**Московский государственный технический университет
имени Н.Э. Баумана**

Факультет «Фундаментальные науки»
Кафедра «Математическое моделирование»

А.Н. Канатников

ДИСКРЕТНАЯ МАТЕМАТИКА

Конспект лекций

Для студентов специальности
«Прикладная математика»

Москва
2006

1. БУЛЕВЫ ФУНКЦИИ

1.1. Булевы алгебры

Булева алгебра: симметричное полукольцо с унарной операцией дополнения $x \to \overline{x}$ с аксиомами $x + \overline{x} = 1$, $x\,\overline{x} = 0$. Единственность дополнения. Булевы алгебры и булевы кольца. Булева алгебра как решетка. Решетка — алгебра $(L, \vee, \wedge)$ в которой операции (нешеточные объединение и пересечение) ассоциативны, коммутативны, идемпотентны и связаны тождествами поглощения $a \vee (a \wedge b) = a$, $a \wedge (a \vee b) = a$. Всякое симметричное полукольцо является решеткой, а дистрибутивная решетка (каждая операция дистрибутивна относительно другой) с нулем и единицей — симметричным полукольцом. Пример булевой алгебры — булев куб $\mathbb{B}^n$. Теорема: любая конечная булева алгебра изоморфна булеву кубу некоторой размерности.

Напомним, что по определению булева алгебра — это симметричное полукольцо, в котором введена еще одна унарная операция — дополнение. Дополнение элемента x, обозначаемое $\overline{x}$, удовлетворяет аксиомам $x + \overline{x} = 1$, $x\,\overline{x} = 0$. Отметим, что эти два равенства для любого x определяют элемент $\overline{x}$ однозначно, т.е. существование дополнения определяется свойствами двух бинарных операций.

Подробнее можно сказать, что булева алгебра — универсальная алгебра вида $(L,\ \{+,\ \cdot,\ ^-\})$, удовлетворяющая условиям:

а) каждая бинарная операция коммутативна, ассоциативна, идемпотентна и имеет нейтральный элемент;

б) каждая бинарная операция дистрибутивна относительно другой;

в) нейтральный элемент каждой бинарной операции является аннулирующим другой бинарной операции;

г) дополнение связано с бинарными операциями аксиомами $x + \overline{x} = 1$, $x\,\overline{x} = 0$.

Обе операции порождают два отношения порядка, являющиеся взаимно обратными. Так, отношение $x \leqslant y$, равносильное равенству $x + y = y$, также определяется равенством $xy = x$.

Если в булевой алгебре B ввести операцию $x \oplus y = x\overline{y} + \overline{x}y$, то получим алгебраическую систему $(L,\ \{\oplus,\ \cdot\})$, которая является кольцом с единицей, в котором операция умножения идемпотентна (булевым кольцом). Короче говоря, булева алгебра является булевым кольцом. Наоборот, в любом булевом кольце $(L,\ \{\oplus,\ \cdot\})$ умножение коммутативно, а сложение удовлетворяет условию $x \oplus x = 0$. Положив $x + y = x \oplus y \oplus xy$, $\overline{x} = x + 1$, получим алгебраическую систему $(L,\ \{+,\ \cdot,\ ^-\})$, представляющую собой булеву алгебру.

К понятию булевой алгебры можно подойти и с другой стороны. Коммутативная идемпотентная полугруппа называется ***полурешеткой***. В полурешетке (как и в полукольце) можно ввести отношение естественного порядка согласно правилу $x \leqslant y \Leftrightarrow x + y = y$. Наоборот, любое упорядоченное множество, в котором любое двухэлементное множество имеет точную верхнюю грань, можно интерпретировать как полурешетку с операцией $x + y = \sup\{x,\ y\}$. Если в упорядоченном множестве каждое двухэлементное множество имеет и точную верхнюю, и точную нижнюю грани, то на это множестве можно ввести структуру полурешетки двумя способами: с операцией $\sup\{x,\ y\}$ и с операцией $\inf\{x,\ y\}$. Первую структуру называют верхней полурешеткой, а вторую — нижней. Две решеточные операции связаны тождествами

$$\sup\{x,\ \inf\{x, y\}\} = x, \quad \inf\{x,\ \sup\{x, y\}\} = x.$$

Алгебра $(R,\ \{\vee,\ \wedge\})$, для которой $(R,\ \vee)$ и $(R,\ \wedge)$ есть полурешетки, причем операции связаны тождествами поглощения $a \vee (a \wedge b) = a$, $a \wedge (a \vee b) = a$, называется решеткой. Бинарные операции называются решеточными (решеточное объединение и решеточное пересечение). Решеточные операции входят в структуру решетки симметрично. Поэтому любую

из них можно назвать объединением, тогда вторая будет пересечением. Структуру решетки имеет любое упорядоченное множество, в котором для любого двухэлементного множества существует точная верхняя и точная нижняя грани. В этом случае решеточными операциями будут $a \vee b = \sup\{a,\ b\}$ и $a \wedge b = \inf\{a,\ b\}$.

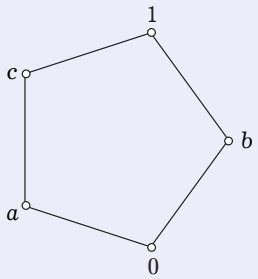


Рис. 1.1

Структуру решетки имеет любое симметричное полукольцо. Решетка в общем случае может и не быть симметричным полукольцом. Например, ***пентагон*** — упорядоченное множество $\{0,\ a,\ b,\ c,\ 1\}$ с отношением порядка, заданным диаграммой Хассе* на рис. 1.1, — является решеткой, но в этой решетке операции не являются дистрибутивными друг относительно друга, как должно быть в симметричном полукольце. Однако всякая дистрибутивная решетка (т.е. решетка, в которой каждая операция дистрибутивна относительно другой) с нулем и единицей (нейтральными элементами объединения и пересечения) является симметричным полукольцом.

