_2(B)$, то $\tilde{A}>\tilde{B}$ и, значит, $f_1(\tilde{A})\geq f_1(\tilde{B})$. Монотонность f_3 следует из соотношения $f_3(A)=f_1(\tilde{A})\geq f_1(\tilde{B})=f_3(B)$.

Примером монотонной функции является функция $f_1(x_1,x_2)=x_1\,\&\,x_2$, т. к. $f_1(0,0)=0\leq f_1(0,1)=0\leq f_1(1,1)=1$ и $f_1(0,0)=0\leq f_1(1,0)=0\leq f_1(1,1)=1$.

4. *Класс линейных функций.*

*Функция $f(x_1,...,x_n)$ называется **линейной**, если степень ее полинома Жегалкина не выше первой.*

Класс линейных функций обозначают через L.

Теорема 2.5. Класс линейных функций L замкнут.

Пусть $f_1(x_1,...,x_n), f_2(x_1,...,x_n)$ – линейные функции. Покажем, что и суперпозиция их $f_3(x_1,...,x_n)=f_1(x_1,f_2(x_1,...,x_n),...,x_m)$ есть линейная функция. Имеем $f_1=a_0\oplus a_{1.}x_1\oplus a_{2.}x_2\oplus...\oplus a_{n.}x_n$ и $f_2=b_0\oplus b_{1.}x_1\oplus b_{2.}x_2\oplus...\oplus b_{n.}x_n$. Очевидно, что $f_3=a_0\oplus a_{1.}x_1\oplus \oplus(b_0\oplus b_{1.}x_1\oplus b_{2.}x_2\oplus...\oplus b_{n.}x_n)x_2\oplus...\oplus a_nx_n$ есть линейная функция. Степень суперпозиции линейных функций может только уменьшиться.

Например, $f_1(x_1, x_2) = x_1 \& x_2 \notin L$, т. к. ее полином Жегалкина второй степени $f_1(x_1, x_2) = x_1 x_2 \oplus 0x_1 \oplus 0x_2 \oplus 0$; а $f_2(x) = \bar{x} \in L$, т. к. $f_2(x) = x \oplus 1$ – полином первой степени.

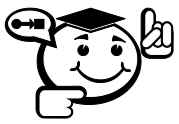
5. *Класс самодвойственных функций.*

Функция $f(x_1, ..., x_n)$ *называется* ***самодвойственной****, если для любого набора значений* $A = (a_1, ..., a_n)$ *списка переменных выполняется* $f(A) = \overline{f}(\overline{A})$.

Это означает, что самодвойственная функция на противоположных наборах принимает противоположные значения. Преобразование $\overline{f}(\overline{x_1}, ..., \overline{x_n})$ обозначают как $f^*(x_1, ..., x_n)$. Тогда f – самодвойственная, если $f(x_1, ..., x_n) = f^*(x_1, ..., x_n)$.

Класс самодвойственных функций обозначают через S.

Теорема 2.6. Класс самодвойственных функций S замкнут.



Пусть $f_1(x_1, ..., x_n), f_2(x_1, ..., x_n)$ – линейные функции. Покажем, что и суперпозиция их $f_3(x_1, ..., x_n) = f_1(x_1, f_2(x_1, ..., x_n), ..., x_m)$ есть линейная функция. Имеем $f_1 = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus ... \oplus a_n x_n$ и $f_2 = b_0 \oplus b_1 x_1 \oplus b_2 x_2 \oplus ... \oplus b_n x_n$. Очевидно, что $f_3 = a_0 \oplus a_1 x_1 \oplus \oplus (b_0 \oplus b_1 x_1 \oplus b_2 x_2 \oplus ... \oplus b_n x_n) x_2 \oplus ... \oplus a_n x_n$ есть линейная функция. Степень суперпозиции линейных функций может только уменьшиться.

Например, $f_1(x_1, x_2) = x_1 \& x_2 \notin L$, т. к. ее полином Жегалкина второй степени $f_1(x_1, x_2) = x_1 x_2 \oplus 0x_1 \oplus 0x_2 \oplus 0$; а $f_2(x) = \bar{x} \in L$, т. к. $f_2(x) = x \oplus 1$ – полином первой степени.

Перечисленные классы функций обладают важным свойством: суперпозиция функций, принадлежащих одному классу, дает функцию из этого же класса, т. е. справедлива следующая теорема.

Теорема 2.7. Класс $\left\{ \begin{matrix} T_0 \\ T_1 \\ S \\ L \\ M \end{matrix} \right\}$ замкнут относительно суперпозиции булевых функций.

Чтобы доказать теорему, достаточно показать, что каждый из этих классов замкнут относительно операции подстановки переменных и подстановки в функцию. Эти доказательства очевидны и требуют только знания определений пяти классов булевых функций.

Теорема 2.8 (*теорема Поста о функциональной полноте*). Для того чтобы система булевых функций $\{ f_1, ..., f_r \}$ была полной, необходимо и достаточно, чтобы она содержала:

1) функцию, не сохраняющую нуль;
2) функцию, не сохраняющую единицу;
3) несамодвойственную функцию;
4) немонотонную функцию;
5) нелинейную функцию.

Для доказательства теоремы нам понадобятся две леммы.

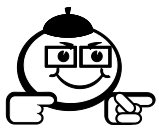
Лемма 2.1 (*о немонотонной функции*). Из немонотонной функции $f(x_1, ..., x_n)$ путем подстановки констант на место $n-1$ аргумента можно получить отрицание.

Пусть булева функция $f(x_1, ..., x_n) \notin M$. Тогда найдется пара двоичных слов $A = (a_1, ..., a_n)$ и $B = (b_1, ..., b_n)$, таких, что $A \le B$,

но $f(a_1, a_2, ..., a_n) > f(b_1, b_2, ..., b_n)$, т. е. $f(A)=1, f(B)=0$. Пару слов A и B всегда можно связать цепочкой соседних слов, предшествующих одно другому (вершина A достижима из вершины B), например: $A=(001001)\leqq(011001)\leq(011011)=B$. Так как в начале цепочки $f(A)=1$, а в конце $f(B)=0$, то найдется пара соседних слов, такая, что $f(a_1, ..., a_{k-1}, 0, a_{k+1}, ..., a_n)=1$, но $f(a_1, ..., a_{k-1}, 1, a_{k+1}, ..., a_n)=0$. Рассмотрим функцию одного аргумента $\eta(x)=f(a_1, ..., a_{k-1}, x, a_{k+1}, ..., a_n)$. По построению $\eta(x)=\bar{x}$.

. . . . . .

. . . . . .



Лемма 2.2 (*о нелинейной функции*). Пусть $f(x_1, ..., x_n)$ – нелинейная функция и $n>2$. Подстановкой констант на места $n-2$ аргументов можно получить нелинейную функцию двух аргументов.

. . . . . .

. . . . . .

Так как $f(x_1, ..., x_n)\notin L$, то полином Жегалкина этой булевой функции содержит хотя бы один член степени выше первой (пусть для определенности он содержит переменные x_1 и x_2). Сгруппируем слагаемые так, чтобы выделить переменные x_1 и x_2:

$$f(x_1, ..., x_n)=x_1x_2\gamma_0(x_3, ..., x_n)\oplus x_1\gamma_1(x_3, ..., x_n)\oplus$$
$$\oplus x_2\gamma_2(x_3, ..., x_n)\oplus\gamma_3(x_3, ..., x_n).$$

Подберем значения $x_1=a_1, ..., x_n=a_n$ так, чтобы $\gamma_0(a_3, ..., x_n)=1$ (это возможно, т. к. $\gamma_0\neq 0$). Получим нелинейную функцию двух переменных:

$$\mu(x_1, x_2)=x_1x_2\oplus\alpha x_1\oplus\beta x_2\oplus\gamma,$$

где $\alpha, \beta, \gamma\in\{0,1\}$.

. . . . . .

Доказательство теоремы Поста.

Необходимость следует из замкнутости классов T_0, T_1, S, L, M. Действительно, предположив противное (т. е. система $\{f_1, ..., f_r\}$, но все функции f_i, $i=\overline{1,r}$ принадлежат одному из пяти классов), по *теореме 2.7* получим, что и

любая суперпозиция этих функций принадлежит этому классу. То есть не всякая булева функция может быть представлена суперпозицией функций данной системы $\{f_1, ..., f_r\}$ – противоречие.

Достаточность. Доказательство проведем по следующей схеме:

1) используя функции системы, не принадлежащие T_0, T_1, S, M, построим константы 0,1 и отрицание $\bar{x}$;
2) используя 0,1 и $\bar{x}$, а также функцию системы, не принадлежащую классу L, построим конъюнкцию;
3) в силу полноты системы $\{\wedge, \neg\}$ любая булева функция может быть представлена в виде суперпозиций $\wedge, \neg$, а следовательно и функций системы $\{f_1, ..., f_r\}$.

Шаг 1. Построим константы и отрицание. Найдем в данной системе функции f и g такие, что $f \notin T_0$, а $g \notin T_1$, и построим $\varphi(x) = f(x, x, ..., x)$ и $\phi(x) = g(x, x, ..., x)$. Тогда $\varphi(0) = 1$, $\phi(1) = 0$ и возможны следующие случаи (табл. 2.4).

Таблица 2.4 – Построение констант и отрицания

x	а		б		в		г	
	$\varphi(x)$	$\phi(x)$	$\varphi(x)$	$\phi(x)$	$\varphi(x)$	$\phi(x)$	$\varphi(x)$	$\phi(x)$
0	1	0	1	1	1	1	1	0
1	0	0	1	0	0	0	1	0

В случае а: $\varphi(x) \equiv \bar{x}$, $\phi(x) \equiv 0$, $\varphi(\phi(x)) \equiv 1$.

В случае б: $\varphi(x) \equiv 1$, $\phi(x) \equiv \bar{x}$, $\varphi(\phi(x)) \equiv 0$.

В случае в: $\varphi(x) \equiv \phi(x) \equiv \bar{x}$. Для построения констант найдем в системе несамодвойственную функцию $h(x_1, ..., x_n)$. Так как $h \notin S$, то найдется набор значений аргументов $(a_1, a_2, ..., a_n)$, такой, что $h(a_1, a_2, ..., a_n) = h(\bar{a}_1, \bar{a}_2, ..., \bar{a}_n)$. Для этого набора рассмотрим функцию $\xi(x) = h(x^{\alpha_1}, x^{\alpha_2}, ..., x^{\alpha_n})$. Для нее

$$\xi(1) = h(1^{\alpha_1}, 1^{\alpha_2}, ..., 1^{\alpha_n}) = h(\alpha_1, \alpha_2, ..., \alpha_n) =$$
$$= h(\bar{\alpha}_1, \bar{\alpha}_2, ..., \bar{\alpha}_n) = h(0^{\alpha_1}, 0^{\alpha_2}, ..., 0^{\alpha_n}) = \xi(0),$$

т. е. $\xi(1)=\xi(0)$ и функция $\xi(x)$ является константой.

В случае г: $\varphi(x)\equiv 1,\ \phi(x)\equiv 0$. Для построения отрицания найдем в системе немонотонную функцию. По лемме 2.1 подстановкой констант 0 и 1 получим отрицание.

Таким образом, суперпозицией функций системы $\{f_1, ..., f_r\}$ получены константы 0, 1 и отрицание $\bar{x}$.

Шаг 2. Построим теперь БФ конъюнкцию. Найдем в системе нелинейную функцию. По лемме 2.2 можно построить нелинейную функцию вида $\mu(x_1, x_2)=x_1x_2\oplus\alpha x_1\oplus\beta x_2\oplus\gamma$. Преобразуем эту функцию следующим образом:

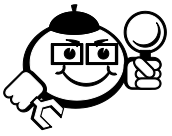
$$\mu(x_1, x_2)=\mu(x_1, x_2)\oplus\alpha\beta\oplus\alpha\beta=x_1(x_2\oplus\alpha)\oplus\beta(x_2\oplus\alpha)\oplus\alpha\beta\oplus\gamma=$$
$$=(x_1\oplus\beta)(x_2\oplus\alpha)\oplus\alpha\beta\oplus\gamma\ .$$

Следовательно, чтобы получить конъюнкцию, необходимо выполнить действия: $x_1x_2=\mu(x_1\oplus\beta, x_2\oplus\alpha)\oplus\alpha\beta\oplus\gamma$.

Учитывая, что $x\oplus\gamma=\begin{cases}\bar{x}, \text{ при } \gamma=0;\\ x, \text{ при } \gamma=1;\end{cases}$

и функция $\bar{x}$, а также константы 0 и 1 уже построены, построение конъюнкции суперпозицией функций системы $\{f_1, ..., f_r\}$ закончено.

Шаг 3. Поскольку система $\{\wedge, \neg\}$ полная и получена суперпозицией функций системы $\{f_1, ..., f_r\}$, заключаем, что система $\{f_1, ..., f_r\}$ также полная, т. е. любую булеву функцию можно представить суперпозицией функций $f_1, ..., f_r$. Теорема Поста о функциональной полноте доказана.



Пример 2.5

Определить принадлежность функций системы пяти замкнутым классам. Проверить выполнение условий теоремы Поста для системы булевых функций:

$$\begin{cases} f_1=x\to y\\ f_2=0\end{cases}.$$

Решение.

Определим, принадлежат ли функции системы к классу T_0. Функция f_1 зависит от двух переменных, x и y. Найдем значение функции $f_1(0,0)=0\to 0=1$, следовательно, $f_1 \notin T_0$. Функция $f_2 \in T_0$.

Определим принадлежность функций системы классу T_1. Для этого найдем значения функций, при значениях входных переменных, равных 1.

$$f_1(1,1)=1\to 1=1 \Rightarrow f_1 \in T_1; \qquad f_2=0 \Rightarrow f_2 \notin T_1.$$

Определим принадлежность функций системы классу S. Очевидно, что $f_1 \notin S,\ f_2 \notin S$.

Определим принадлежность функций системы классу L. Для этого построим полиномы Жегалкина для каждой функции системы.

$f_1 = xy \oplus x \oplus 1$ – степень полинома равна двум, следовательно, $f_1 \notin L$.

$f_2 = 0$ – степень полинома равна единице, следовательно, $f_2 \in L$.

Определим принадлежность функций системы классу M.

Функция f_1 зависит от двух переменных ($n=2$) и при $(10) \geqq (00)$ условие $f_1(1,0) > f_1(0,0)$ не выполняется, следовательно, $f_1 \notin M$.

Занесем все данные в таблицу Поста, знак «+» будет обозначать, что функция принадлежит к классу, знак «–» – функция не принадлежит классу.

f	T_0	T_1	S	L	M
f_1	–	+	–	–	–
f_2	+	–	–	+	+

Система БФ будет полной, если в каждом столбце таблицы Поста встретится хотя бы один знак «–». Таким образом, система булевых функций $\{\to, 0\}$ по теореме Поста является полной системой.

. . . . . .

2.7 Практическое применение булевых функций

Булевы функции широко применяются при описании работы дискретных управляющих систем (контактных схем, схем из функциональных элементов, логических сетей и т. д.), при исследовании некоторых электрических цепей, так называемых релейно-контактных схем.

Под *релейно-контактной схемой* понимается устройство из проводников и двухпозиционных контактов. Оно может быть предназначено, например, для соединения (или разъединения) полюсов источника тока с некоторым потребителем. Контакты релейно-контактной схемы могут быть двух типов: *замыкающие* и *размыкающие*. Каждый контакт подключен к некоторому реле (переключателю). К одному реле может быть подключено несколько контактов – как замыкающих, так и размыкающих. Технически реле представляет собой катушку с металлическим сердечником (магнитопроводом), вблизи которого находится соответствующий контакт.

Когда через катушку пропускается электрический ток, металлический сердечник намагничивается и замыкает все находящиеся при нем замыкающие контакты. Одновременно все размыкающие контакты, относящиеся к данному реле, размыкаются. Поскольку замыкающие контакты при отсутствии в реле электрического тока разомкнуты, они называются также *нормально разомкнутыми*. Аналогично, размыкающие контакты называются *нормально замкнутыми*. При обесточивании обмоток реле (т. е. когда реле отключается) все замыкающие контакты снова размыкаются, а все размыкающие – замыкаются.

Рассмотрим поведение контакта на примере работы физического устройства – реле и обычной кнопки дверного звонка. Схема реле представлена на рисунке 2.7.

Суть работы реле – это его способность находиться в одном из двух состояний. Одно состояние связывается с наличием напряжения на обмотке сердечника, а другое – с его отсутствием. Упругая пластина реле имеет свойство отклоняться вверх при отсутствии напряжения на обмотке сердечника. При наличии на обмотке напряжения сердечник намагничивается и притягивает упругую пластину, устанавливая ее в горизонтальное положение. Именно это свойство реле изменять положение упругой пластины используется в электрических цепях в качестве переключателя.

Различают два способа включения реле в электрические цепи. При первом способе реле используется как нормально разомкнутый контакт (рис. 2.8), а при втором – как нормально замкнутый контакт (рис. 2.9).

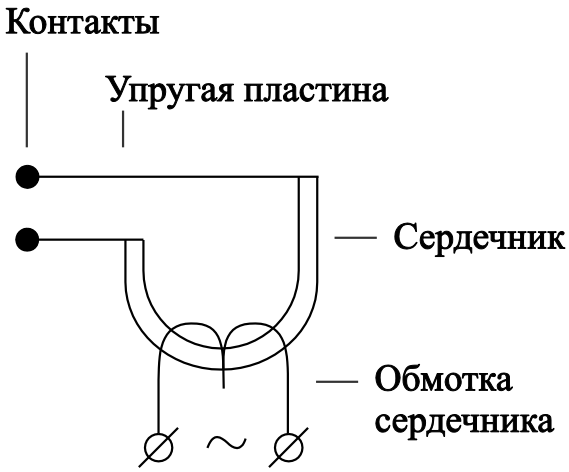


Рис. 2.7 – Реле

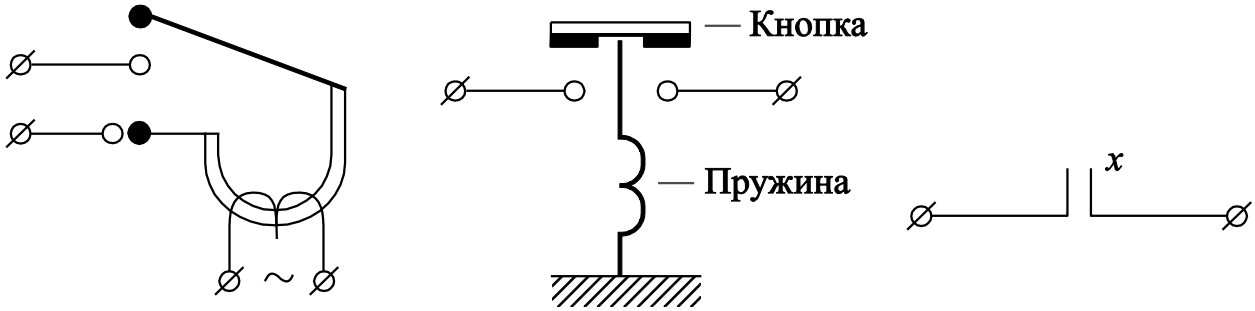


Рис. 2.8 – Нормально разомкнутый контакт

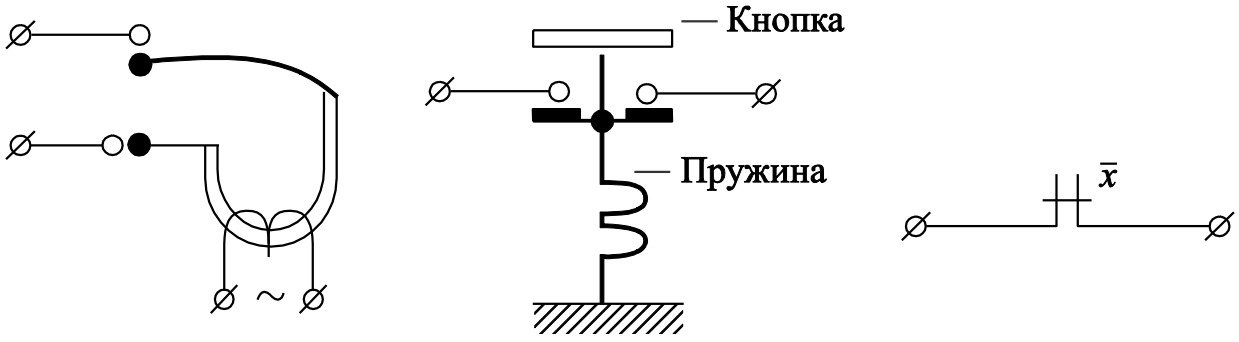


Рис. 2.9 – Нормально замкнутый контакт

Таким образом, объектами релейно-контактных схем являются контакты и контактные схемы. Различают два вида контактов – нормально разомкнутые и нормально замкнутые. Контактные схемы также имеют два состояния – проводящее и непроводящее [6].

Каждому реле ставится в соответствие своя булева переменная x_1 или $x_2 \ldots$, или x_n, которая принимает значение 1, когда реле срабатывает, и значение 0 при отключении реле.

Всей релейно-контактной схеме ставится в соответствие булева переменная y, зависящая от булевых переменных $x_1, ..., x_n$, сопоставленным тем реле, которые участвуют в схеме. Если при данном наборе состояний реле $x_1, ..., x_n$ (некоторые из этих реле находятся в рабочем состоянии под током, остальные отключены, т. е. «обесточены») вся релейно-контактная схема проводит электрический ток, то переменной y ставится в соответствие значение 1. Если же при этом наборе состояний реле $x_1, ..., x_n$ схема не проводит электрический ток, то считаем, что переменная y принимает значение 0.

Поскольку каждый набор состояний реле $x_1, ..., x_n$ характеризуется набором, составленным из нулей и единиц и имеющим длину n, то данная релейно-контактная схема определяет некоторое правило, по которому каждому такому набору длины n, составленному из нулей и единиц, сопоставляется либо 0, либо 1. Таким образом, каждая релейно-контактная схема, в которой занято n независимых реле (контактов в ней может быть n или больше), определяет некоторую булеву функцию y от n аргументов. Она принимает значение 1 на тех и только тех наборах значений аргументов $x_1, ..., x_n$, которые соответствуют тем состояниям реле $x_1, ..., x_n$, при которых данная схема проводит электрический ток. Такая булева функция $y = f(x_1, ..., x_n)$ называется *функцией проводимости* данной релейно-контактной схемы.

Таким образом, теория булевых функций предоставляет математические модели реальных физических релейно-контактных схем [7].

Булевы функции можно рассматривать как математические модели устройств цифровой техники. При этом физические принципы работы устройств могут быть разными, общим же является то, что входы и выходы такого устройства могут находиться в одном из двух физических состояний [14]. На рисунке 2.10 приведена одна из возможных схем устройств, реализующих булеву функцию: на каждом из входов и на выходе возможно высокое или низкое напряжение.

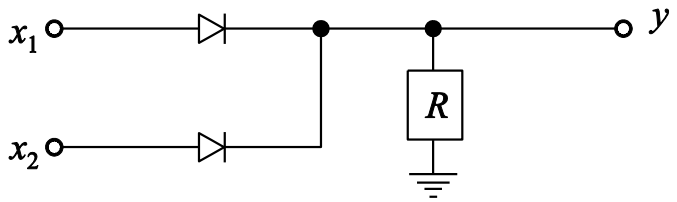


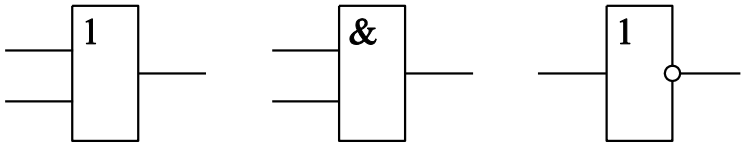
Рис. 2.10 – Функциональный элемент «дизъюнкция»

При появлении хотя бы на одном из входов высокого уровня напряжения через соответствующий диод и сопротивление R протекает ток, и в точке y регистрируется высокий уровень напряжения. Если на обоих входах низкий уровень напряжения, то на выходе – тоже низкий. Таким образом, схему рисунка 2.10 можно представить булевой функцией:

$$y = f\left(x_1, x_2\right) = x_1 \vee x_2 .$$

Аналогичную схему можно построить из электронных ламп, электромеханических переключателей и пр. Символически такую схему (логический элемент «или») будем обозначать, как на рисунке 2.11, *а*.

Логический элемент «и» соответствует булевой функции конъюнкции $y = f\left(x_1, x_2\right) = x_1 \,\&\, x_2$, логический элемент «не» реализует булеву функцию отрицание $y = f\left(x\right) = \bar{x}$.



а *б* *в*

Рис. 2.11 – Логические элементы: *а*) логический элемент «или», *б*) логический элемент «и», *в*) логический элемент «не»

Эти логические (функциональные) элементы являются наиболее распространенными: в силу полноты системы $\{\neg, \wedge, \vee\}$ любую булеву функцию можно представить в виде суперпозиции дизъюнкции, конъюнкции и отрицания.

В качестве функциональных элементов (ФЭ) можно рассматривать любые булевы функции, при этом их можно соединять друг с другом, подавая выходы одних элементов на входы других (суперпозиция булевых функций). Полученная схема из функциональных элементов (СФЭ) является математической моделью некоторого дискретного устройства. Итак, для построения СФЭ необходим исходный «строительный материал»: полюсы (будем обозначать $\circ$), соответствующие входным переменным булевой функции, и набор функциональных элементов. Кроме этого, должны быть определены правила, позволяющие собирать из ФЭ схемы:

1) каждому полюсу ставится в соответствие одна из булевых переменных $x_1, x_2, ..., x_n$ (разным полюсам – разные переменные);

2) каждому ФЭ с k входами ставится в соответствие БФ $f(x_1, x_2, ..., x_k)$;

3) выходы одних ФЭ можно подавать на входы других ФЭ;

4) выделены некоторые точки схемы, называемые ее выходами.

На рисунке 2.12 приведен пример СФЭ с двумя полюсами x_1, x_2 и выходом y, составленной из трех ФЭ. Эта схема реализует булеву функцию:

$$f(x_1, x_2) = \overline{x_2} \vee x_1 x_2.$$

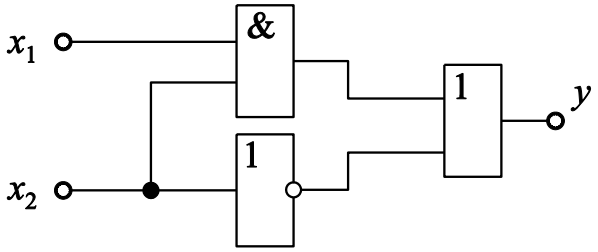


Рис. 2.12 – Пример схемы из функциональных элементов

СФЭ может содержать и один, и не содержать ни одного ФЭ (в последнем случае в ее состав будут входить только полюсы), но гораздо часто встречаются СФЭ, содержащие большое число ФЭ. Для таких СФЭ рассматриваются задачи анализа и синтеза СФЭ.

Задача анализа СФЭ заключается в следующем: дана СФЭ; требуется выяснить, какие БФ реализуются на выходах схемы.

Задача синтеза: задан набор ФЭ $f_1, f_2, ..., f_r$ и булева функция $f(x_1, ..., x_n)$; требуется, используя только ФЭ заданного типа, построить СФЭ, на одном из выходов которой реализуется БФ f.

При решении задачи синтеза вначале выясняется, можно ли реализовать БФ f с помощью только заданного набора ФЭ. Если система БФ $\{f_1, ..., f_r\}$ является полной, то ответ на этот вопрос будет положительным. Полноту системы можно проверить по теореме Поста.

Контрольные вопросы по главе 2

1. Сколько может быть булевых функций двух аргументов?
2. Какие булевы функции называются элементарными?
3. С помощью каких операций получают неэлементарные булевы функции?

4. Какую систему булевых функций называют полной?
5. Является ли самодвойственной булева функция «эквивалентность»?

3 Логика предикатов

3.1 Основные понятия логики предикатов

Известно, что математическая логика – это логика, развиваемая с помощью математических методов. В то же время этот термин имеет и другой смысл: математическая логика – это логика, используемая в математике. Центральная идея математической логики состоит в том, чтобы записывать математические утверждения в виде последовательностей символов и оперировать ими по формальным правилам. При этом правильность рассуждений можно проверять механически, не вникая в их смысл. Не всякие высказывания и не любые логические рассуждения могут быть описаны на языке логики высказываний. Иногда высказывания касаются свойств объектов или отношений между объектами. Кроме того, необходимо иметь возможность утверждать, что любые или какие-то объекты обладают определенными свойствами или находятся в некоторых отношениях. Поэтому следует расширить логику высказываний и построить такую логическую систему, в рамках которой можно было бы исследовать структуру и содержание тех высказываний, которые в рамках алгебры высказываний считались бы элементарными. Такой логической системой является логика предикатов, а алгебра высказываний – ее составной частью [2].

Логика предикатов *– это расширение возможностей логики высказываний, позволяющее строить высказывания с учетом свойств изучаемых объектов или отношений между ними.*

Понятие *предиката* восходит к Аристотелю. Предикат – это высказывание, содержащее неизвестную (или несколько неизвестных). Сам Аристотель ограничился в своей логике рассмотрением предикатов только от одной переменной (одноместных предикатов). Но позднее, после работ Дж. Буля, в рассмотрение вошли и предикаты от нескольких переменных.

Помимо элементарных высказываний в логике предикатов рассматриваются высказывания, отнесенные к предмету, т. е. высказывания разделяются на субъект и предикат.

S есть P:

все S есть P; *некотрые S есть P*;

все S не есть P; *некотрые S не есть P*.

Субъект – это то, о чем что-то утверждается, *предикат* – это то, что утверждается о субъекте. *Логика предикатов* – это расширение алгебры высказываний за счет использования предикатов в роли логических функций.

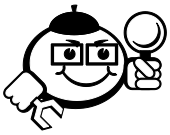
Одноместным предикатом $P(x)$ *называется произвольная функция одной переменной x, значениями которой являются высказывания об объектах, представляющих значения аргумента.*

Следовательно, одноместный предикат – это произвольная функция переменной x, определенная на некотором множестве и принимающая (логические) значения из множества $\{0,1\}$. Множество M, на котором определен предикат $P(x)$, называется *предметной областью*, или *областью определения предиката*, а сама переменная – *предметной переменой*.

Предметная область (область определения) предиката *– это множество M, на котором определен предикат.*

Предметные переменные *– это элементы множества M, на котором определен предикат.*

Таким образом, одноместный предикат $P(x)$ – это утверждение об объекте x, где x рассматривается как переменная. При фиксации значения переменной об утверждении $P(x)$ можно сказать, истинно оно или ложно. То есть если в $P(x)$ вместо x подставить конкретный изучаемый объект a, то получаем высказывание, принадлежащее алгебре высказываний.



Пример 3.1

Установить истинностное значение предиката $P(x)$ = «Город x является столицей России», заданного на множестве M = {Томск, Кемерово, Москва}.

Решение.

Обозначим a_1 = «Томск», a_2 = «Кемерово», a_3 = «Москва».

Подставим значения a_1, a_2, a_3 в предикат $P(x)$:

$P(a_1)$ = «Город Томск является столицей России» = 0;

$P(a_2)$ = «Город Кемерово является столицей России» = 0;

$P(a_3)$ = «Город Москва является столицей России» = 1.

Областью истинности *предиката* $P(x)$*, заданного на множестве M, называется совокупность всех* x *из M, при которых данный предикат обращается в истинное высказывание.*

Область истинности обозначим I_P, где $I_P = \{x \in M \mid P(x) \equiv 1\}$. Иными словами, область истинности предиката есть подмножество его предметной области (рис. 3.1), на котором данный предикат принимает значение «истина».