Простейший пример булевой алгебры — *двухэлементная булева алгебра*

$$\mathbb{B} = (\{0,\ 1\},\ \{+,\ \cdot,\ ^-\})$$

с операциями $x+y = \max\{x,\ y\}$, $xy = \min\{x,\ y\}$, $\overline{x} = x+1$. Обобщением этого примера является булева алгебра $\mathbb{B}^n$ с покомпонентным выполнением операций алгебры $\mathbb{B}$ на кортежах. Алгебру $\mathbb{B}^n$ называют n-й декартовой степенью $\mathbb{B}$ или ***n-мерным булевым кубом***. Оказывается, что любая конечная булева алгебра изоморфна некоторому булеву кубу. Попросту говоря, других примеров конечных алгебр, отличных от булева куба, нет.

Другой важный пример булевой алгебры — алгебра $\mathcal{S}_A = \left(2^A,\ \{\cup,\ \cap,\ ^-\}\right)$, т.е. множество всех подмножеств множества A с теоретико-множественными операциями объединения, пересечения и дополнения. Отметим, что все практически значимые примеры булевых алгебр можно интерпретировать в рамках алгебры $\mathcal{S}_A$ или какой-либо ее подалгебры. Например, элементы булевой алгебры $\mathbb{B}^n$ (n-мерные булевы векторы) можно рассматривать как запись характеристической функции подмножества в множестве из n элементов. А тогда операции в $\mathbb{B}^n$ будут соответствовать теоретико-множественным операциям. Симметричное полукольцо $([a,\ b],\ \{\max,\ \min\})$ можно преобразовать в алгебру множеств, поставив в соответствие каждому $x \in [a,\ b]$ отрезок $[a,\ x]$. Множество таких отрезков замкнуто относительно операций объединения и пересечения, т.е. образует подалгебру в $2^{[a,b]}$. Правда это полукольцо не является булевой алгеброй, так как в нем нельзя ввести операцию дополнения.

1.2. Булевы функции

Булева функция — отображение $f\colon B^n \to B$, где B — некоторая булева алгебра. Наибольший интерес — двухэлементная алгебра $\mathbb{B}$. Табличный способ задания булевой функции. Примеры – функции одной и двух переменных. Вектор значений булевой функции. Аналитический способ. Язык булевой алгебры: тройка (P, C, F) (переменные, константы, функциональные символы). Интерпретация языка: сопоставление каждому функциональному символу арности n конкретной булевой функции от n аргументов. Синтаксическое дерево. Подформулы и значение формулы. Неоднозначность соответствия формул и функций. Эквивалентность функций и эквивалентность формул. Теорема о замене подформулы эквивалентной и теорема о замене переменной в эквивалентных формулах. Замыкание множества булевых функций. Замкнутые и полные множества функций.

Пусть задана булева алгебра $(B,\ \{\vee,\ \wedge,\ ^-\})$. Булева функция — это отображение $f\colon B^n \to B$, т.е. функция от n переменных, область изменения каждой из которых есть сама алгебра, причем значениями функции также являются элементы булевой алгебры. Мы остановимся на

*Диаграмма Хассе — изображение в виде неориентированного графа конечного упорядоченного множества. При этом ребрами соединяются вершины, связанные отношением доминирования, причем вершина, изображенная ниже, предшествует расположенной выше.

простейшем случае, имеющем наибольший интерес, когда B — это двухэлементная алгебра $\mathbb{B}$. В этом случае каждый аргумент функции принимает два возможных значения 0 и 1 и сама функция принимает лишь два тех же значения. Подобная функция представляет собой отображение $f\colon \mathbb{B}^n \to \mathbb{B}$. Комбинируя такие функции, можно получить любое отображение вида $F\colon \mathbb{B}^n \to \mathbb{B}^m$, а в конечном счете булевы функции нескольких переменных для любой конечной булевой алгебры.

Нетрудно подсчитать количество возможных булевых функций вида $f\colon \mathbb{B}^n \to \mathbb{B}$: это количество равно 2^{2^n} (по формуле Y^X, где X количество элементов области определения, а Y — количество элементов области значений).

Таблица 1.1

$x_1, x_2, \ldots, x_n$	$f(x_1, x_2, \ldots, x_n)$
$0, 0, \ldots, 0$	$f(0, 0, \ldots, 0)$
. . . . .	. . . . .
$\alpha_{k1}, \alpha_{k2}, \ldots, \alpha_{kn}$	$f(\alpha_{k1}, \alpha_{k2}, \ldots, \alpha_{kn})$
. . . . .	. . . . .
$1, 1, \ldots, 1$	$f(1, 1, \ldots, 1)$

Поскольку и область определения, и область значений конечны, булевы функции удобно задавать с помощью таблиц. Элементы области определения — n-мерные векторы, которые можно рассматривать как двоичные записи целых неотрицательных чисел от 0 до $2^n - 1$. Таблица, определяющая булеву функцию, может выглядеть следующим образом (табл. 1.1).

В качестве примера приведем таблицы булевых функций одного переменного (табл. 1.2) и семи наиболее важных функций двух переменных (табл. 1.3).

Таблица 1.2

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	0	0	1	1
1	1	0	1	0

Таблица 1.3

$x_1,\ x_2$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$	$f_7(x)$
00	0	0	0	1	1	1	1
01	1	0	1	1	0	1	0
10	1	0	1	0	0	1	0
11	1	1	0	1	1	0	0

Отметим, что булева функция n переменных представляет собой n-арную операцию на булевой алгебре $\mathbb{B}$. В частности, функции двух переменных — это бинарные операции, для которых обычно используют инфиксную форму записи. Так, для функций приведенных в табл. 1.3, используют следующие названия и символы операций:

$f_1(x_1, x_2) = x_1 \vee x_2 = x_1 + x_2$ — дизъюнкция;
$f_2(x_1, x_2) = x_1 \wedge x_2 = x_1 \cdot x_2$ — конъюнкция;
$f_3(x_1, x_2) = x_1 \oplus x_2$ — сложение по модулю 2;
$f_4(x_1, x_2) = x_1 \to x_2$ — импликация;
$f_5(x_1, x_2) = x_1 \sim x_2$ — эквивалентность;
$f_6(x_1, x_2) = x_1 \mid x_2$ — штрих Шеффера („не и“);
$f_7(x_1, x_2) = x_1 \downarrow x_2$ — стрелка Пирсона („не или“).