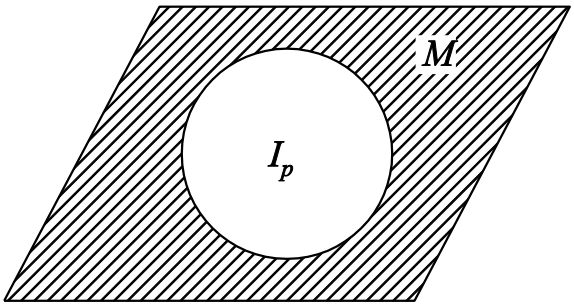
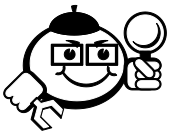


Рис. 3.1 – Область истинности предиката $P(x)$



Пример 3.2

Определить область истинности предиката $P(x)$= «Число *x* является четным», заданного на множестве M = {1, 2, 3, 4, 5}.

Решение.

$P(1)$ = «Число 1 является четным» = 0;

$P(2)$ = «Число 2 является четным» = 1;

$P(3)$ = «Число 3 является четным» = 0;

$P(4)$ = «Число 4 является четным» = 1;

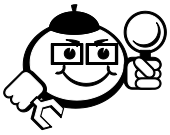
$P(5)$ = «Число 5 является четным» = 0;

Таким образом, получаем, что I_P = {2, 4}.

Предикат $P(x)$*, определенный на множестве* M*, называется* ***тождественно истинным****, если* $I_P = M$.

Предикат $P(x)$*, определенный на множестве* M*, называется* ***тождественно ложным****, если* $I_P = \varnothing$.

Предикат $P(x)$*, определенный на множестве* M*, называется* ***выполнимым****, если существует по меньшей мере один элемент* $x_i \in M$*, при котором* $P(x_i) = 1$.



Пример 3.3

Определить к какому классу предикатов относятся предикаты $P(x) = «x^2 - 4 = 0»$ и $Q(x) = «x^2 + 4 = 0»$, определенные на множестве R.

Решение.

Определим I_P: $x^2 - 4 = 0 \Rightarrow x_1 = 2;\ x_2 = -2$. Таким образом, $I_p = \{2, -2\}$ и предикат $P(x) = «x^2 - 4 = 0»$ является выполнимым.

Определим I_P: $x^2 + 4 = 0 \Rightarrow x^2 = -4$. У данного уравнения вещественных корней нет, $I_Q = \varnothing$ и данный предикат является тождественно ложным.

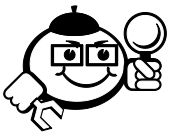
Чтобы задать предикат от n аргументов (n-местный предикат), прежде всего следует указать множества $M_1 \times ... \times M_n$ – области изменения *предметных переменных* $x_1, ..., x_n$.

n-местным предикатом *называется произвольная функция переменных* $P(x_1, ..., x_n)$*, определенная на множестве* $M = M_1 \times ... \times M_n$ *и принимающая (логические) значения из множества* $\{0, 1\}$.

Например, $P(x, y)$ = « y делится на x без остатка» – предикат от двух переменных, или двуместный предикат, где x и y могут принимать значения или из множества натуральных чисел, или из множества целых чисел, или, возмож-

но, из множества многочленов. При этом, вообще говоря, получаются различные предикаты.

Таким образом, *n*-местный предикат – это утверждение об объектах $x_1, ..., x_n$, где $x_1, ..., x_n$ рассматриваются как переменные. Важно отметить, что при замене переменных некоторыми значениями из области определения предикат превращается в высказывание, являющееся истинным или ложным. Значение истинности этого высказывания называется значением предиката от данных переменных. Например, при подстановке в предикат $P(x_1, ..., x_n)$ значений переменных $x_1 = a_1, ..., x_n = a_n$ мы получим высказывание $P(a_1, ..., a_n)$.



Пример 3.4

Найти область истинности предиката $P(x, y) = «x - y^2 \geq 0»$ и изобразить его на координатной плоскости.

Решение.

Неравенство, составляющее исходный предикат, ограничивает часть плоскости, заключенной между ветвями параболы $x \geq y^2$ (рис. 3.2).

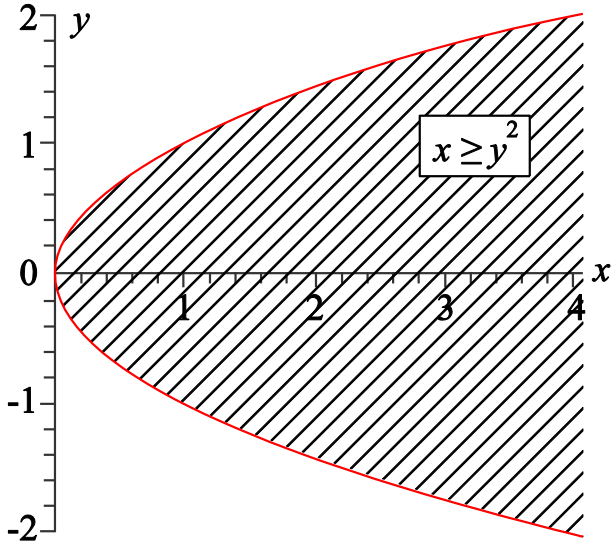


Рис. 3.2 – Область истинности предиката $P(x, y) = «x - y^2 \geq 0»$

3.2 Логические операции над предикатами

Предикаты, так же, как и высказывания, принимают два истинностных значения «истина» и «ложь», поэтому к ним применимы все операции алгебры

высказываний. Пусть на некотором множестве M определены два предиката $P(x)$ и $Q(x)$.

Конъюнкцией двух предикатов – $P(x)$ *и* $Q(x)$ – *называется новый предикат* $P(x) \wedge Q(x)$, *который принимает значение «истина» при тех и только тех значениях* $x \in M$, *при которых каждый из предикатов принимает значение «истина», а значение «ложь» – во всех остальных случаях.*

Областью истинности предиката $P(x) \wedge Q(x)$ является пересечение областей истинности обоих предикатов, то есть $I_{P \wedge Q} = I_P \cap I_Q$ (рис. 3.3).

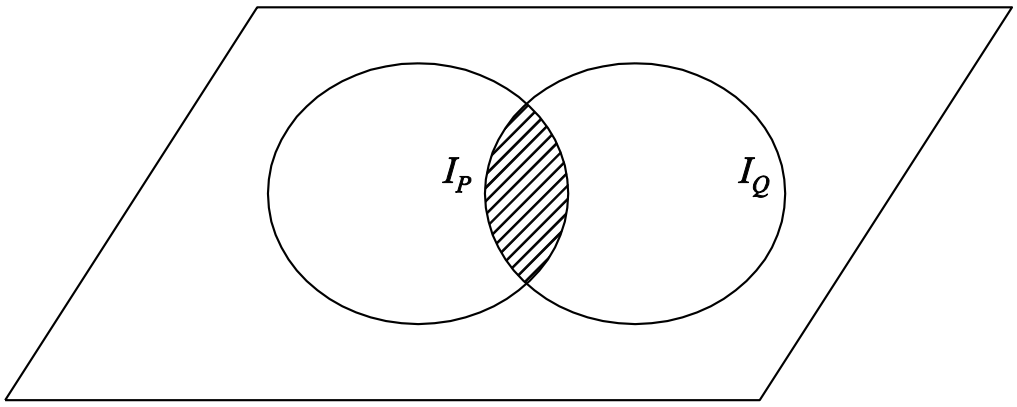
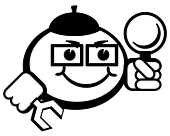


Рис. 3.3 – Область истинности предиката $P(x) \wedge Q(x)$



Пример 3.5

Найдите область истинности предиката $P(x) \wedge Q(x)$, заданного над множеством $M = \{1, 2, 3, 4, \ldots, 10\}$, если $P(x) =$ «x – нечетное число»; $Q(x) =$ «x – кратно 3».

Решение.

$P(x) \wedge Q(x) =$ {«x – нечетное число и x – кратно 3}. Определим области истинности предикатов $P(x)$ и $Q(x)$.

$P(x) = 1$ при $x \in \{1, 3, 5, 7, 9\}$, следовательно $I_P = \{1, 3, 5, 7, 9\}$. Аналогичным образом определяем $I_Q = \{3, 6, 9\}$.

$$I_{P \wedge Q} = I_P \cap I_Q = \{1, 3, 5, 7, 9\} \cap \{3, 6, 9\} = \{3, 9\}.$$

Дизъюнкцией двух предикатов $P(x)$ *и* $Q(x)$ *называется новый предикат* $P(x) \vee Q(x)$, *который принимает значение «ложь» при тех и только тех значениях* $x \in M$, *при которых каждый из предикатов принимает значение «ложь», и принимает значение «истина» во всех остальных случаях.*

Областью истинности предиката $P(x) \vee Q(x)$ является объединение областей истинности обоих предикатов, то есть $I_{P \vee Q} = I_P \cup I_Q$ (рис. 3.4).

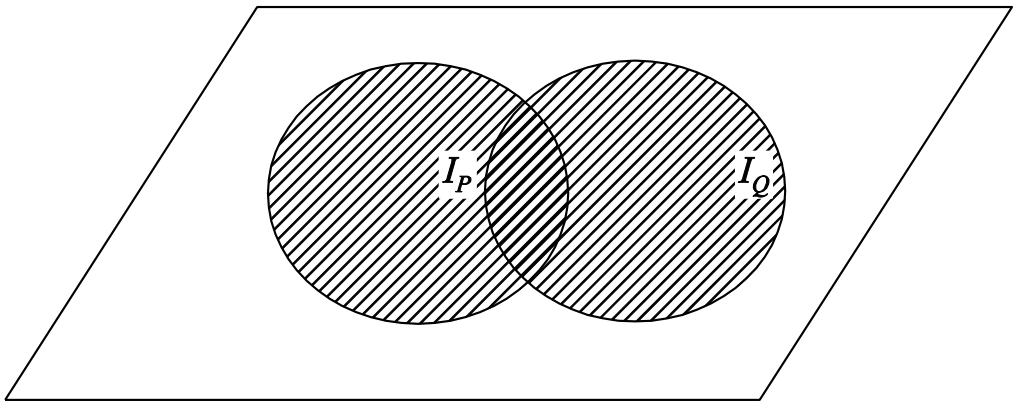
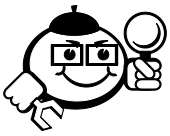


Рис. 3.4 – Область истинности предиката $P(x) \vee Q(x)$



Пример 3.6

Найдите множества истинности предиката $P(x) \vee Q(x)$, заданного над множеством M = {1, 2, 3, 4, …, 10}, если $P(x)$ = «x – четное число»; $Q(x)$ = = «x – кратно 3».

Решение.

$P(x) \vee Q(x)$ = {«x – четное число или x – кратно 3}. Определим области истинности предикатов $P(x)$ и $Q(x)$.

$P(x) = 1$ при $x \in \{3, 4, 6, 8, 10\}$, следовательно $I_P = \{2, 4, 6, 8, 10\}$. Аналогичным образом определяем $I_Q = \{3, 6, 9\}$.

$$I_{P \vee Q} = I_P \cup I_Q = \{2, 3, 4, 6, 8, 9, 10\}.$$

Отрицанием предиката $P(x)$ *называется новый предикат* $\overline{P(x)}$*, который принимает значение «истина» при всех значениях* $x \in M$*, при которых предикат* $P(x)$ *принимает значение «ложь», и принимает значение «ложь» при тех значениях* $x \in M$*, при которых предикат* $P(x)$ *принимает значение «истина»*

Областью истинности предиката $\overline{P(x)}$ является дополнение множества истинности предиката $P(x)$ до множества M, то есть $I_{\overline{P}} = M \setminus I_P$.

Импликацией предикатов $P(x)$ *и* $Q(x)$ *называется новый предикат* $P(x) \to Q(x)$*, который является ложным при тех и только тех значениях* $x \in M$*, при которых одновременно* $P(x)$ *принимает значение «истина», а* $Q(x)$ *принимает значение «ложь», и принимает значение «истина» во всех остальных случаях.*

Так как при каждом фиксированном x справедлива равносильность $P(x) \to Q(x) \equiv \overline{P(x)} \vee Q(x)$, то область истинности предиката $P(x) \to Q(x)$ является объединением дополнения области истинности предиката $P(x)$ до множества M и области истинности предиката $Q(x)$, то есть $I_{P \to Q} = I_{\overline{P}} \cup I_Q$.

Эквиваленцией предикатов $P(x)$ и $Q(x)$ называется новый предикат $P(x) \leftrightarrow Q(x)$, который принимает значение «истина» при тех значениях $x \in M$, при которых истинностные значения предикатов $P(x)$ и $Q(x)$ одинаковы.

Так как при каждом фиксированном x справедлива равносильность $P(x) \leftrightarrow Q(x) \equiv (P(x) \wedge Q(x) \vee (\overline{P(x)} \wedge \overline{Q(x)})$, то область истинности предиката $P(x) \leftrightarrow Q(x)$ является $I_{P \leftrightarrow Q} = (I_P \cap I_Q) \cup (I_{\overline{P}} \cap I_{\overline{Q}})$.

3.3 Кванторные операции

Рассмотренные выше логические операции над предикатами в определенном смысле аналогичны соответствующим операциям над высказываниями.

Специфика природы предикатов позволяет ввести такие операции над ними, которые не имеют аналогов среди операций над высказываниями. Имеются в виду две кванторные операции над предикатами (или операции квантификации) – квантор всеобщности (общности) и квантор существования. Для их обозначения используются символы:

- $\forall$ – квантор всеобщности, перевернутая первая буква *A* английского слова *All* (все);
- $\exists$ – квантор существования, перевернутая первая буква *E* английского слова *Exist* (существовать).

Кванторы в явном виде впервые были введены немецким математиком Готлобом Фреге в работе «Исчисление понятий» в 1879 г. В 1885 г. английский логик Чарльз Пирс ввел термины «квантор», «квантификация», происшедшие соответственно от лат. *quantun* – «сколько» и лат. *quantun* + *facio* – «делать». Это означает, что квантор показывает, о скольких (всех или некоторых) объектах говорится в том или ином предложении. Символику для кванторов в виде перевернутых латинских букв ввел итальянский математик Дж. Пеано в 90-е гг. XIX в. После использования кванторов математиками Пеано, Шредером, Расселом они стали широко использоваться [7].

Квантор *– это общее название для логических операций, ограничивающих область истинности какого-либо предиката.*

Пусть $P(x)$ – одноместный предикат, определенный на множестве M. Под выражением $\forall xP(x)$ понимают высказывание, истинное, если $P(x)$ истинно для каждого элемента, и ложное в противном случае. Иными словами, истинность высказывания означает, что область истинности предиката совпадает с областью изменения переменной. Читается это высказывание: «для всякого x, такого что $P(x)$ истинно».

Операцией ***связывания квантором всеобщности*** *называется правило, по которому каждому одноместному предикату $P(x)$, определенному на множестве M, сопоставляется высказывание, обозначаемое $\forall xP(x)$, которое истинно в том и только в том слу-*

чае, когда предикат $P(x)$ *тождественно истинен, и ложно в противном случае.*

. . . . . .

$$\lambda\left(\forall x P(x)\right)=\begin{cases}1, \text{ если } P(x) - \text{тождественно истинный предикат},\\ 0, \text{ если } P(x) - \text{выполнимый предикат}.\end{cases}$$

Обозначение $\forall x$ называют квантором всеобщности по переменной x. Это высказывание уже не зависит от *x*. Переменную *x* в предикате $P(x)$ называют *свободной*, а в высказывании $\forall x P(x)$ – связанной квантором всеобщности.

Под выражением $\exists x P(x)$ понимают высказывание, истинное, если существует $x \in M$, для которого истинно, и ложное в противном случае. Иными словами, истинность высказывания означает, что область истинности предиката не пуста. Читается это высказывание: «существует *x*, при котором $P(x)$ истинно».

. . . . . .

Операцией ***связывания квантором существования*** *называется правило, по которому каждому одноместному предикату* $P(x)$, *определенному на множестве* M, *ставится в соответствие высказывание, обозначаемое* $\exists x P(x)$, *которое ложно в том и только в том случае, когда* $P(x)$ *тождественно ложен, и истинно в противном случае.*

. . . . . .

$$\lambda\left(\exists x P(x)\right)=\begin{cases}0, \text{ если } P(x) - \text{тождественно ложный предикат},\\ 1, \text{ если } P(x) - \text{выполнимый предикат}.\end{cases}$$

Обозначение $\exists x$ называют квантором существования по переменной *x*. Это высказывание уже не зависит от *x*. Переменную *x* в предикате $P(x)$ называют *свободной*, а в высказывании $\exists x P(x)$ – связанной квантором существования.