Функции f_1 и f_2 — базовые операции булевой алгебры $\mathbb{B}$, т.е. сложение и умножение (или, по-другому, решеточное объединение и пересечение), но в теории булевых функции традиционно предпочитают названия, указывающие на связь с математической логикой. Сложение по модулю 2 — альтернатива дизъюнкции, превращающая полукольцо $\mathbb{B}$ в поле $\mathbb{Z}_2$. Штрих Шеффера и стрелка Пирсона — это отрицания соответственно конъюнкции и дизъюнкции, т.е.

$$x_1 \mid x_2 = \overline{x_1 \wedge x_2}, \qquad x_1 \downarrow x_2 = \overline{x_1 \vee x_2}.$$

Инфиксная форма записи и представление штриха Шеффера и стрелки Пирсона с помощью операций булевой алгебры указывает еще на один способ представления булевых функций — аналитический. Вообще говоря, любая алгебраическая система содержательна, если для нее удается создать такие формы записи операций и их последовательностей, для которых возникают многообразные способы преобразования выражений. Речь идет о понятии „формула“.

Формула интуитивно понимается как запись с помощью определенного языка некоторой последовательности действий над объектами алгебраической системы.

В самом общем смысле под языком понимают следующее. Рассмотрим некоторое непустое конечное множество V, элементы которого будем называть ***символами***. Произвольные последовательности символов называются ***словами*** или ***цепочками***. Слова могут быть любой длины. Вводится специальное слово нулевой длины. Множество всех слов обозначим V^*. Это множество можно представить как объединение всех декартовых степеней множества V (нулевой, содержащей слово нулевой длины, первой со словами длины 1, второй со словами длины 2 и т.д.). ***Язык*** — это некоторое подмножество в V^*, т.е. некоторый фиксированный набор слов, составленных с помощью элементов множества V, которое называют ***алфавитом языка***.

Конкретный язык формируется правилами, по которым из всех слов, составленных с помощью данного алфавита, выбираются „правильные“ или допустимые слова. В нашем случае словами должны быть правильно записанные формулы булевой алгебры, а символами — символы переменных, знаков операций и элементов алгебры. Язык формул можно построить в рамках данного выше понятия формального языка. Однако, упрощая изложение, мы снимем ограничение конечности алфавита языка*, полагая, что алфавит может быть счетным множеством. Строго формальный математический язык можно ввести следующим образом.

Пусть дана тройка объектов $(P,\ C,\ F)$, где P — счетное множество, элементы которого мы будем называть ***булевыми переменными***; C — счетное множество символов, используемых для обозначения элементов алгебраической системы (в данном случае двух элементов 0 и 1 алгебры $\mathbb{B}$), которые мы будем называть ***константами***; F — счетное множество ***функциональных символов***, каждый из которых имеет натуральную характеристику — ***арность***.

Понятие формулы в рассматриваемом языке вводится индуктивно, т.е. указываются конкретные способы построения правильных формул на основе уже построенных формул. Базис индукции — изначальный набор ***элементарных формул***, под которыми мы будем понимать элементы множества $P \cup C$. Шаг индукции описывает, как из уже построенных формул получаются новые: если $X_1,\ X_2,\ \ldots,\ X_n$ — формулы и $f \in F$ — функциональный символ арности n, то $f(X_1, X_2, \ldots, X_n)$ — формула. Кратко сказанное можно сформулировать так:

1) элементы множества $P \cup C$ — формулы;

2) если $X_1,\ X_2,\ \ldots,\ X_n$ — формулы и $f \in F$ — функциональный символ арности n, то $f(X_1, X_2, \ldots, X_n)$ — формула

Первый пункт определения — базис индукции. Второй (и последующие, если они есть) определяет правила построения новых формул из уже построенных. Это шаг индукции.

В данном случае под алфавитом следует понимать объединение $P \cup C \cup F$, к которому добавлены еще три служебных символа: две круглые скобки и запятая. Но заметим, что при таком подходе нарушается требование конечности алфавита. Мы сознательно идем на такое нарушение, чтобы не усложнять язык формул. ***Язык булевой алгебры*** — это множество всех правильно составленных формул.

Хотя мы выделили константы, т.е. символы, обозначающие объекты рассматриваемой алгебраической системы, в самостоятельный класс, часто удобно рассматривать их как специальный вид функциональных символов арности 0. Таким символам соответствуют постоянные булевы функции.

Индуктивное определение формулы позволяет корректно составлять сами формулы, как последовательности символов, но не придает им какого-либо смысла. Чтобы наполнить формулы смыслом, необходимо каждому функциональному символу сопоставить конкретную операцию

*Ситуацию здесь можно пояснить следующим образом. При записи математических формул используют числовые константы, которые фигурируют в десятичной форме записи. Такая запись включает десять цифр знак числа и десятичную запятую. правильных записей чисел существует счетное число. Но рассматривая сложные конструкции формул, можно каждую такую запись считать самостоятельным символом, не раскрывая способ формирования записи.

соответствующей арности (конкретную булеву функции с соответствующим количеством аргументов). Другими словами, необходимо задать отображение множества F в множество всех булевых функций, при котором арность функционального символа совпадает с арностью операции. Кроме того, необходимо фиксировать смысл символов, предназначенных для обозначения констант, т.е. задать отображение множества C в алгебру $\mathbb{B}$. Два указанных отображения составляют ***интерпретацию языка***.

Следует учесть еще одно обстоятельство. Общепринятым является запись бинарных операций не в виде $f(x_1, x_2)$, а в виде $x_1\ f\ x_2$ (как правило, со специальным символом операции). Поэтому в наш язык следует ввести символы инфиксных операций, а в качестве шага индукции ввести дополнительное правило:

3) если X_1 и X_2 — формулы, то и слова $(X_1 + X_2)$, $(X_1 \cdot X_2)$, $(X_1 \oplus X_2)$, $(X_1 \to X_2)$, $(X_1 \sim X_2)$, $(X_1 \mid X_2)$, $(X_1 \downarrow X_2)$ являются формулами.