. . . . . .

В формулах вида $\forall x F$ *или* $\exists x F$ *формула* F *называется* ***областью действия квантора*** $\forall x$ *или* $\exists x$ *соответственно.*

. . . . . .

Вхождение предметной переменной в формулу будет связанным, если эта переменная находится в области действия квантора по этой переменной.

Полезно отметить, что если некоторый предикат $P(x)$ определен на конечном множестве $M = \{a_1, ..., a_k\}$, то справедливы следующие тождества:

$$\forall x P(x) \equiv P(a_1) \wedge ... \wedge P(a_k),$$

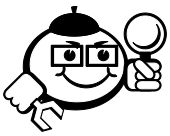
$$\exists x P(x) \equiv P(a_1) \vee ... \vee P(a_k).$$

Таким образом, кванторы можно рассматривать как обобщения логических связок. В случае предикатов, определенных на бесконечных множествах, квантор всеобщности обобщает конъюнкцию, а квантор существования – дизъюнкцию.

Кванторные операции применяются и к многоместным предикатам. Применение кванторной операции к предикату $P(x, y)$ по переменной x ставит в соответствие двухместному предикату $P(x, y)$ одноместный предикат $\forall x P(x, y)$ или $\exists x P(x, y)$, зависящий от y и не зависящий от x. К двухместному предикату можно применить кванторные операции по обеим переменным. Тогда получим восемь различных высказываний:

1) $\forall x \forall y P(x, y)$;
2) $\exists x \forall y P(x, y)$;
3) $\forall x \exists y P(x, y)$;
4) $\exists x \exists y P(x, y)$;
5) $\forall y \forall x P(x, y)$;
6) $\exists y \forall x P(x, y)$;
7) $\forall y \exists x P(x, y)$;
8) $\exists y \exists x P(x, y)$.

Изменение порядка следования кванторов изменяет смысл высказывания и его логическое значение.



Пример 3.7

Пусть $P(x,y)$ = «x является матерью y». Проанализируйте смысл предикатов $\forall y \exists x P(x,y)$ и $\exists x \forall y P(x,y)$.

Решение.

$\forall y \exists x P(x,y)$ =«У каждого человека есть мать».

$\exists x \forall y P(x,y)$ = «Существует мать всех людей».

Таким образом, перестановка кванторов изменяет смысл высказывания и его логическое значение (первое высказывание истинно, второе – ложно).

3.4 Формулы логики предикатов

Как уже отмечалось, расширение логики высказываний до логики предикатов получается за счет включения в формулы утверждений, являющихся предикатами.

В логике предикатов пользуются следующей символикой:

1) предметные переменные: $x, y, z ... x_i, y_i, z_i ... (i \in N)$;
2) нульместные предикатные переменные: $P, Q, R ... \ P_i, Q_i, Z_i ... (i \in N)$;
3) n-местные ($n \geq 1$) предикатные переменные с указанием числа свободных мест в них: $P(x_1, ..., x_n)$, $Q(x_1, ..., x_n)$ … $P_i(x_1, ..., x_n)$, $Q_i(x_1, ..., x_n) ... (i \in N)$;
4) символы логических операций: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$;
5) кванторы: $\exists, \forall$;
6) вспомогательные символы: (,) – скобки и запятые.

Формула логики предикатов определяется индуктивно по следующей схеме:

1. Всякая логическая переменная (т. е. нульместный предикат) есть формула.
2. Если P пользуются следующей символикой n-местный предикат, то $P(x_1, ..., x_n)$ – формула. Все переменные $x_1, ..., x_n$ – свободные переменные, связанных переменных в этой формуле нет.

3. Если A – формула, то $\overline{A}$ – формула с теми же свободными и связанными переменными, что и в формуле A.
4. Если A и B – формулы, причем нет таких переменных, которые были бы связанными в одной формуле и свободными в другой, то выражения $(A \wedge B), (A \vee B), (A \to B), (A \leftrightarrow B)$ – формулы, в которых свободные переменные формул и остаются свободными, а связанные переменные формул и остаются связанными.
5. Если A – формула, содержащая свободную предметную переменную x, то $\forall A$ и $\exists A$ – тоже формулы, причем переменная x в них связана. Остальные же переменные, которые в формуле были свободны, остаются свободными и в новых формулах. Переменные, которые были связаны в A, связаны и в новых формулах.
6. Других формул, кроме построенных по правилам пяти предыдущих пунктов, нет.

Из этого определения ясно, что всякая формула алгебры высказываний является формулой логики предикатов. Как обычно, часть скобок, определяющих порядок действий в формуле, можно опускать.

Формула $P(x_1,...,x_n)$, где P – n-местный предикат, называется *элементарной* (или атомарной). Например: $P; Q(x,y,z); R(x_1,x_2)$.

Формулы, не являющиеся элементарными, называются *сложными* (или составными). Например: $\exists y P(x,y,z);\ \forall x \exists y P(x,y,z);\ \exists x P(x) \vee Q(y)$.

Причем в формуле $\exists y P(x,y,z)$ предметная переменная y связана квантором существования, а переменные x, y – свободные.

Формулы логики предикатов, в которых нет свободных предметных переменных, называются ***замкнутыми.***

Примеры открытых формул: $Q(x,y,z); R(x_1,x_2); \exists y P(x,y,z)$.

Формулы логики предикатов, содержащие свободные предметные переменные, называются ***открытыми.***

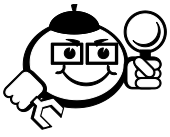
Примеры замкнутых формул: $P; \forall x Q(x); \exists x_1 \forall x_2 R(x_1,x_2)$.

Если в формулу логики предикатов вместо каждой предикатной переменной подставить конкретный предикат, определенный на некотором выбранном множестве M, то формула превратится в конкретный предикат, заданный над множеством M. При этом, если исходная формула была замкнутой, то полученный конкретный предикат окажется нульместным, т. е. будет высказыванием. Если же исходная формула была открытой, т. е. содержала свободные вхождения предметных переменных, то в результате подстановки получим предикат, зависящий от некоторых предметных переменных. Если теперь подставить вместо этих предметных переменных конкретные предметы из множества M, то полученный предикат, а следовательно и исходная формула превратятся в конкретное высказывание.

Превращение формулы логики предикатов в высказывание описанным выше способом (а также само получаемое высказывание) называется *интерпретацией* этой формулы на множестве M.

Итак, если формула логики предикатов замкнутая, т. е. не содержит свободных предметных переменных, то ее интерпретация состоит из одного этапа и сводится к подстановке вместо всех предикатных переменных конкретных предикатов, в результате чего формула превращается в конкретное высказывание (нульместный предикат).

Если же формула логики предикатов открытая, т. е. содержит ряд свободных предметных переменных, то ее интерпретация состоит из двух этапов. Во-первых, вместо всех предикатных переменных необходимо подставить конкретные предикаты, в результате чего формула превратится в конкретный предикат, зависящий от такого количества предметных переменных, сколько было свободных предметных переменных в исходной формуле. Во-вторых, нужно придать значение каждой предметной переменной, от которой зависит получившийся предикат, в результате чего этот предикат (и, значит, вся исходная формула) превратится в конкретное высказывание (истинное или ложное).



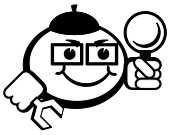
Пример 3.8

Приведите две различные интерпретации замкнутой формулы $\forall x \exists y P(x, y)$.

Решение.

В первом случае в качестве множества M возьмем множество всех мужчин, а вместо предикатной переменной $P(x,y)$ подставим конкретный предикат, определенный на M: $P(x,y)$ = «x есть отец y». Тогда исходная формула превратится в следующее высказывание: $\forall x \exists y P(x,y)$ = «У каждого мужчины есть сын». Очевидно, что это ложное высказывание.

Во втором случае возьмем в качестве M множество N всех натуральных чисел, а вместо предикатной переменной $P(x,y)$ подставим предикат «$x < y$», определенный на N^2. Тогда исходная формула превратится в (очевидно, истинное) высказывание: $\forall x \exists y P(x,y)$ = «Для каждого натурального числа существует большее по сравнению с ним натуральное число».



Пример 3.9

Приведите интерпретацию открытой формулы: $\exists z\big(P(x,y,z) \to Q(x,y,z)\big) \to R$.

Решение.

В качестве множества M возьмем множество N всех натуральных чисел. Вместо предикатных переменных $P(x,y,z)$ и $Q(x,y,z)$ подставим трехместные предикаты «$x \cdot y = z$» и «$x + y = z$» соответственно, а вместо нульместного предиката R подставим (ложное) высказывание «2 = 4». Тогда данная формула превратится в двухместный предикат (от предметных переменных x, y):

$$\exists z\big((x \cdot y = z) \to (x + y = z)\big) \to (2 = 4).$$

Посмотрим, в какие высказывания может превращаться данный предикат при подстановке вместо его переменных x, y конкретных предметов (чисел) из N. Нетрудно понять, что двухместный предикат $\exists z\big((x \cdot y = z) \to (x + y = z)\big)$ превращается в истинное высказывание при любой подстановке вместо его предметных переменных x, y натуральных чисел. В самом деле, для натуральных m и n получаем высказывание:

$$\exists z\big((m \cdot n = z) \to (m + n = z)\big).$$

Одноместный предикат (зависит от z) $(m \cdot n = z) \to (m + n = z)$, стоящий под знаком квантора $\exists z$, выполним, потому что всегда можно найти такое натуральное число k, что $m \cdot n \neq k$ и $m + n \neq k$, т. е. высказывания $m \cdot n = k$

и $m+n=k$ будут ложны, а значит, высказывание $(m\cdot n=z)\to(m+n=z)$ – истинно.

Следовательно, высказывание $\exists z\big((m\cdot n=z)\to(m+n=z)\big)$ истинно. Поэтому высказывание $\exists z\big((x\cdot y=z)\to(x+y=z)\big)\to(2=4)$, в которое превращается данный предикат, ложно.

Итак, исходная открытая формула логики предикатов превращена в тождественно ложный предикат. Нетрудно понять, что если вместо предикатных переменных $P(x,y,z)$ и $Q(x,y,z)$ подставить только что рассмотренные предикаты, а вместо нульместной предикатной переменной R – любое истинное высказывание, то исходная формула превратится в тождественно истинный предикат.

. . . . . .

Рассмотрим классификацию формул логики предикатов.

Формула логики предикатов называется ***выполнимой*** *(опровержимой) на множестве M, если при некоторой подстановке вместо предикатных переменных конкретных предикатов, заданных на этом множестве, она превращается в* ***выполнимый*** *(опровержимый) предикат.*

. . . . . .

Другими словами, формула выполнима (опровержима) на M, если существует истинная (ложная) ее интерпретация на *M*. Формула из примера 3.8 является как выполнимой, так и опровержимой.

Формула логики предикатов называется ***тождественно истинной*** *на множестве M, если при всякой подстановке вместо предикатных переменных любых конкретных предикатов, заданных на этом множестве, она превращается в тождественно истинный предикат.*

Формула логики предикатов называется ***тождественно ложной*** *на множестве M, если при всякой подстановке вместо предикатных переменных любых конкретных предикатов, заданных на этом множестве, она превращается в тождественно ложный предикат.*

. . . . . .

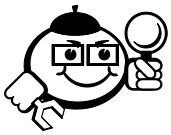
Так, формула из примера 3.9 не является тождественно истинной, потому что при одной подстановке она превратилась в тождественно истинный предикат, при другой подстановке – в предикат тождественно ложный. Поэтому данная формула является выполнимой формулой логики предикатов.

3.5 Равносильные формулы логики предикатов

Две формулы, F и H, логики предикатов называются ***равносильными на множестве*** *M, если при любой подстановке в эти формулы вместо предикатных переменных любых конкретных предикатов, определенных на M, формулы превращаются в равносильные предикаты.*

Две формулы логики предикатов ***равносильны всюду****, если они равносильны на любых множествах.*

Равносильность формул F и H опознают: $F \equiv H$. Рассмотрим пример проверки равносильности двух формул логики предикатов.



Пример 3.10

Являются ли равносильными формулами логики предикатов формулы $F = \forall x\big(P(x) \vee Q(x)\big)$ и $H = \forall x\big(P(x) \vee \forall x Q(x)\big)$?

Решение.

Подставим вместо предикатных переменных $P(x)$ и $Q(x)$ конкретные предикаты $A(x)$ и $B(x)$, определенные на множестве N соответственно, где $A(x)$ есть «x – четно», а $B(x)$ есть «x – нечетно». Тогда левая формула превратится в высказывание (нульместный предикат) «каждое натуральное число либо нечетно, либо четно», которое истинно. Правая формула превращается в высказывание (нульместный предикат) «либо каждое натуральное число четно, либо каждое натуральное число нечетно», которое ложно.

Нетрудно понять на основании признака равносильности формул логики высказываний, что формулы F и H равносильны тогда и только тогда, когда формула $F \leftrightarrow H$ является тавтологией:

$$F \equiv H \Leftrightarrow F \leftrightarrow H \equiv 1.$$

Как и в алгебре высказываний, можно заменять одну равносильную формулу другой. Переход от одной равносильной формулы к другой называется *равносильным преобразованием* исходной формулы. В процессе равносильных преобразований формул логики предикатов могут использоваться равносильности, известные из алгебры высказываний.

Рассмотрим законы равносильных преобразований формул логики предикатов. Пусть $P(x)$ и $Q(x)$ – два произвольных одноместных предиката и $R(x, y)$ двухместный предикат, а S – нульместный предикат. Тогда имеют место следующие равносильности:

1) $\overline{\forall x P(x)} \equiv \exists x \overline{P(x)}$ (*первый закон де Моргана для кванторов*);

2) $\overline{\exists x P(x)} \equiv \forall x \overline{P(x)}$ (*второй закон де Моргана для кванторов*);

3) $\forall x P(x) \equiv \overline{\exists x \overline{P(x)}}$;

4) $\exists x P(x) \equiv \overline{\forall x \overline{P(x)}}$;

(Соотношения 1–4 показывают, что можно выразить один квантор через другой.)

5) $\forall x\big(P(x) \wedge S\big) \equiv \forall x P(x) \wedge S$;

6) $\forall x\big(S \wedge P(x)\big) \equiv S \wedge \forall x P(x)$;

7) $\forall x\big(P(x) \vee S\big) \equiv \forall x P(x) \vee S$;

8) $\forall x\big(S \vee P(x)\big) \equiv S \vee \forall x P(x)$;

(Соотношения 5–8 показывают, что произвольный нульместный предикат (высказывание) можно вносить под знак квантора всеобщности и выносить из-под знака этого квантора в конъюнкции и дизъюнкции.)

9) $\exists x\big(P(x) \wedge S\big) \equiv \exists x P(x) \wedge S$;

10) $\exists x\big(S \wedge P(x)\big) \equiv S \wedge \exists x P(x)$;

11) $\exists x\big(P(x) \vee S\big) \equiv \exists x P(x) \vee S$;

12) $\exists x\big(S \vee P(x)\big) \equiv S \vee \exists x P(x)$;

(Соотношения 9–12 показывают, что произвольный нульместный предикат (высказывание) можно вносить под знак квантора существования и выносить из-под знака этого квантора в конъюнкции и дизъюнкции.)

13) $\forall x\big(P(x) \to S\big) \equiv \exists x P(x) \to S$;

14) $\forall x\big(S \to P(x)\big) \equiv S \to \forall x P(x)$;

15) $\exists x\big(P(x) \to S\big) \equiv \forall x P(x) \to S$;

16) $\exists x\big(S \to P(x)\big) \equiv S \to \exists x P(x)$;

17) $\forall x S \equiv S$;

18) $\exists x S \equiv S$;

19) $\forall x\big(P(x) \wedge Q(x)\big) \equiv \forall x\big(P(x) \wedge Q(x)\big)$ (*дистрибутивность квантора всеобщности относительно конъюнкции*);

20) $\exists x\big(P(x) \vee Q(x)\big) \equiv \exists x\big(P(x) \vee Q(x)\big)$ (*дистрибутивность квантора существования относительно дизъюнкции*);

21) $\exists x\big(P(x) \wedge Q(x)\big) \to (\exists x\big(P(x) \wedge \exists x Q(x)\big) \equiv 1$;

22) $\big(\forall x P(x) \vee \forall x Q(x)\big) \to \forall x\big(P(x) \vee Q(x)\big) \equiv 1$;

23) $\forall x P(x) \equiv \forall y P(y)$;

24) $\exists x P(x) \equiv \exists y P(y)$;

25) $\forall x \forall y R(x, y) \equiv \forall y \forall x P(x, y)$ (*коммутация одноименных кванторов*);

26) $\exists x \exists y R(x, y) \equiv \exists y \exists x P(x, y)$ (*коммутация одноименных кванторов*);

27) $\exists y \forall x R(x, y) \to \forall x \exists y R(x, y) \equiv 1$;

28) $\big(\forall x P(x)\big) \vee \big(\forall x Q(x)\big) \equiv \forall x \forall y\big(P(x) \vee Q(y)\big)$;

29) $\big(\exists x P(x)\big) \wedge \big(\exists x Q(x)\big) \equiv \exists x \exists y\big(P(x) \wedge Q(y)\big)$.

Соотношения 21, 22 и 27 не будут верны, если поменять направления стрелок на противоположные.

3.6 Нормальная форма записи формул логики предикатов

Равносильные преобразования позволяют приводить формулы к тому или иному более удобному виду. Один из таких видов носит название приведенной формы.

Приведенной формой *для формулы логики предикатов называется такая равносильная ей формула, в которой из операций алгебры высказываний имеются только операции* $\{\neg, \wedge, \vee\}$*, причем знаки отрицания относятся лишь к предикатным переменным и к высказываниям.*

Для каждой формулы логики предикатов существует приведенная форма.

Проведем доказательство методом математической индукции по числу логических связок в формуле (включая кванторы общности и существования). Если формула не имеет логических связок, т. е. является элементарной, то она сама имеет приведенную форму.

Предположим, что всякая формула, содержащая не больше $k-1$ логических связок, обладает приведенной формой. Покажем теперь, что приведенной формой обладает также и всякая формула, содержащая k логических связок. Пусть F – такая формула. Тогда, на основании определения равносильных формул логики предикатов, она имеет один из следующих видов:

$$\neg F_1, (F_1 \wedge F_2), (F_1 \vee F_2), (F_1 \to F_2), (F_1 \leftrightarrow F_2), \forall x F_1, \exists x F_1.$$

Каждая из формул F_1, F_2 содержит логических связок не более $k-1$, а поэтому, по предположению индукции, обладает приведенной формой. Пусть F_1^* и F_2^* – приведенные формы для формул F_1, F_2. Отсюда формулы $(F_1^* \wedge F_2^*), (F_1^* \vee F_2^*), \forall x F_1^*, \exists x F_1^*$ являются приведенными формами для формул $(F_1 \wedge F_2), (F_1 \vee F_2), \forall x F_1, \exists x F_1$ соответственно.

Остается рассмотреть случаи, когда F имеет один из следующих видов: $\neg F_1$, $(F_1 \to F_2), (F_1 \leftrightarrow F_2)$. Пусть F есть $\neg F_1$. Тогда формула $\neg F_1^*$ может не быть приведенной формой для формулы $\neg F_1$. Строго говоря, для этого случая следует провести доказательство также методом математической индукции по числу логических связок формулы F_1^*. Если с элементарная формула, т. е. F_1^* – предикатная переменная P, то $(\neg F_1)^*$ есть $\neg P$ – приведенная форма. Если же F_1^* – составная формула, то задача сводится к «пронесению» знака отрицания $\{\neg\}$ через кванторы и операции $\{\wedge, \vee\}$ (другие логические операции не входят в приведенную форму $\neg F_1^*$).

Это пронесение осуществляется на основании законов равносильных преобразований логики предикатов и алгебры высказываний, называемых законами де Моргана. Итак, если F есть $\neg F_1$ и F_1 обладает приведенной формой $\neg F_1^*$, то мы сможем найти приведенную форму и для F.

Далее, пусть F есть $(F_1 \to F_2)$. Тогда на основании закона № 11 замены импликации формула F равносильна формуле $(\neg F_1 \vee F_2)$. Заменив формулы F_1 и F_2 на равносильные им приведенные формы F_1^* и F_2^* соответственно, получим равносильную формулу $(F_1^* \vee F_2^*)$, которая не является приведенной. Но можно найти для формулы $\neg F_1^*$ приведенную форму F_1^{**}. Тогда ясно, что формула $(F_1^{**} \vee F_2^*)$ имеет приведенный вид и равносильна исходной формуле F.

Наконец, если F есть $(F_1 \leftrightarrow F_2)$, то на основании закона равносильности F равносильна формуле $((F_1 \to F_2) \wedge (F_2 \to F_1))$. В предыдущем абзаце было показано, как найти приведенные формы $(F_1^{**} \vee F_2^*)$ и $(F_2^{**} \vee F_1^*)$ для формул $(F_1 \to F_2)$ и $(F_2 \to F_1)$ соответственно. Тогда ясно, что формула $(F_1^{**} \vee F_2^*)$ и $(F_2^{**} \vee F_1^*)$ имеет приведенный вид и равносильна исходной формуле F.

Итак, в любом случае формула F обладает приведенной формой. Теорема доказана.

. . . . . .

Еще одним удобным видом формулы, к которому ее можно привести равносильными преобразованиями, является предваренная нормальная форма.

. . . . . .

Предваренной нормальной формой *для формулы логики предикатов называется такая ее приведенная форма, в которой все кванторы стоят в ее начале, а область действия каждого из них распространяется до конца формулы.*

. . . . . .

Это формула вида $(K_1x_1)...(K_mx_m)(F(x_1,...,x_n))$, где K_i есть один из кванторов $\forall$ или $\exists$ $(i=1,...,m), m \le n$, причем формула F не содержит кванторов и является приведенной формулой. (Заметим, что кванторы в формуле могут вообще отсутствовать.)

Для каждой формулы логики предикатов существует предваренная нормальная форма.

Проведем доказательство по индукции, следуя правилу построения формул логики предикатов.

Если формула элементарная, то она сама представляет собой предваренную нормальную форму. Поскольку каждая формула F равносильна формуле, полученной из более простых формул F_1, F_2 с помощью операций $\{\neg, \wedge, \vee, \forall, \exists\}$ (операции $\{\rightarrow, \leftrightarrow\}$ выражаются через $\{\neg, \wedge, \vee\}$), то теперь остается научиться находить предваренные нормальные формы для формул:

$$\neg F_1, (F_1 \wedge F_2), (F_1 \vee F_2), \forall x F_1, \exists x F_1.$$

Если известны предваренные нормальные формы F_1^* и F_2^* формул F_1 и F_2 соответственно. Пусть, например, F_1^* имеет вид: $\forall x_1 \forall x_2 \exists x_3 .... \exists x_n (G_1(x_1,...,x_n))$, а F_2^* имеет вид $\exists y_1 \forall y_2 .... \forall y_m (G_2(y_1,...,y_m))$.

Рассмотрим формулу $\neg F_1^*$. Поскольку $F_1 \equiv F_1^*$, то $\neg F_1 \equiv \neg F_1^*$. Но формула $\neg F_1^*$, в свою очередь, равносильна, на основании законов де Моргана для кванторов, формуле: $\exists x_1 \exists x_2 \forall x_3 .... \forall x_n (\neg G_1(x_1,...,x_n))$.

Остается по законам де Моргана для конъюнкции и дизъюнкции пронести знак отрицания до предикатных переменных: $\neg G_1 \equiv \neg G_1^*$. Тогда $\neg F_1$ будет иметь предваренную нормальную форму: $\exists x_1 \exists x_2 \forall x_3 .... \forall x_n (G_1^*(x_1,...,x_n))$.

Далее покажем, как отыскивается предваренная нормальная форма для F, если F есть $(F_1 \wedge F_2)$. Поскольку при переименовании связанной переменной формула, очевидно, переходит в равносильную, то можно считать, что переменные $x_1, ..., x_n$ не входят в формулу F_2^*, а переменные $y_1, ..., y_m$ не входят в F_1^*. Ясно, что $F_1 \wedge F_2 \equiv F_1^* \wedge F_2^*$, но последняя формула еще не представляет собой предваренной нормальной формы. Покажем, как ее можно преобразовать к такой форме. На основании равносильности (дистрибутивность квантора всеобщности относительно конъюнкции) формула $F_1^* \wedge F_2^*$ равносильна формуле:

$$\forall x_1 \left[\forall x_2 \exists x_3 .... \exists x_n \left(G_1(x_1, ..., x_n) \right) \wedge \exists y_1 \forall y_2 ... \forall y_m \left(G_2(y_1, ..., y_m) \right) \right].$$

Теперь можно производить равносильные преобразования под знаком квантора $\forall x_1$ в квадратных скобках, потому что в результате связывания квантором по одной и той же переменной x_1 двух равносильных формул мы снова получим равносильные формулы. Тогда, на основании той же равносильности, последняя формула равносильна формуле:

$$\forall x_1 \forall x_2 \left[\exists x_3 .... \exists x_n \left(G_1(x_1, ..., x_n) \right) \wedge \exists y_1 \forall y_2 ... \forall y_m \left(G_2(y_1, ..., y_m) \right) \right].$$

И так далее. Наконец, на основании равносильности № 9 последняя формула равносильна формуле:

$$\forall x_1 \forall x_2 \exists x_3 .... \exists x_n \left[\left(G_1(x_1, ..., x_n) \right) \wedge \exists y_1 \forall y_2 ... \forall y_m \left(G_2(y_1, ..., y_m) \right) \right].$$

Таким же образом кванторы, стоящие перед формулой G_2, выносятся за квадратные скобки. В результате получим формулу:

$$\forall x_1 \forall x_2 \exists x_3 .... \exists x_n \exists y_1 \forall y_2 ... \forall y_m \left[\left(G_1(x_1, ..., x_n) \right) \wedge \left(G_2(y_1, ..., y_m) \right) \right],$$

равносильную формуле $(F_1 \wedge F_2)$ и являющуюся предваренной нормальной формой этой формулы.

Аналогичным образом при помощи законов равносильных преобразований можно построить предваренную нормальную форму для формулы $(F_1 \vee F_2)$, исходя из предваренных нормальных форм F_1^* и F_2^* формул F_1 и F_2 соответственно.

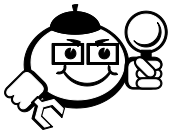
Наконец, нетрудно понять, что формула $\forall x F_1^*$ равносильна формуле $\forall x F_1$ и является ее предваренной нормальной формой. Аналогично, $\exists x F_1^*$ – предваренная нормальная форма для формулы $\exists x F_1$.

Итак, в каждом случае F обладает предваренной нормальной формой. Теорема доказана.

. . . . . .

Алгоритм получения формулы в предваренной нормальной форме:

1) перейти от операций $\{\to, \leftrightarrow\}$ к операциям $\{\neg, \wedge, \vee\}$ (используем законы замены импликации № 11 и замены эквиваленция № 12 алгебры высказываний);
2) внести все отрицания внутрь формулы, «приклеив» их к предикатным символам (используя законы де Моргана);
3) вынести все кванторы в начало формулы.

. . . . .  Пример 3.11 . . . . . .

Записать формулу логики предикатов $F = \neg\big(\exists x\big(P(x) \to \forall y P(y)\big)\big)$ в предваренной нормальной форме.

Решение.

В преобразованиях будем использовать законы логики высказываний и логики предикатов:

$F \equiv \neg\big(\exists x\big(P(x) \to \forall y Q(y)\big)\big) \equiv$ *закон замены импликации № 11*;

$\equiv \forall x\big(\neg\big(\neg P(x) \vee \forall y Q(y)\big)\big) \equiv$ *закон де Моргана относительно квантора*;

$\equiv \forall x\big(P(x) \wedge \neg\big(\forall y Q(y)\big)\big) \equiv$ *закон двойного отрицания*;

$\forall x\big(P(x) \wedge \big(\exists y \neg Q(y)\big)\big) \equiv$ *закон де Моргана относительно квантора*;

$\forall x \exists y\big(P(x) \wedge \neg Q(y)\big)$ – *предваренная нормальная форма*.

. . . . . .

Контрольные вопросы по главе 3

1. Что такое предметные переменные?
2. Что такое порядок (местность) предиката?
3. Что такое область истинности предиката?
4. Что такое выполнимая формула?
5. Как привести формулу логики предикатов к предваренной нормальной форме?

4 Теория алгоритмов

4.1 Характерные черты алгоритма

Современное формальное определение алгоритма было дано в 30–50-х гг. XX в. в работах А. Тьюринга, Э. Поста, А. Чёрча, Н. Винера, А. А. Маркова.

Само слово «алгоритм» происходит от имени учёного Абу Абдуллах Мухаммеда ибн Муса аль-Хорезми. Около 825 г. он написал сочинение, в котором впервые дал описание придуманной в Индии позиционной десятичной системы счисления. К сожалению, арабский оригинал книги не сохранился. Аль-Хорезми сформулировал правила вычислений в новой системе и, вероятно, впервые использовал цифру 0 для обозначения пропущенной позиции в записи числа (её индийское название арабы перевели как as-sifr или просто sifr, отсюда такие слова, как «цифра» и «шифр»). Приблизительно в это же время индийские цифры начали применять и другие арабские учёные.

Под понятием «алгоритм» подразумевают решение задач в виде точных последовательно выполняемых предписаний. Это интуитивное определение сопровождается описанием интуитивных свойств (признаков) алгоритмов: эффективность, определенность, конечность [9].

Эффективность *алгоритма – возможность исполнения предписаний за конечное время.*

Например, алгоритм – процедура, состоящая из конечного числа команд, каждая из которых выполняется механически за фиксированное время и с фиксированными затратами [3].

Функция алгоритмически эффективно вычислима, если существует механическая процедура, следуя которой для конкретных значений ее аргументов можно найти значение этой функции [11].

Определенность *алгоритма – возможность точного математического определения или формального описания содержания команд и последовательности их применения в этой процедуре.*

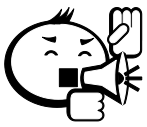
Конечность *алгоритма – выполнение алгоритма при конкретных исходных данных за конечное число шагов.*

Алгоритм *– это общий, единообразный, точно установленный способ решения любой задачи из данной массовой проблемы*.

Приведенное понятие нестрогое, но оно позволяет выделить некоторые характерные черты алгоритма:

1. Дискретность: каждая последующая величина получается из значений предыдущих по определенному закону и все величины получаются последовательно друг за другом.
2. Детерминированность: между всеми величинами, получаемыми алгоритмом, существует жесткая причинная связь и все последующие значения зависят от предыдущих.
3. Элементарность шагов алгоритма: закон получения последующей системы величин из предшествующей должен быть простым.
4. Массовость: начальная система величин выбирается из некоторого множества и начальные условия могут варьироваться в бесконечных пределах.
5. Результативность: конечный результат всегда должен быть получен.

Интуитивное определение алгоритма приемлемо, когда речь о найденном алгоритме решения конкретной задачи. В случае, когда возникает алгоритмическая проблема, решение которой не найдено, необходимо доказать либо существование алгоритма решения, либо его отсутствие. Для доказательства отсутствия (отрицание существования) алгоритма необходимо точное формальное определение.



Общая *теория алгоритмов* занимается проблемой эффективной вычислимости.

Разработано несколько формальных определений алгоритма, в которых эффективность и конечность вычислений могут быть определены количественно – числом элементарных шагов и объемом требуемой памяти.

Подобными моделями алгоритмических преобразований символьной информации являются:

- рекурсивные функции;
- машина Тьюринга;
- машина Поста;

- конечные автоматы;
- ассоциативное исчисление или нормальные алгоритмы Маркова.