При использовании инфиксной формы записи бинарных операций в формулах оказывается большое количество пар скобок, что затрудняет восприятие таких формул. Скобки нужны для того, чтобы однозначно определить последовательность шагов, которые приводят от элементарных формул к конечной формуле. Чтобы сократить количество скобок и сделать формулы более читаемыми, используют правило приоритетов, согласно которому каждой бинарной операции с инфиксной формой записи присваивается приоритет — некоторое натуральное число. Сперва выполняются операции с большим приоритетом, затем с меньшим. Среди операций с одинаковым приоритетом в первую очередь выполняется та, знак которой находится левее. Мы будем использовать инфиксную форму на практике, как более привычную и удобную, но в теоретических рассуждениях с целью упрощения ограничимся лишь префиксной формой записи операций.

Рассматривая различные множества F функциональных символов и различные их интерпретации, мы можем строить различные схемы построения формул, которые различаются набором базовых функций. Например, в качестве базовых могут рассматриваться три базовые операции булевой алгебры: сложение, умножение и отрицание, но могут быть и другие варианты (например, сложение по модулю 2 и умножение — базовые операции в $\mathbb{Z}_2$).

Любая корректная формула либо элементарная, либо получена в результате шага индукции, т.е. с помощью функционального символа f и некоторого набора формул $X_1, \ldots, X_n$. Каждая из формул X_i в свою очередь либо элементарная формула, либо получена с помощью своего функционального символа f_i и некоторого набора формул $X_{I1}, \ldots, X_{in_i}$ и т.д. Двигаясь от конечной формулы, мы за конечное число шагов придем к элементарным формулам — переменным и константам. Представленный анализ формулы называется синтаксическим. Его результаты можно представить в виде ориентированного дерева. Корень дерева — сам формула, вершины глубины 1, формулы X_i и т.д. Каждому листу дерева соответствует переменная или константа. Вершинам, не являющимся корнем соответствуют формулы, которые входят как составная часть в анализируемую формулу. Они называются ***подформулами***.

Пример 1.1. Рассмотрим формулу $f_1(f_2(x_1, x_2), f_3(x_2), f_4(x_1, f_2(x_3), x_4))$. Эта формула получена с помощью функционального символа f_1 арности 3. Подформулами глубины 1 являются $f_2(x_2)$, $f_3(x_2)$ и $f_4(x_1, f_2(x_3), x_4)$. Первая из них имеет подформулу — переменную x_2, вторая имеет также одну подформулу — переменную x_2, а третья — уже три подформулы: первая и третья — переменные x_1 и x_4, вторая подформула — сложная подформула $f_2(x_3)$. В результате получаем дерево синтаксического анализа, представленное на рис. 1.2. Высоту дерева (в данном случае 3) называют ***сложностью формулы***.

Основная задача в связи с аналитическим способом задания булевых функций — описание класса функций, представимых формулами. В дальнейшем мы для упрощения изложения будем отождествлять множество функциональных символов языка с соответствующим множеством булевых функций. Можно считать, что каждой булевой функции соответствует свой уникальный символ и F состоит как раз из таких символов, так что, задав F, мы тем самым задаем и интерпретацию формул.

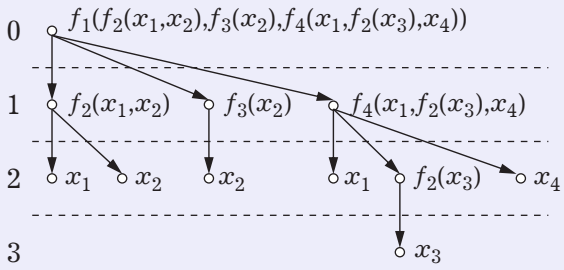


Рис. 1.2

Один из способов получения новых формул — замена одной из подформул другой. Такое преобразование называется ***подстановкой***. Замене может подлежать как сложная формула, так и любая элементарная.

В формулу, как правило, входят переменные (булевы переменные). Но есть формулы, в которых элементарные составляющие — все константы. Такие формулы имеют определенное значение, соответствующее конкретному объекту алгебраической системы (т.е. в нашем случае 0 и 1). Если в формулу входят переменные, то ее значение не определено. Однако если в формуле заменить каждую переменную константой (т.е. дать переменной конкретное значение), получим новую формулу, имеющую значение. Таким образом, каждому набору значений переменных, входящих в формулу, соответствует конкретное значение формулы.

Это обстоятельство позволяет рассматривать формулу как форму записи функции. Однако надо иметь в виду, что с помощью одной и той же формулы можно задавать различные функции. Например, формула $x-t$ определяет функции $f(x,t=x-t$ и $f(t,x)=x-t$, а также, например, функцию $f(x,y,t)=x-t$.

Из этого примера можно сделать вывод: чтобы с помощью формулы задать функцию, необходимо установить соответствие между аргументами функции и переменными, входящими в формулу. Каждой переменной входящей в формулу, должен соответствовать один аргумент функции, некоторые из аргументов могут быть ене связаны с какими-либо переменными. Если такое соответствие установить, то набор значений переменных, входящих в формулу, можно записать как булев вектор. При этом каждому булевому вектору будет соответствовать значение формулы, и мы получаем корректно заданную булеву функцию.

Рассматривая формулы как способ описания функций, естественно не различать те формулы, которые порождают одинаковые функции при фиксированном соответствии между переменными и аргументами функции.

На практике мы часто пользуемся эквивалентными преобразованиями. Например, если f_1 — булева функция из табл. 1.3, то $f_1(x_1,x_2)=f_1(x_2,x_1)$. Знак равенства означает, что формулы задают одну и ту же функцию, т.е. эквивалентны. Вообще говоря, знак равенства в данном контексте рассматривается как знак тождества, т.е. формулы дают одно и то же значение при любых значениях входящих в равенство переменных. Если состав переменных слева и справа один и тот же, то тождество как раз и означает, что левая и правая части задают одну и ту же функцию. Однако возможно и такое равенство $x\oplus x=0$, также верное при любых значениях неизвестных. Но в левую часть входит переменная x, а в правой части вообще стоит константа. Левая часть задает функцию двух переменных, а правая часть — нульарную операцию (константу). Надо либо отказаться признать такое равенство, либо установить эквивалентность каких-либо функций. Второе более удобно.