Некоторые из этих моделей лежат в основе методов программирования и используются в алгоритмических языках. В современной *программной инженерии* алгоритмы как методы решения задач занимают ведущее место по сравнению с традиционной математикой. Причем не важно, существует или нет чистое алгоритмическое решение в абстрактных моделях алгоритмов. Если решение задачи необходимо, широко используется эвристика, а «доказательством» работоспособности алгоритма является успешное его тестирование.

4.2 Машины Тьюринга

Идея создания машины Тьюринга, предложенная английским математиком А. Тьюрингом в 30-х гг. XX века, связана с его попыткой дать точное математическое определение понятия алгоритма [8].

Машина Тьюринга (МТ) – это математическая модель идеализированной цифровой вычислительной машины.

Машина Тьюринга (рис. 4.1) является таким же математическим объектом, как функция, производная, интеграл, группа и т. д. Так же, как и другие математические понятия, понятие машины Тьюринга отражает объективную реальность, моделирует некие реальные процессы.

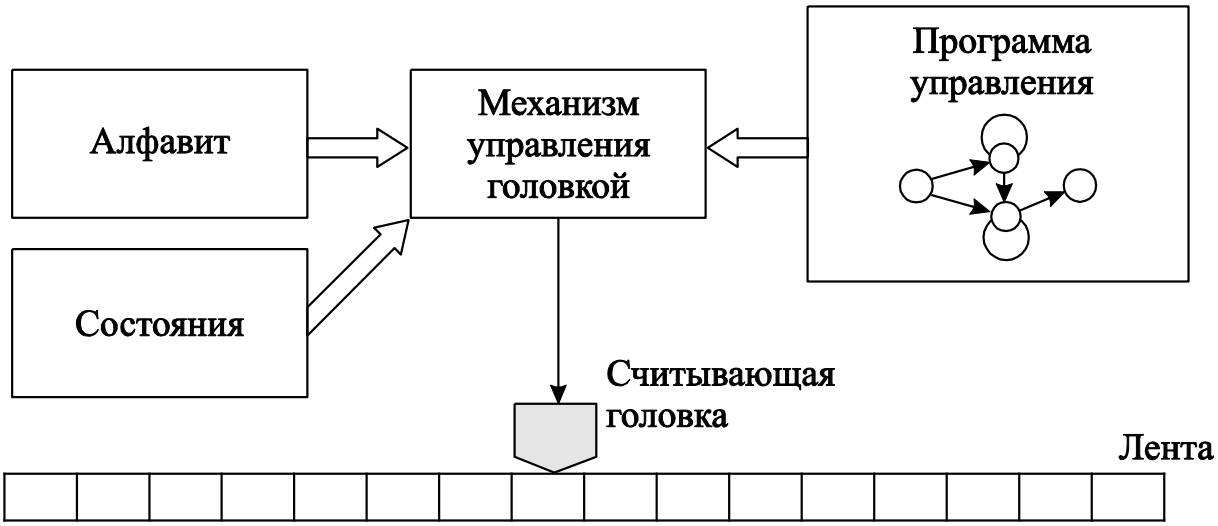


Рис. 4.1 – Машина Тьюринга

Для описания алгоритма МТ удобно представлять некоторое устройство, состоящее из четырех частей: ленты, считывающей головки, устройства управления и внутренней памяти.

1. *Лента* предполагается потенциально бесконечной, разбитой на ячейки (равные клетки). При необходимости к первой или последней клетке, в которой находятся символы, достраивается пустая клетка. Машина работает во времени, которое считается – дискретным, и его моменты пронумерованы 1, 2, 3, … . В каждый момент лента содержит конечное число клеток. В ячейки в дискретный момент времени может быть записан только один символ (буква) из внешнего алфавита $A=\{a_0, a_1, a_2, ..., a_{n-1}\}, n \geq 2$. Пустая ячейка обозначается символом a_0, а сам символ a_0 называется пустым, при этом остальные символы называются непустыми. В этом алфавите A в виде слова (конечного упорядоченного набора символов) кодируется та информация, которая подается в МТ. Машина «перерабатывает» информацию, поданную в виде слова, в новое слово.

2. *Считывающая головка* (некоторый считывающий элемент) перемещается вдоль ленты так, что в каждый момент времени она обозревает ровно одну ячейку ленты. Головка может считывать содержимое ячейки и записывать в нее новый символ из алфавита A. В одном такте работы она может сдвигаться только на одну ячейку вправо (R), влево (L) или оставаться на месте (C). Обозначим множество перемещений (сдвигов) головки $D = \{R, L, C\}$. Если в данный момент времени t головка находится в крайней клетке и сдвигается в отсутствующую клетку, то пристраивается новая пустая клетка, над которой окажется головка в момент $t+1$.

3. *Внутренняя память* машины представляет собой некоторое конечное множество внутренних состояний $Q=\{q_0, q_1, ..., q_m\}, m \geq 1$. Будем считать, что мощность $|Q| \geq 2$.

Два состояния машины имеют особое значение:

- q_1 – начальное внутреннее состояние (начальных внутренних состояний может быть несколько),
- q_0 – заключительное состояние, или стоп-состояние (заключительное состояние всегда одно).

В каждый момент времени МТ характеризуется положением головки и внутренним состоянием. Например, над ячейкой (напротив которой находится головка) указывается внутреннее состояние машины (рис. 4.2).

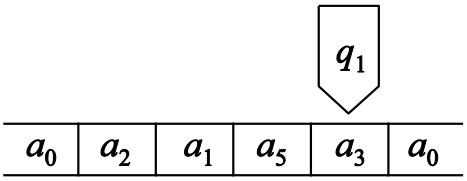


Рис. 4.2 – Работа машины Тьюринга

4. *Устройство управления* в каждый момент t в зависимости от считываемого в этот момент символа на ленте и внутреннего состояния машины выполняет следующие действия:

- изменяет считываемый в момент t символ a_i на новый символ a_j (в частности, оставляет его без изменений, т. е. $a_i = a_j$);
- передвигает головку в одном из следующих направлений: R, L, C; изменяет имеющееся в момент t внутреннее состояние машины q_i на новое q_j, в котором будет машина в момент времени $t+1$ (может быть, что $q_i = q_j$).

Такие действия устройства управления называют командой, которую можно записать в виде:

$$a_i q_i \Rightarrow a_j D q_j \quad \text{или} \quad a_i q_i \Rightarrow a_j \begin{Bmatrix} R \\ L \\ C \end{Bmatrix} q_j ,$$

где q_i – внутреннее состояние машины в данный момент; a_i – считываемый в этот момент символ; a_j – символ, на который изменяется символ a_i (может быть $a_i = a_j$); D или символы R, L, C указывают направление движения головки; q_j – внутреннее состояние машины в следующий момент (может быть $q_i = q_j$).

Выражения $a_i q_i$ и $a_j D q_j$ называются левой и правой частями этой команды соответственно. Число команд, в которых левые части попарно различны, является конечным числом, так как множества $Q \setminus \{q_0\}$ и A конечны.

Не существует команд с одинаковыми левыми частями, т. е. если программа МТ содержит выражения $a_i q_i \Rightarrow a_j D q_j$ и $a_t q_t \Rightarrow a_k D q_k$, то $q_i \neq q_t$ или $a_i \neq a_t$ и $D \in \{R, L, C\}$.

Совокупность всех команд называется *программой машины Тьюринга*. Максимальное число команд в программе равно $(n+1)\cdot m$, где $n+1=|A|$ и $m+1=|Q|$. Считается, что заключительное состояние команды q_0 может стоять только в правой части команды, начальное состояние q_1 может стоять как в левой, так и в правой части команды.

Выполнение одной команды называется *шагом*. Вычисление (или работа) машины Тьюринга является последовательностью шагов одного за другим без пропусков, начиная с первого.

Итак, МТ задана, если известны четыре конечных множества: внешний алфавит A , внутренний алфавит Q , множество D перемещений головки и программа машины, представляющая собой конечное множество команд.

Работа машины Тьюринга полностью определяется заданием в первый (начальный) момент:

1) слова на ленте, т. е. последовательности символов, записанных в клетках ленты (слово получается чтением этих символов по клеткам ленты слева направо);
2) положения головки;
3) внутреннего состояния машины.

Совокупность этих трех условий (в данный момент) называется *конфигурацией* (в данный момент). Обычно в начальный момент внутренним состоянием машины является q_1, а головка находится либо над первой слева, либо над первой справа клеткой ленты.

Заданное слово на ленте с начальным состоянием q_1 и положение головки над первым словом называется *начальной конфигурацией*.

Другими словами, в начальный момент конфигурация представима в следующем виде: на ленте, состоящей из некоторого числа клеток, в каждой клетке записан один из символов внешнего алфавита A , головка находится над первой слева или первой справа клеткой ленты и внутренним состоянием машины является q_1. Получившееся в результате реализации этой команды слово на ленте и положение головки называется *заключительной конфигурацией*.

Например, если в начальный момент на ленте записано слово $a_1a_3a_2a_1a_1$, то начальная конфигурация будет иметь вид (рис. 4.3). Слово считывается слева направо, где a_0 – пустая клетка (пробел между словами).

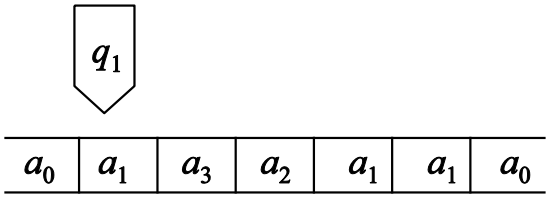


Рис. 4.3 – Начальная конфигурация машины Тьюринга

Работа машины Тьюринга состоит в последовательном применении команд, причем применение той или команды определяется текущей конфигурацией. Так, в приведенном выше примере должна примениться команда с левой частью $a_1 q_1$.

Итак, зная программу и задав начальную конфигурацию, полностью определяем работу машины над словом в начальной конфигурации.

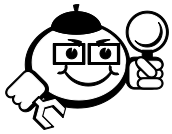
Если в работе машины Тьюринга в некоторый момент t выполняется команда, правая часть которой содержит q_0, то в такой момент работа машины считается законченной и говорят, что машина применима к слову на ленте в начальной конфигурации. В самом деле, q_0 не встречается в левой части ни одной команды – этим и объясняется название q_0 «заключительное состояние». Результатом работы машины в таком случае считается слово, которое будет записано на ленте в заключительной конфигурации, т. е. в конфигурации, в которой внутреннее состояние машины есть q_0. Если же в работе машины ни в один из моментов не реализуется команда с заключительным состоянием, то процесс вычисления будет бесконечным. В этом случае говорят, что машина неприменима к слову на ленте в начальной конфигурации.

Программу машины Тьюринга можно представить в виде двумерной таблицы (табл. 4.1), называемой *функциональной схемой*, в каждой клетке которой записываются отдельные команды. Работа машины Тьюринга полностью определяется ее программой.

Таблица 4.1 – Функциональная схема машины Тьюринга

Состояния	**Символы внешнего алфавита**			
	a_0	a_1	…	a_n
q_1	$a_2 L q_3$	$a_1 R q_2$	…	$a_2 L q_1$
q_2	$a_1 C q_0$	$a_2 C q_1$	…	$a_1 C q_2$

Состояния	**Символы внешнего алфавита**			
	a_0	a_1	…	a_n
…	…	…	…	…
q_m	a_1Pq_3	a_0Rq_{m-1}	…	$a_{n-1}Rq_1$



Пример 4.1

Пусть $A=\{a_0,a_1,a_2\}$, $Q=\{q_0,q_1,q_3,q_4\,R,\,L,\,C\}$ и машина Тьюринга управляется функциональной схемой, приведенной в таблице 4.2.

Таблица 4.2 – Функциональная схема машины Тьюринга

Состояния	**Символы внешнего алфавита**		
	a_0	a_1	a_2
q_1	a_2Lq_3	a_1Rq_2	a_2Lq_1
q_2	a_1Cq_0	a_2Cq_1	a_1Cq_2
q_3	a_0Rq_0	a_1Rq_4	a_2Cq_1
q_4	a_1Cq_3	a_0Rq_4	a_2Rq_4

Определить конечный результат работы машины, если начальная конфигурация имеет вид:

а) $\underset{q_1}{a_0}a_1a_2a_2a_0$; б) $\underset{q_1}{a_0}a_1a_0$.

Решение.

а) Рассмотрим работу машины пошагово:

1. Считывающим устройством обозревается буква a_2 (слово считывается слева направо, a_0 – пустая клетка), а машина находится в состоянии q_1: $\underset{q_1}{a_0}a_1a_2\underset{\Uparrow}{a_2}a_0$. При этом вырабатывается команда a_2Lq_1, т. е. считывающее устройство сдвигается влево, a_2 заменя-

ется на a_2, состояние q_1 меняется на q_1, получаем конфигурацию $a_0 \underset{q_1 \Uparrow}{a_1 a_2} a_2 a_0$.

2. Следующая конфигурация, по аналогии, будет: $a_0 \underset{\Uparrow\ q_1}{a_1 a_2} a_2 a_0$ (считывающее устройство сдвинулось влево). Теперь обозревается буква a_1, машина находится в состоянии q_1, т. е. вырабатывается команда $a_1 R q_2$ – считывающее устройство сдвигается вправо, a_1 заменяем на a_1, состояние q_1 меняется на q_2, получаем конфигурацию $a_0 \underset{q_2 \Uparrow}{a_1 a_2} a_2 a_0$.

3. Обозревается буква a_2, машина находится в состоянии q_2, т. е. вырабатывается команда $a_1 C q_2$ – считывающее устройство стоит на месте, a_2 заменяется на a_1, состояние q_2 меняется на q_2, получаем конфигурацию $a_0 \underset{q_2 \Uparrow}{a_1 a_1} a_2 a_0$.

4. Обозревается буква a_1, машина находится в состоянии q_2, т. е. вырабатывается команда $a_2 C q_1$ – считывающее устройство стоит на месте, a_1 заменяется на a_1, состояние q_2 меняется на q_1, получаем конфигурацию $a_0 \underset{q_1 \Uparrow}{a_1 a_2} a_2 a_0$.

Таким образом, мы пришли в начальное состояние. Процесс работы машины повторяется, и, следовательно, конечный результат не может быть получен. Данная машина Тьюринга неприменима к исходной информации.

б) Пусть начальная информация имеет вид $a_0 \underset{q_1}{a_1} a_0$. Тогда, действуя аналогично, придем к следующим конфигурациям:

1. $a_0 \underset{q_1 \Uparrow}{a_1} a_0$ обозревается буква a_1, машина находится в состоянии q_1, вырабатывается команда $a_1 R q_2$ – считывающее устройство сдви-

гается вправо, a_1 заменяется на a_1, состояние q_1 меняется на q_2, получаем конфигурацию $a_0 \underset{q_2}{a_1} a_0$.

2. Обозревается буква a_0, машина находится в состоянии q_2, вырабатывается команда $a_1 C q_0$ – считывающее устройство стоит на месте, a_0 заменяется на a_1, состояние q_2 меняется на q_0, получаем конфигурацию $a_0 \underset{q_0}{a_1} a_1 a_0$. Мы пришли в стоп-состояние q_0, следовательно, слово $a_1 a_1$ – результат ее работы и машина применима к исходной информации.

. . . . . .

Говорят, что непустое слово α в алфавите A воспринимается машиной в стандартном положении, если оно записано в последовательных клетках ленты, все другие клетки пусты и машина обозревает крайнюю клетку справа из тех, в которых записано слово α.

Если данное состояние описывается машинным словом M, то машинное слово, описывающее следующее состояние машины, будет обозначаться через $M^{(1)}$. Далее аналогично $M^{(i+1)} = \left(M^{(i)}\right)^{(1)}$, $i = 0, 1, 2, \ldots$. Переход машины Тьюринга из начального в последующие состояния можно изобразить *в виде цепочки* слов $M \vdash M^{(1)} \vdash M^{(2)} \vdash \ldots$

Чтобы описывать работу машины Тьюринга более удобным образом, текущее состояние машины пишут не внизу алфавита, а перед обозреваемой ячейкой. Например, пусть $A = \{0,1\}, Q = \{q_0, q_1, q_2\}$, q_0 – символ остановки. Начальная информация: $q_1 11$. Тогда программа строится следующим образом:

$$q_1 0 \to q_2 R,\ q_2 0 \to q_0 1,\ q_1 1 \to q_1 R,\ q_2 1 \to q_2 R.$$

Машинная арифметика сначала уточняет понятие алгоритма, а затем определяет класс вычислимых функций. Основная идея этого направления заключается в том, что алгоритмические процессы – это процессы, которые могут имитироваться на специально построенных машинах, которые описываются в точных математических терминах. В результате оказывается, что сложные процессы можно моделировать на простых устройствах. Всякий алгоритм может быть задан некоторой функциональной схемой и реализован в соответствующей машине Тьюринга. Эта гипотеза называется *тезисом Тьюринга*.

Работа машины Тьюринга складывается из тактов, по ходу которых происходит преобразование начальной информации I_1 в промежуточную. В качестве начальной информации на ленту можно подать любую конечную систему знаков внешнего алфавита, расставленного произвольным образом по ячейкам. При этом работа машины Тьюринга может заканчиваться так:

- После конечного числа тактов машина останавливается в q_0-состоянии. При этом на ленте оказывается преобразованная информация. В этом случае говорят, что машина применима к начальной информации I_1 и преобразует ее в результирующую информацию I_2.
- Машина никогда не останавливается (не переходит в q_0-состояние) – машина неприменима к начальной информации I_1.

4.3 Рекурсивные функции

Всякий алгоритм однозначно ставит в соответствие исходным данным (в случае, если он определен на них) результат, поэтому с каждым алгоритмом однозначно связана функция, которую он вычисляет. Исследование проблемы остановки машины Тьюринга показывает, что не для всякой функции существует вычисляющий ее алгоритм. Поэтому в 30-х гг. XX в. была создана *теория рекурсивных функций*. В этой теории, как и вообще в теории алгоритмов, принят подход, основной чертой которого является то, что все множество исследуемых объектов (в нашем случае функции) строится из конечного числа исходных объектов – базиса – с помощью простых операций, эффективная выполнимость которых достаточно очевидна. Операции над функциями в дальнейшем будем называть операторами.

Данное направление связывает понятие алгоритма с традиционным представлением – процедурами вычисления значений числовых функций. Основной теоретической моделью этого типа являются рекурсивные функции – исторически первая формализация понятия алгоритма. А. Черчем и К. Геделем выделен класс частично рекурсивных функций, имеющих строгое математическое определение.

Рассмотрим класс *вычислимых функций*, предложенный в 1930-х гг. (Гедель, Клини, Черч) в качестве уточнения понятия алгоритма, – класс *частично рекурсивных функций*.

Пусть функция y зависит от целочисленных аргументов $x_1, x_2, ..., x_n$.

Функция $y = f(x_1, x_2, ..., x_n)$ *называется* ***эффективно вычислимой****, если существует алгоритм, позволяющий вычислять ее значения.*

Данный класс определяется путем указания конкретных исходных функций и фиксированного множества операций получения новых функций из заданных. В качестве базисных эффективно вычислимых функций берутся следующие:

1) $O(x) = 0$ – оператор аннулирования (нуль-функция);
2) $\lambda(x) = x + 1$ – оператор сдвига (функция следования*)*;
3) $I_n^n(x_1, x_2, ..., x_n) = x_m, 1 \le m \le n$ – оператор проектирования функции выбора аргументов.

Допустимыми операциями над функциями являются операции суперпозиции (подстановки), рекурсии и минимизации.

Операция суперпозиции. Пусть даны n-местная функция g и n функций $f_1, ..., f_n$. Считаем, что функции $f_1, ..., f_n$ зависят от одних и тех же переменных $x_1, ..., x_m$. Это можно сделать путем введения фиктивных переменных. Суперпозицией (подстановкой) функций g и $f_1, ..., f_n$ называется функция: $h(x_1, ..., x_m) = g\big(f_1(x_1, ..., x_m), ..., f_n(x_1, ..., x_m)\big)$.

Если среди заданных функций имеются частичные, то и функция h будет частичной. Функция h на наборе переменных $x_1, ..., x_m$ определена тогда и только тогда, когда определены все функции $f_1(x_1, ..., x_m), ..., f_n(x_1, ..., x_m)$ и функция g определена на наборе $f_1(x_1, ..., x_m), ..., f_n(x_1, ..., x_m)$. Операцию суперпозиции обозначают $h = S(g, f_1, ..., f_n)$.

Операция рекурсии. Пусть заданы n-местная функция $g(x_1, ..., x_n)$ и $(n+2)$-местная функция $h(x_1, ..., x_n, y, z)$. Определим $(n+1)$-местную функцию f индуктивным образом с помощью соотношения:

$$f(x_1, ..., x_n, 0) = g(x_1, ..., x_n),$$

$$f(x_1, ..., x_n, y+1) = h\big(x_1, ..., x_n, y, f(x_1, ..., x_n, y)\big).$$

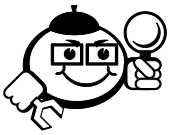
Ясно, что данные соотношения однозначно определяют функцию f. Если функции g и h частичные, то $f(x_1, ..., x_n, y+1)$ считается определенной в том и только в том случае, когда определены $f(x_1, ..., x_n, y)$ и $h(x_1, ..., x_n, y, t)$ при $t = f(x_1, ..., x_n, y)$. Таким образом, если $f(x_1, ..., x_n, y_0)$ не определено, то и $f(x_1, ..., x_n, y)$ не определено при $y > y_0$. Тогда про функцию f говорят, что она получена рекурсией из функций g и h, и обозначают: $f = R(g, h)$.

Операция минимизации. Пусть задана *n*-местная функция $g(x_1, ..., x_{n-1}, y)$. Зафиксируем набор $(x_1, ..., x_{n-1}, x_n)$ и рассмотрим уравнение относительно y: $g(x_1, ..., x_{n-1}, y) = x_n$.

Будем решать данное уравнение, вычисляя последовательно $g(x_1, ..., x_{n-1}, 0)$, $g(x_1, ..., x_{n-1}, 1)$, $g(x_1, ..., x_{n-1}, 2)$, ... и сравнивая с x_n. Наименьшее y, для которого выполнено исходное уравнение, обозначим через $\mu_y\left(g(x_1, ..., x_{n-1}, y) = x_n\right)$. При этом считаем, что y определено, если $g(x_1, ..., x_{n-1}, z)$ определено при всех $z \le y$. В противном случае считаем, что y не определено. Значение y есть функция f от переменных $x_1, ..., x_n$, про которую говорят, что она получена из функции g операцией минимизации.

Заметим, что определенные выше операции S и R, будучи примененными к всюду определенным функциям, дают всюду определенные функции. Операция M может давать частичные функции даже при применении к всюду определенным функциям.

*Функция $f(x_1, ..., x_n)$ называется **примитивно рекурсивной**, если она может быть получена из базисных функций $O(x), \lambda(x), I_m^n(x_1, ..., x_n)$ применением конечного числа раз операций суперпозиции, рекурсии и минимизации.*



Пример 4.2

Доказать общерекурсивность функции $\operatorname{sgn}(x) = \begin{cases} 1, & x \neq 0, \\ 0, & x = 0. \end{cases}$

Решение.

Функцию $\mathrm{sgn}(x)$ можно определить с помощью простейшей рекурсии таким образом:

$$\begin{cases} \mathrm{sgn}(0) = 0, \\ \mathrm{sgn}(y+1) = 1 \end{cases} \text{ или } \begin{cases} \mathrm{sgn}(0) = O(x), \\ \mathrm{sgn}(y+1) = C_1(y) \end{cases}, \text{ где } C_1(y) \equiv 1.$$

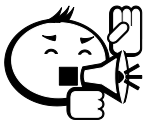
Тогда по определению функция $\mathrm{sgn}(x)$ общерекурсивна.

*Функция $f(x_1, ..., x_n)$ называется **частично рекурсивной**, если она определена не для всех значений аргументов.*

Иногда частично рекурсивные функции называют функциями, вычислимыми по Черчу.

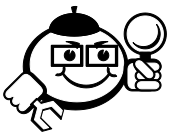
*Всюду определенная частично рекурсивная функция $f(x_1, ..., x_n)$ называется **общерекурсивной**.*

Обозначим: $Ч$ – класс частично рекурсивных функций, $Ч_0$ – класс общерекурсивных функций, $Ч_{\text{пр}}$ – класс примитивно рекурсивных функций. Класс частично рекурсивных функций – одно из главных понятий теории алгоритмов. Это объясняется тем, что какие бы классы точно очерченных «алгоритмов» до сих пор не рассматривались, во всех случаях оказывалось, что соответствующие числовые функции, вычислимые посредством алгоритмов этих классов, были частично рекурсивными. Поэтому общепринятой является гипотеза, формулируемая как *тезис Черча* (для частично рекурсивных функций):



Класс алгоритмически вычислимых функций совпадает с классом всех частично рекурсивных функций.

Принятие данного тезиса позволяет истолковывать доказательство, что некоторая функция не является частично рекурсивной, как доказательство отсутствия алгоритма вычисления ее значений.



Пример 4.3

Определить аналитический вид общерекурсивной функции:

$$\begin{cases} f(0,x) = x, \\ f(y+1,x) = f(y,x)+1. \end{cases}$$

Решение.

Очевидно, $\varphi(x) = x$, будем устанавливать функцию $\psi(x,y,z)$.

По определению примитивной рекурсии:

$$f(y+1,x) = \psi\big(y,\, f(y,x),\, x\big) = \left\langle \begin{array}{c} \textit{обозначим} \\ x = y \\ y = f(y,x) \\ z = x \end{array} \right\rangle = \psi(x,y,z).$$

Вернемся к функции $f(y+1,x) = f(y,x)+1$, принимая во внимание введенные обозначения $y = f(y,x)$, получаем:

$$f(y+1,x) = y+1.$$

Итак, $\psi(x,y,z) = y+1$. Тогда числовые значения также легко находятся:

$f(0,0) = 0, \quad f(0,1) = 1, \quad f(0,2) = 2,$

$f(1,0) = \psi(0,\, f(0,0),0) = \psi(0,0,0) = 0+1 = 1;$

$f(2,0) = \psi(1,\, f(1,0),0) = \psi(1,1,0) = 1+1 = 2;$

$f(3,0) = \psi(2,\, f(2,0),0) = \psi(2,2,0) = 2+1 = 3;$

$f(1,1) = \psi(0,\, f(0,1),1) = \psi(0,1,1) = 1+1 = 2;$

$f(2,1) = \psi(1,\, f(1,1),1) = \psi(1,2,1) = 2+1 = 3;$

$f(3,1) = \psi(2,\, f(2,1),1) = \psi(2,3,1) = 3+1 = 4;$

$f(1,2) = \psi(0,\, f(0,2),2) = \psi(0,2,2) = 2+1 = 3;$

$f(2,2) = \psi(1,\, f(1,2),2) = \psi(1,3,2) = 3+1 = 4;$

$f(3,2) = \psi(2\ f(2,2),2) = \psi(2,4,2) = 4+1 = 5.$

Выясним, как аналитически выглядит функция $f(y,x)$:

$$\left.\begin{array}{c} f(0,x)=0 \\ f(1,x)=1+f(0,x)=1+x \\ f(2,x)=2+f(1,x)=1+(1+x)=2+x \\ f(3,x)=1+f(2,x)=1+(2+x)=3+x \end{array}\right\} f(y,x)=y+x.$$

Таким образом, аналитический вид функции имеет вид: $f(y,x)=y+x$.

. . . . . .

4.4 Нормальные алгоритмы Маркова

Теория нормальных алгоритмов была разработана советским математиком А. А. Марковым (1903–1979) в конце 1940-х – начале 1950-х гг. Эти алгоритмы представляют собой некоторые правила по переработке слов в каком-либо алфавите, так что исходные данные и искомые результаты для алгоритмов являются словами в некотором алфавите.

Алфавитом называется любое непустое множество. Его элементы называются буквами, а любые последовательности букв – *словами* в данном алфавите. Для удобства рассуждений допускаются пустые слова (они не имеют в своем составе ни одной буквы). *Пустое слово* будем обозначать $\wedge$. Если A и B – два алфавита, причем $A \subseteq B$, то алфавит B называется *расширением* алфавита.

Слова будем обозначать латинскими буквами: P, Q, R … P_1, P_2, P_3…. Одно слово может быть составной частью другого слова. Тогда первое называется *подсловом* второго или вхождением во второе. Например, если A – алфавит русских букв, то можем рассмотреть такие слова: $P_1 = параграф$, $P_2 = граф$, $P_3 = ра$. Слово P_2 является подсловом слова P_1, а P_3 – подсловом P_1 и P_2, причем в P_1 оно входит дважды. Особый интерес представляет первое вхождение.

. . . . . .

Марковской подстановкой *называется операция над словами, задаваемая с помощью упорядоченной пары слов* (P, Q)*, состоящая в следующем. В заданном слове* R *находят первое вхождение слова* P *(если таковое имеется) и, не изменяя остальных частей слова* R*, заменяют в нем это вхождение словом* Q.

. . . . . .

Полученное слово называется *результатом* применения марковской подстановки (P, Q) к слову R. Если же первого вхождения P в слово R нет, то считается, что марковская подстановка (P, Q) неприменима к слову R.

Частными случаями марковских подстановок являются подстановки с пустыми словами: $(\wedge, Q),(P, \wedge),(\wedge, \wedge)$.

Примеры марковских подстановок рассмотрены в таблице 4.3. В каждой строке таблицы сначала дается преобразуемое слово, затем применяемая к нему марковская подстановка и, наконец, получающееся в результате слово.

Таблица 4.3 – Примеры марковских подстановок

Преобразуемое слово	**Марковская подстановка**	**Результат**
138 578 926	(8 578 9, 00)	130 026
тарарам	(ара, Λ)	трам
шрам	(ра, ар)	шарм
функция	(Λ, ζ-)	ζ-функция
логика		лог
книга		книга
поляна	(пор, т)	[*неприменима*]

Для обозначения марковской подстановки (P, Q) используется запись $P \to Q$. Она называется *формулой подстановки* (P, Q). Некоторые подстановки (P, Q) будем называть заключительными. Для обозначения таких подстановок будем использовать запись $P \to .Q$, называя ее *формулой заключительной подстановки*. Слово P называется левой частью, а Q – правой частью в формуле подстановки.

Упорядоченный конечный список формул подстановок

$$\begin{cases} P_1 \to (.)\, Q_1, \\ P_2 \to (.)\, Q_2, \\ \quad .... \\ P_r \to (.)\, Q_r \end{cases}$$

в алфавите называется *схемой* (или записью) нормального алгоритма в A, причем запись точки в скобках означает, что она может стоять в этом месте, а может отсутствовать. Данная схема определяет (детерминирует) алгоритм преобразования слов, называемый *нормальным алгоритмом Маркова*.

Нормальным алгоритмом Маркова в алфавите A называется следующее правило построения последовательности V_i слов в алфавите A, исходя из данного слова V в этом алфавите. В качестве начального слова V_0 последовательности берется слово V. Пусть для некоторого $i \geq 0$ слово V_i построено и процесс построения рассматриваемой последовательности еще не завершился. Если при этом в схеме нормального алгоритма нет формул, левые части которых входили бы в V_i, то V_{i+1} полагают равным V_i и процесс построения последовательности считается завершившимся.

Если же в схеме имеются формулы с левыми частями, входящими в V_i, то в качестве V_{i+1} берется результат марковской подстановки правой части первой из таких формул вместо первого вхождения ее левой части в слово V_i; процесс построения последовательности считается *завершившимся*, если на данном шаге была применена формула заключительной подстановки, и *продолжающимся* – в противном случае.

Если процесс построения упомянутой последовательности обрывается, то говорят, что рассматриваемый *нормальный алгоритм применим к слову* V. Последний член W последовательности называется *результатом применения нормального алгоритма к слову* V. Говорят, что *нормальный алгоритм перерабатывает* V и W.

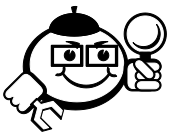
Последовательность V_i будем записывать следующим образом:

$$V_0 \Rightarrow V_1 \Rightarrow V_2 \Rightarrow \ldots \Rightarrow V_{m-1} \Rightarrow V_m,$$

где $V_0 = V$ и $V_m = W$.