Две формулы* $\Phi[x_1,\ldots,x_n]$ и $\Psi[y_1,\ldots,y_m]$ с наборами переменных x_1, …, x_n и y_1, …, y_m соответственно назовем ***эквивалентными***, если равенство $\Phi[x_1,\ldots,x_n]=\Psi[y_1,\ldots,y_m]$ остается верным при любых значениях всех переменных. Если наборы переменных не совпадают, то значение одной из формул или обеих не зависит от значений некоторых переменных. Такие

*Использование квадратных скобок подчеркивает, что это не обозначение функции. Здесь имеется в виду формула, обозначенная как Φ, все переменные которой входят в указанный список переменных.

переменные назовем ***фиктивными***. Переменную, не являющуюся фиктивной, будем называть ***существенной***. У двух эквивалентных формул существенные переменные должны быть общими, хотя часть общих переменных может быть и фиктивными.

Следующая теорема отражает обычные правила преобразования формул, приводящие к эквивалентным формулам.

Теорема 1.1. Если формулы $\Phi[x_1, \ldots, x_n]$ и $\Psi[y_1, \ldots, y_m]$ эквивалентны, то замена в каждой из них всех вхождений какой-либо из переменных произвольной формулой приведет к двум новым формулам, эквивалентным между собой.

◄ Объединим списки переменных у формул и напишем равенство

$$\Phi[z_1, \ldots, z_k] = \Psi[z_1, \ldots, z_k].$$

Предположим, что все вхождения переменной z_k заменены формулой $\Delta[u_1, \ldots, u_l]$. Получим формулы $\widetilde{\Phi}[z_1, \ldots, z_{k-1}, u_1, \ldots, u_l]$ и $\widetilde{\Psi}[z_1, \ldots, u_1, \ldots, u_l]$. Обозначения исходят из того, что переменные формулы Δ не входят в формулы Φ и Ψ. На самом деле это допущение несущественно и служит лишь упрощению рассуждений.

Задав произвольные значения переменным z_1, …, z_{k-1}, u_1, …, u_l, мы получим для переменной z_k значение $\Delta(u_1, \ldots, u_l)$. Проверка равенства значений двух формул равносильна проверке исходного равенства, в котором в качестве значения переменной z_k взято $\Delta(u_1, \ldots, u_l)$. Поскольку исходное равенство верно, то и после подстановки равенство останется верным. ►

Теорема 1.2. Если в формуле заменить одну из подформул эквивалентной, то новая формула будет эквивалентна исходной.

◄ Пусть дана формула $\Phi[x_1, \ldots, x_n]$, в которую входит подформула $\Psi[x_1, \ldots, x_n]$ (мы можем считать, что подформула включает все переменные исходной формулы, рассматривая недостающие как фиктивные). Заменим подформулу Ψ новой переменной z, которая не входит в список x_1, …, x_n. Получим новую формулу $\Gamma[x_1, \ldots, x_n, z]$, связь которой с исходной формулой можно записать в виде

$$\Phi[x_1, \ldots, x_n] = \Gamma[x_1, \ldots, x_n, z]\Big|_{z=\Psi[x_1,\ldots,x_n]}.$$

Замена формулы $\Psi[x_1, \ldots, x_n]$ эквивалентной формулой $\widetilde{\Psi}[x_1, \ldots, x_n]$ приведет к новой формуле

$$\Gamma[x_1, \ldots, x_n, z]\Big|_{z=\widetilde{\Psi}[x_1,\ldots,x_n]}.$$

Надо показать, что при любых значениях неизвестных верно равенство

$$\Gamma[x_1, \ldots, x_n, z]\Big|_{z=\Psi[x_1,\ldots,x_n]} = \Gamma[x_1, \ldots, x_n, z]\Big|_{z=\widetilde{\Psi}[x_1,\ldots,x_n]}.$$

Но нетрудно заметить, что, задавая значения неизвестных, мы в силу эквивалентности формул Ψ и $\widetilde{\Psi}$ получим один и тот же набор значений переменных в формуле Γ. Значит, в обеих частях равенства мы получим одно и то же значение. ►

В функции $f(x_1, \ldots, x_n)$ назовем i-й аргумент ***фиктивным аргументом***, если значение функции не зависит от значения этого аргумента, т.е. функции

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \quad \text{и} \quad f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

совпадают. Вместо двух этих равных друг другу функций можно рассмотреть функцию

$$\tilde{f}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n),$$

у которой количество аргументов на единицу меньше. Такое преобразование назовем ***удалением фиктивного аргумента***.

Возможно и противоположное преобразование — ***введение фиктивных переменных***, а именно:

$$\hat{g}(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) = g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n).$$

Две функции назовем ***эквивалентными***, если после удаления в каждой из них всех фиктивных аргументов, мы получим равные функции.

Нетрудно проверить, что введенное отношение на множестве булевых функций действительно является отношением эквивалентности. Точно так же на множестве все булевых формул эквивалентность (в смысле равенства значений при любых значениях переменных) — это отношение эквивалентности.

Введение этих отношений преследует единственную цель: обозначить степень неоднозначности, с которой функции записываются формулами. Можно утверждать, что при фиксированном порядке переменных каждому классу эквивалентных формул соответствует ровно один класс эквивалентных функций. Вопрос: является ли это соответствие биекцией? Можно ли утверждать, что каждый класс эквивалентных функций определяется некоторым классом эквивалентных формул? Ответ зависит от выбора набора базовых функций, или ***базиса***, т.е. множества F.

Пусть F — некоторое множество функций. Его ***замыканием*** назовем множество $[F]$ всех булевых функций, которые представимы формулами с базисом F (***формулами над множеством F***). Если замыкание множества F совпадает с F, то это множество называется ***замкнутым***. Множество F называется ***полным***, если его замыкание совпадает с множеством всех булевых функций.