Мы определили понятие нормального алгоритма в алфавите A. Если же алгоритм задан в некотором расширении алфавита A, то говорят, что он есть *нормальный алгоритм над* A.

Рассмотрим примеры нормальных алгоритмов.

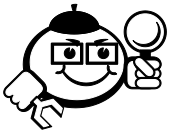


Пример 4.4

Пусть $A=\{a,b\}$ – алфавит. Рассмотрим следующую схему нормального алгоритма в A:

$$\begin{cases} a \to .\wedge \\ b \to b. \end{cases}$$

Всякое слово V в алфавите A, содержащее хотя бы одно вхождение буквы a, он перерабатывает в слово, получающееся из V вычеркиванием в нем самого левого (первого) вхождения буквы a. Пустое слово он перерабатывает в пустое. Например, $aabab \Rightarrow abab,\ ab \Rightarrow b,\ aa \Rightarrow a,\ bbab \Rightarrow bbb,\ baba \Rightarrow bba$.



Пример 4.5

Пусть $A=\{a_0, a_1, ..., a_n\}$ – алфавит. Рассмотрим схему:

$$\begin{cases} a_0 \to \wedge \\ a_1 \to \wedge \\ \quad .... \\ a_n \to \wedge \\ \wedge \to .\wedge . \end{cases}$$

Она определяет нормальный алгоритм, перерабатывающий всякое слово (в алфавите A) в пустое слово.

Например,

$$a_1a_2a_1a_3a_0 \Rightarrow a_1a_2a_1a_3 \Rightarrow a_2a_1a_3 \Rightarrow a_2a_3 \Rightarrow a_3 \Rightarrow \wedge;$$

$$a_0a_2a_2a_1a_3a_1 \Rightarrow a_2a_2a_1a_3a_1 \Rightarrow a_2a_2a_3a_1 \Rightarrow a_2a_2a_3 \Rightarrow a_2a_3 \Rightarrow a_3 \Rightarrow \wedge.$$

4.5 Классы сложности

В рамках классической теории алгоритмические задачи различаются по классам сложности: P-сложные, NP-сложные, экспоненциально сложные и др.

Классы сложности *– множества вычислительных задач, примерно одинаковых по сложности вычислений.*

Более узко, классы сложности – это множества предикатов (функций, получающих на вход слово и возвращающих ответ 0 или 1), использующих для вычисления примерно одинаковые количества ресурсов.

Каждый класс сложности определяется как множество предикатов, обладающих некоторыми свойствами. Классом сложности X называется множество предикатов $P(x)$, вычислимых на машинах Тьюринга и использующих для вычисления $O(f(n))$ ресурса, где n – длина слова x. В качестве ресурсов обычно берутся время вычисления (количество рабочих тактов машины Тьюринга) или рабочая зона (количество использованных ячеек на ленте во время работы).

Класс P – задачи, которые могут быть решены за время, полиномиально зависящее от объёма исходных данных, с помощью детерминированной вычислительной машины (например, машины Тьюринга).

Класс NP – задачи, которые могут быть решены за полиномиально выраженное время с помощью недетерминированной вычислительной машины, то есть машины, следующее состояние которой не всегда однозначно определяется предыдущими.

К классу NP относятся задачи, решение которых с помощью дополнительной информации полиномиальной длины, данной дополнительно, возможно проверить за полиномиальное время. В частности, к классу NP относятся все задачи, решение которых можно проверить за полиномиальное время. Класс P содержится в классе NP. Классическими примерами NP-задач являются задачи о коммивояжёре, нахождение гамильтонова цикла, раскраска вершин графа [12].

Поскольку класс P содержится в классе NP, принадлежность той или иной задачи к классу NP зачастую отражает текущее представление о способах решения данной задачи и носит неокончательный характер. В общем случае нет оснований полагать, что для той или иной NP-задачи не может быть найдено P-решение.

Вопрос о возможной эквивалентности классов P и NP (то есть о возможности нахождения P-решения для любой NP-задачи) считается одним из основных вопросов современной теории сложности алгоритмов.

Рассмотрим часто встречающиеся классы сложности в зависимости от числа входных данных n (в порядке нарастания сложности, т. е. увеличения времени работы алгоритма, при стремлении n к бесконечности): $O(1)$ – количество шагов алгоритма не зависит от количества входных данных. Обычно это алгоритмы, использующие определённую часть данных входного потока и игнорирующие все остальные данные.

Ряд алгоритмов имеют порядок, включающий $\log_2 n$, и называются логарифмическими. Эта сложность возникает, когда алгоритм неоднократно подразделяет данные на подсписки длиной 1/2, 1/4, 1/8 и так далее от оригинального размера списка. Логарифмические порядки возникают при работе с бинарными деревьями.

Алгоритм со сложностью $O(n)$ – алгоритм линейной сложности. Например, просмотр обложки каждой поступающей книги – для каждого входного объекта выполняется только одно действие.

Алгоритмы, имеющие порядок $O(n^2)$, являются квадратичными. К ним относятся наиболее простые алгоритмы сортировки; алгоритм Дейкстры – нахождение кратчайших путей в графе, n – число вершин графа; алгоритм Прима – построение минимального связывающего дерева, n – число вершин графа [10]. Квадратичные алгоритмы используются на практике только для относительно небольших значений n. Всякий раз, когда n удваивается, время выполнения такого алгоритма увеличивается на множитель четыре.

Алгоритм показывает кубическое время, если его порядок равен $O(n^3)$, и такие алгоритмы очень медленные. Всякий раз, когда n удваивается, время выполнения алгоритма увеличивается в восемь раз. Алгоритм Флойда – Уоршелла (динамический алгоритм для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа) – это алгоритм со сложностью порядка $O(n^3)$.

Алгоритм со сложностью $O(2^n)$ имеет экспоненциальную сложность. Такие алгоритмы выполняются настолько медленно, что они используются только при малых значениях n. Этот тип сложности часто ассоциируется с проблемами, требующими неоднократного поиска дерева решений. Алгоритмы со сложностью $O(n!)$ – факториальные алгоритмы, в основном используются в комбинаторике для определения числа сочетаний, перестановок.

В таблице 4.4 сравниваются значения n, n^2 и $n\log_2 n$. Результаты показывают, что более эффективным является алгоритм сортировки $O(n\log_2 n)$, чем обменная сортировка.

Например, в случае со списком из 10 000 элементов количество сравнений для обменной сортировки ограничивается величиной 100 000 000, тогда как более эффективный алгоритм имеет количество сравнений, ограниченное величиной 132 877,1. Новая сортировка приблизительно в 752 раза более эффективна.

Таблица 4.4 – Значения n, n^2 и $n\log_2 n$

n	n^2	$n\log_2 n$
5	25	11,6
10	100	33,2
100	1 000	664,3
1 000	1 000 000	9 965,7
10 000	100 000 000	132 877,1

В таблице 4.5 приводятся линейный, квадратичный, кубический, экспоненциальный и логарифмический порядки величины для выбранных значений n. Из таблицы 4.5 очевидно, что следует избегать использования кубических и экспоненциальных алгоритмов, если только значение *n* не мало.

Таблица 4.5 – Различные порядки величины для выбранных значений n

n	$\log_2 n$	$N\log_2 n$	n^2	n^3	2^n
2	1	2	4	8	4
4	2	8	16	64	16
8	3	24	64	512	256
16	4	64	256	4 096	65 536
32	5	160	1 024	32 768	4 294 967 296
128	7	896	16 384	2 097 152	3.4x10
1 024	10	10 240	1 048 576	1 073 741 824	1.8x10
65 536	16	1 048 576	4 294 967 296	2.8x10	Избегайте!

Важность проведения резкой границы между полиномиальными и экспоненциальными алгоритмами вытекает из сопоставления числовых примеров роста допустимого размера задачи с увеличением быстродействия (*Б*) используе-

мых ЭВМ. В таблице 4.6 указаны размеры задач, решаемых за одно и то же время (T) на ЭВМ с быстродействием ($Б_1$) при различных зависимостях сложности (Q) от размера n.

Эти примеры показывают, что, выбирая ЭВМ в k раз более быстродействующую, получаем увеличение размера решаемых задач при линейных алгоритмах в k раз, при квадратичных алгоритмах – в $k^{\frac{1}{2}}$ раз и т. д.

Таблица 4.6 – Сопоставление числовых примеров роста допустимого размера задачи с увеличением быстродействия используемых ЭВМ

$Q(n)$	$Б_1$	$Б = 100\ Б_1$	$Б = 1000\ Б_1$
n	n_1	$100n_1$	$1000\,n_1$
n^2	n_2	$10n_2$	$31{,}6n_2$
n^3	n_3	$4{,}64n_3$	$10n_3$
2^n	n_4	$6{,}64+ n_4$	$9{,}97+ n_4$

Исследования сложности алгоритмов позволили по-новому взглянуть на решение многих классических математических задач и найти для ряда таких задач решения, требующие меньше ресурсов, нежели традиционные.

Контрольные вопросы по главе 4

1. Какие направления в уточнении понятия алгоритма существуют?
2. Какой проблемой занимается теория алгоритмов?
3. Каковы составляющие машины Тьюринга?
4. Какая функция называется эффективно вычислимой?
5. Какие классы сложности алгоритмических задач существуют?

Заключение

Математическая логика в ее современном понимании возникла как средство обеспечить математику логически безупречными основаниями, окончательно оформить математическое доказательство, сделать его независимым от естественных языков и «интуитивной очевидности». Однако впоследствии методы математической логики нашли применение и в других областях науки. Методы матлогики используются в информатике, лингвистике, искусственном интеллекте, экспертных системах, физике и т. д.

Вопрос о существовании алгоритмов имеет для математики первостепенное значение. В последние годы большое внимание уделяется теории сложности алгоритмов и вычислений. Выяснилось, что одного только существования алгоритма, решающего ту или иную массовую проблему, далеко не достаточно для практики. После уточнения понятия сложности вычисления стали исследовать вопросы такого рода, как внутренняя сложность вычислимой функции, ее криптографическая стойкость, приобретающие особую актуальность с развитием сетей связи, вычислительной техники и автоматизированных систем управления.

Автор надеется, что данное пособие поможет читателям в освоении теоретических положений дисциплины и приобретении практических навыков решения задач.

Литература

1. Авдошин С. М., Ахметсафина Р. З., Максименкова О. В. Информатика. Логика и алгоритмы. Эффективные методы решения задач. – СПб. : Просвещение, 2013. – 174 с.
2. Агарева О. Ю., Селиванов Ю. В. Элементы математической логики : учеб. пособие. – М. : МАТИ, 2008. – 52 с.
3. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. – М. : Мир, 1978. – Т. 1. – 612 с.
4. Гусев Д. А. Удивительная логика. – М. : ЭНАС-Книга, 2013. – 240 с.
5. Зыков А. Г., Поляков В. И., Скорубский В. И. Математическая логика. – СПб. : НИУ ИТМО, 2013. – 131 с.
6. Иванов Б. Н. Дискретная математика. Алгоритмы и программы. Полный курс. – М. : ФИЗМАТЛИТ, 2007. – 408 с.
7. Игошин В. И. Математическая логика и теория алгоритмов : учеб. пособие для вузов. – М. : Академия, 2004. – 446 с.
8. Кацаран Т. К., Строева Л. Н. Машина Тьюринга и рекурсивные функции : учеб. пособие для вузов. – Воронеж : Изд-во ВГУ, 2008. – 36 с.
9. Кнут Д. Искусство программирования. Т. 1 : Основные алгоритмы. – 3-е изд. – М. : Вильямс, 2006. – 720 с.
10. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ. – 4-е изд. – М. : Вильямс, 2011 – 1296 с.
11. Мендельсон Э. Введение в математическую логику. – М. : Наука, 1976. – 320 с.
12. Поляков В. И., Скорубский В. И. Основы теории алгоритмов. – СПб. : СПб НИУ ИТМО, 2012. – 51 с.
13. Рузавин Г. И. Логика и аргументация. – М. : ЮНИТИ, 1997. – 351 с.
14. Смыслова З. А., Пермякова Н. В. Спецглавы математики : учеб. пособие. – Томск : ТМЦДО, 2001. – Ч. 3. – 80 с.
15. Лихтарников Л. М., Сукачева Т. Г. Математическая логика : курс лекций : задачник-практикум и решения. – СПб. : Лань, 1999. – 288 с.
16. Шапорев С. Д. Математическая логика : курс лекций и практических занятий : учеб. пособие для вузов. – СПб. : БХВ-Петербург, 2005. – 410 с.

Глоссарий

Аксиома – основное положение рассматриваемой теории, принимаемое без доказательств.

Аксиоматическая система – совокупность основных и производных понятий, аксиом и теорем.

Алгебра высказываний – раздел математической логики, занимающийся построением и преобразованием высказываний с помощью логических операций, а также изучающий свойства и отношения между ними.

Алгоритм – это общий, единообразный, точно установленный способ решения любой задачи из данной массовой проблемы.

Выполнимые формулы – формулы, принимающие значение «истина» хотя бы на одном наборе логических переменных.

Высказывание – повествовательное предложение, для которого имеет смысл говорить о его истинности или ложности.

Дизъюнктивная нормальная форма (*ДНФ*) – форма записи алгебры высказываний, представленная в виде дизъюнкции элементарных конъюнкций.

Дизъюнкция двух высказываний A и B – высказывание $A \vee B$, ложное в том и только в том случае, когда оба высказывания A и B ложны.

Импликация двух высказываний A и B – высказывание $A \to B$, ложное тогда и только тогда, когда A истинно, а B – ложно.

Квантор – это общее название для логических операций, ограничивающих область истинности какого-либо предиката.

Конъюнктивная нормальная форма (*КНФ*) – форма записи алгебры высказываний, представленная в виде конъюнкции элементарных дизъюнкций.

Конъюнкция двух высказываний A и B – высказывание $A \wedge B$, истинное тогда и только тогда, когда истинны оба высказывания A и B.

Логика предикатов – это расширение возможностей логики высказываний, позволяющее строить высказывания с учетом свойств изучаемых объектов или отношений между ними.

Логическая переменная – это переменная, обозначающая любое высказывание, которая может принимать логические значения «истина» или «ложь».

Математическая логика – естественнонаучная дисциплина, изучающая математические доказательства и вопросы оснований математики.

Область истинности предиката $P(x)$, заданного на множестве M – совокупность всех x из *M*, при которых данный предикат обращается в истинное высказывание.

Отрицание (инверсия) высказывания *A* – высказывание $\neg A$, истинное тогда и только тогда, когда *A* ложно.

Предметная область (область определения) предиката – это множество *M*, на котором определен предикат.

Предметная переменная – это элементы множества *M*, на котором определен предикат.

Равносильные формулы – формулы, принимающие одинаковые истинностные значения при любых наборах своих логических переменных.

Сложное высказывание – высказывание, образованное из элементарных высказываний с помощью грамматических связок.

Совершенная ДНФ (СДНФ) – ДНФ, в которой нет одинаковых элементарных конъюнкций и все конъюнкции состоят из одного и того же набора переменных, в который каждая переменная входит только один раз (возможно с отрицанием).

Совершенная КНФ (СКНФ) – КНФ, в которой нет одинаковых элементарных дизъюнкций и все дизъюнкции состоят из одного и того же набора переменных, в который каждая переменная входит только один раз (возможно с отрицанием).

Тождественно истинные формулы (тавтологии) – формулы, принимающие значение «истина» на всех наборах логических переменных.

Тождественно ложные формулы (противоречия) – формулы, принимающие значение «ложь» на всех наборах логических переменных.

Эквиваленция двух высказываний *A* и *B* – высказывание $A \leftrightarrow B$, истинное тогда и только тогда, когда истинностные значения *A* и *B* одинаковы.

Элементарное высказывание – высказывание, представляющее собой одно утверждение.

Элементарная дизъюнкция (ЭД) – дизъюнкция k логических переменных или их отрицаний $(k \geq 1)$.

Элементарная конъюнкция (ЭК) – конъюнкция k логических переменных или их отрицаний $(k \geq 1)$.