Термин „полный базис“ как раз и означает, что любую булеву функцию можно записать аналитически, используя в качестве исходных функции базиса. Замыкание множества — совокупность всех функций, которые могут быть представлены формулами над этим множеств. Замкнутое множество F — такое множество, которое не может быть расширено добавлением функций, представимых формулами над F.

Замыкание может рассматриваться как операция на совокупности всевозможных множеств булевых функций. Такая операция обладает следующими свойствами:

1) $[\varnothing] = \varnothing$;
2) $[[X]] = [X]$;
3) $X \subset [X]$;
4) $[X] \cup [Y] \subset [X \cup Y]$.

Первое свойство носит формальный характер. Второе вытекает из конечности процедуры построения любой формулы: достаточно, следуя по дереву синтаксического анализа, последовательно заменять функции из $[X]$ их формулами над X. Третье и четвертое свойства очевидны.

Из четвертого свойства вытекает, что если $X \subset Y$, то $[X] \subset [Y]$. Действительно, включение $X \subset Y$ равносильно равенству $X \cup Y = Y$. Если $X \subset Y$, то в силу свойства 4 заключаем, что $[X] \cup [Y] \subset [X \subset Y] = [Y]$. Следовательно, $[X] \subset [Y]$.

Из указанных свойств замыкания вытекает следующее утверждение.

Теорема 1.3. Если F — полное множество булевых функций, каждая из которых представима формулой над множеством G, то и G — полное множество.

◀ Так как каждая функция из F есть формула над G, то $F \subset [G]$. Из свойств замыкания вытекает, что

$$[F] \subset [[G]] = [G].$$

Но поскольку $[F]$ совпадает с множеством всех булевых функций, то и $[G]$ совпадает с множеством всех булевых функций, т.е. G — полный базис. ▶

1.3. ДНФ и КНФ

Стандартный базис. Элементарные формулы — литералы. Элементарная конъюнкция (дизъюнкция). Дизъюнктивная (конъюнктивная) нормальная форма и совершенная форма. Теорема: любая булева функция, отличная от 0 (от 1) представима в виде СДНФ (СКНФ). Полнота стандартного базиса. Примеры полных базисов: базис Жегалкина, штрих Шеффера, стрелка Пирса.

Стандартный базис — это набор из трех исходных операций булевой алгебры: сложения (объединения), умножения (пересечения) и отрицания.

Здесь мы будем называть ***литералом*** переменную x или ее отрицание $\overline{x}$ и обозначать $\hat{x}$. Булево пересечение нескольких литералов, определяемых различными переменными, т.е. выражение вида $X = \hat{x}_1\hat{x}_2\dots\hat{x}_k$, называется ***элементарной конъюнкцией***. Требование, чтобы все переменные были различны обусловливается следующим. Если в конъюнкцию входит несколько одинаковых литералов, то в силу коммутативности, ассоциативности и идемпотентности конъюнкции можно, переходя к эквивалентной формуле, оставить лишь один литерал (например, $\overline{x}_1\overline{x}_1 = \overline{x}_1$). Если в конъюнкцию входит переменная и ее отрицание, то формула эквивалентна константе 0, поскольку $x\,\overline{x} = 0$ и для любой формулы Y имеем $Y\,x\,\overline{x} = 0$.

Дизъюнкция нескольких элементарных конъюнкций называется ***дизъюнктивной нормальной формой***, или ***ДНФ***. Например,

$$x_1\,x_3 + \overline{x}_2\,x_3\overline{x}_4 + \overline{x}_1\,x_2\,\overline{x}_3\,x_5.$$

Если состав переменных в каждой элементарной конъюнкции данной ДНФ один и тот же, то ДНФ называется ***совершенной***. Приведенный пример — это ДНФ, не являющаяся совершенной. Напротив, формула

$$x_1\,x_2\,x_3\,\overline{x}_4 + \overline{x}_1\,\overline{x}_2\,x_3\overline{x}_4 + \overline{x}_1\,x_2\,\overline{x}_3\,x_4$$

есть совершенная форма.

Поскольку в булевой алгебре сложение и умножение — симметричные операции и всегда можно интерпретировать сложение как умножение, а умножение как сложение, существует и двойственное понятие — ***конъюнктивная нормальная форма*** (***КНФ***), представляющая собой конъюнкцию элементарных дизъюнкций, и ***совершенная конъюнктивная форма*** (***СКНФ***). Из принципа двойственности для симметричных полуколец вытекает, что любому утверждению относительно ДНФ отвечает двойственное утверждение относительно КНФ, которое получается заменой сложения (дизъюнкции) умножением, умножения (конъюнкции) сложением, константы 0 константой 1, константы 1 константой 0, отношения порядка двойственным (обратным) порядком. Поэтому далее мы остановимся на изучении только ДНФ.

Теорема 1.4. Любая булева функция, отличная от константы 0 представима в виде СДНФ.

◀ Условимся под x^σ понимать формулу x, если $\sigma = 1$, и формулу $\overline{x}$, если $\sigma = 0$. Пусть функция $f(y_1, \dots, y_n)$ принимает значение 1 на векторе $(t_1, \dots, t_n)$ (такой вектор называют ***конституэнтой единицы***). Тогда элементарная конъюнкция $x_1^{t_1}x_2^{t_2}\dots x_n^{t_n}$ также принимает значение 1 на этом наборе, но обращается в нуль на всех остальных n-мерных булевых векторах. Рассмотрим формулу

$$\Phi[x_1, x_2, \dots, x_n] = \sum_{f(t_1,\dots,t_n)=1} x_1^{t_1}x_2^{t_2}\dots x_n^{t_n},$$

в которой сумма (объединение) распространяется на все те наборы $(t_1, \dots, t_n)$ значений аргументов, на которых заданная функция принимает значение 1. Отметим, что множество таких наборов не пусто, так что в сумме есть по крайней мере одно слагаемое.

Нетрудно заметить, что формула Φ обращается в 1 при тех, и только при тех значениях переменных, при которых обращается в 1 рассматриваемая функция. Значит, формула Ψ представляет функцию f. ▶

Следствие 1.1. Стандартный базис является полным.

◀ Действительно, если функция не является константой 0, то она представима либо в виде СДНФ, которая является формулой над стандартным базисом. Константу 0 можно представить, например, формулой $f(x_1, x_2, \ldots, x_n) = x_1\overline{x}_1$. ▶

Пример 1.2. Рассмотрим функцию трех переменных $m(x_1, x_2, x_3)$ (табл. 1.4), называемую ***мажоритарной функцией***. Эта функция принимает значение 1, если больше половины ее аргументов имеют значение 1. Поэтому ее часто называют функцией голосования. Построим для нее СДНФ.

Таблица 1.4

x_1, x_2, x_3	$m(x)$
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

Мажоритарная функция имеет 4 конституэнты единицы, а значит, в ее СДНФ должно быть четыре слагаемых. Результат будет следующий:

$$m(x_1, x_2, x_3) = \overline{x}_1\, x_2\, x_3 + x_1\, \overline{x}_2\, x_3 + x_1\, x_2\, \overline{x}_3 + x_1\, x_2\, x_3$$

Аналогично строится СКНФ. Выбираем конституэнты нуля и для каждой составляем элементарную дизъюнкцию. Получим:

$$m(x_1, x_2, x_3) = (x_1 + x_2 + x_3)(x_1 + x_2 + \overline{x}_3)(x_1 + \overline{x}_2 + x_3)(\overline{x}_1 + x_2 + x_3). \ \#$$

Полнота стандартного базиса позволяет подбирать и другие полные системы функций. Полнота множества F может быть установлена из следующих соображений. Предположим, каждая из трех функций стандартного бузиса представима формулой над F. Тогда в силу теоремы 1.3 иножество F будет полным.

Пример 1.3. Множество из операций сложения по модулю 2, умножения и константы 1 называют ***базисом Жегалкина***. Сложение по модулю 2 и умножение — базовые операции кольца $\mathbb{Z}_2$, выражения, составленные с их помощью — это многочлены над кольцом $\mathbb{Z}_2$. Константа 1 в данном случае необходима для записи свободного члена. Поскольку $xx = x$, то все сомножители в многочлене имеют степень 1. Поэтому при записи многочлена можно обойтись без понятия степени. Примеры формул над базисом Жегалкина:

$$xy \oplus x \oplus y, \quad x \oplus 1, \quad xyz \oplus xz \oplus x \oplus y \oplus 1.$$

Любую такую формулу называют ***полиномом Жегалкина***. Фактически полином Жегалкина — это многочлен над кольцом $\mathbb{Z}_2$.

Нетрудно сконструировать формулы над базисом Жегалкина, представляющие операции сложения и отрицания стандартного базиса (умножение у двух базисов общее):

$$x + y = x \oplus y \oplus xy, \quad \overline{x} = x \oplus 1.$$

Поэтому базис Жегалкина — полное множество.

Можно показать, что для любой булевой функции полином Жегалкина определен однозначно (точнее, с точностью до порядка слагаемых). Коэффициенты полинома Жегалкина при небольшом количестве переменных можно найти методом неопределенных коэффициентов.

Пример 1.4. Рассмотрим множество из единственной функции — штриха Шеффера*. Это множество полно, что следует из следующих легко проверяемых тождеств:

$$\overline{x} = x \mid x, \quad xy = \overline{x \mid y} = (x \mid y) \mid (x \mid y), \quad x + y = \overline{x} \mid \overline{y} = (x \mid x) \mid (y \mid y).$$

Пример 1.5. Базис, состоящий из единственной функции — стрелки Пирса, также является полным. Проверка этого аналогична случаю штриха Шеффера. Впрочем, это заключение можно сделать и на основании принципа двойственности для симметричных полуколец.

*Штрих Шеффера — бинарная, но не ассоциативная операция. Поэтому при использовании инфиксной формы следует быть внимательным: результат зависит от порядка выполнения операций. В этом случае рекомендуется явно указывать порядок операций при помощи скобок, например писать $(x \mid y) \mid z$, а не $x \mid y \mid z$, хотя обе формы равнозначны.

1.4. Критерий Поста

Пять классов Поста T_0, T_1, S. M, L. Их замкнутость. Критерий Поста: множество полно, если оно не содержится ни в одном из классов Поста. Примеры.

Рассмотрим некоторые классы булевых функций:

• класс функций T_0, сохраняющих константу 0, т.е. функций, удовлетворящих условию $f(0,\ldots,0)=0$;

• аналогичный класс функций T_1, сохраняющих константу 1, т.е. удовлетворящих условию $f(1,\ldots,1)=1$;

• класс S самодвойственных функций, т.е. функций, удовлетворяющих условию $\forall\alpha\ \overline{f(\overline{\alpha})} = f(\alpha)$ (здесь α — n-мерный булев вектор, отрицание которого выполняется покомпонентно);

• класс M функций, монотонных относительно естественного порядка полукольца $\mathbb{B}^n$, т.е. функций, удовлетворяющих условию $\alpha \leqslant \beta \Rightarrow f(\alpha) \leqslant f(\beta)$;

• класс L линейных функций — функций, представимых полиномом Жегалкина первой степени (попросту говоря суммой по модулю 2 литералов, в число которых может входить константа 0).

Выделенные классы называют ***классами Поста***. Они интересны тем, что каждый определяется свойством, сохраняющимся при конструировании формул. Точнее говоря, если функции f, g_1, …, g_k принадлежат одному из указанных классов, то и их композиция $f(g_1, g_2, \ldots, g_k)$ принадлежит этому же классу. Это означает, что каждый из названных классов есть замкнутое множество. Отметим также, что классы Поста замкнуты относительно операций введения или удаления фиктивных аргументов.

Рассмотрение классов Поста приводит к следующему необходимому условию полноты рассматриваемого базиса: базис, целиком попадающий в один из классов Поста, не может быть полным. Действительно, если базис включен в один из классов Поста, то замыкание базиса также оказывается в этом классе. Но ни один из классов Поста не совпадает со всем множеством булевых функций. Следовательно, замыкание базиса не будет совпадать с множеством всех булевых функций. Менее очевидным является то, что это условие и достаточно.

Теорема 1.5 (критерий Поста). Множество F полно тогда и только тогда, когда оно не является подмножеством никакого из классов Поста.

◀ Доказательство достаточности критерия состоит в построении из заданных функций формул, представляющих функции стандартного базиса. Отметим, что в стандартном базисе одна из бинарных операций выражается через отрицание и другую бинарную операцию, например

$$x + y = \overline{\overline{x}\,\overline{y}}.$$

Поэтому достаточно построить формулы, определяющие отрицание и умножение.

Пусть F не является подмножеством ни одного из классов Поста. Для каждой функции $f \in F$ рассмотрим формулу $g(x) = f(x, x, \ldots, x)$. Поскольку F не содержится в T_0 и в T_1, в F, во-первых, есть непостоянные функции, а во-вторых есть хотя бы одна функция f_1, для которой $g_1(0) = 1$, и есть хотя бы одна функция f_2, для которой $g_2(1) = 0$. Это возможно, если одна из функций $g_1(x)$ и $g_2(x)$ есть отрицание, либо обе функции постоянны и представляют константы0 и 1. Рассмотрим оба случая.

Пусть функции $g_1(x)$ и $g_2(x)$ являются константы 0 и 1. Тогда для любой функции $f \in F$ формула $f(\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, x, \alpha_{i+1}, \ldots, \alpha_n)$, в которой α_j — константы, есть формула над F. Выберем функцию f, не являющуюся монотонной. Тогда существуют два булевых вектора p и q, удовлетворяющие условиям $p < q$ и $f(p) = 1$, $f(q) = 0$. Векторы p и q можно соединить цепочкой $p = p_0, p_1, \ldots, p_k = q$ непосредственно предшествующих друг другу векторов (соседних). В этой цепочке найдется два соседних вектора p_{j-1} и p_j, которые отличаются только одной компонентой с некоторым номером i и на которых $f(p_{j-1}) = 1$, $f(p_j) = 0$. Пусть α_j, $j \neq i$, одинаковые

компоненты этих векторов. Тогда формула $f(\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, x, \alpha_{i+1}, \ldots, \alpha_n)$ представляет собой операцию отрицания.

Предположим, например, что функция $g_1(x)$ является отрицанием. Тогда мы можем составлять формулы вида $f(x \oplus \sigma^1, \ldots, x \oplus \sigma_n)$, где $x \oplus \sigma$ есть переменная x при $\sigma = 0$ и ее отрицание при $\sigma = 1$. Выберем функцию $f \in F$, не являющуюся самодвойственной. Тогда можно указать такой булев вектор $p \in \mathbb{B}^n$, что $f(\overline{p}) \neq \overline{f(p)}$, откуда $f(\overline{p}) = f(p)$. Пусть σ_1, …, σ_n — компоненты вектора p. Рассмотрим функцию $g(x) = f(x \oplus \sigma^1, \ldots, x \oplus \sigma_n)$, определяемую выбранной несамодвойственной функцией f и вектором p. Так как $0 \oplus \sigma_i = \sigma_i$, $1 \oplus \sigma_i = \overline{\sigma}_i$, то $g(0) = f(p) = f(\overline{p}) = g(\overline{x})$. Следовательно, функция $g(x)$ является константой. Пусть, например, $g(x) = 1$. Тогда $\overline{g}(x) = 0$ — другая константа.

Итак, мы показали, что множество F, не являющееся подмножеством классов T_0, T_1, S, M, позволяет получить в виде формул отрицание и обе константы. Для построения формулы для произведения выберем функцию $f \in F$, не являющуюся линейной. Составляя формулы вида $f(X_1, X_2, \ldots, X_n)$, где X_i — это либо переменная x, либо переменная y, мы получим функции двух аргументов. Покажем, что среди таких функций есть нелинейные. В полиноме Жегалкина, представляющем функцию f, выберем нелинейное (степени два или выше) слагаемое наименьшей степени. Пусть это слагаемое имеет вид $x_{i_1} x_{i_2} \ldots x_{i_k}$. В формуле $f(X_1, X_2, \ldots, X_n)$ положим $X_{i_1} = x$, $X_{i_j} = y$, $j = \overline{2, k}$, а для остальных переменных, не вошедших в выбранное слагаемое выберем значение 0. Тогда выбранное слагаемое преобразуется в xy, остальные нелинейные слагаемые обнулятся, и мы получим функцию вида $g(x, y) = xy \oplus \alpha x \oplus \beta y \oplus \gamma$.

Так как

$$g(x, y) = xy \oplus \alpha x \oplus \beta y \oplus \gamma = (x \oplus \beta)(y \oplus \alpha) \oplus \alpha\beta \oplus \gamma = (x \oplus \beta)(y \oplus \alpha) \oplus \gamma',$$

заключаем, что $g(x \oplus \beta, y \oplus \alpha) \oplus \gamma' = xy$. Но $x \oplus \sigma$ — это переменная x при $\sigma = 0$ и ее отрицание $\overline{x}$ при $\sigma = 1$. Поэтому, если $g(x, y)$ принадлежит замыканию F, то и функция $g(x \oplus \beta, y \oplus \alpha) \oplus \gamma' = xy$, т.е. конъюнкция, принадлежит замыканию F.

Итак, мы показали, что если множество F не является подмножеством никакого класса Поста, то формулами над F можно представить отрицание и конъюнкцию, а значит, и дизъюнкцию. В этом случае, согласно теореме 1.3, множество F полное. ►

1.5. Минимизация ДНФ

Минимальная ДНФ — наименьшее число литералов (вхождений переменных). Кратчайшая ДНФ — наименьшее число элементарных конъюнкций. Геометрия булева куба. Импликанты и простые импликанты. Сокращенная ДНФ. Избыточные импликанты и тупиковые ДНФ. Методы построения сокращенной ДНФ. Алгоритм Квайна — Мак-Клоски. Склейка. Таблицы Квайна и простые импликанты. Ядровые импликанты. Тупиковые ДНФ и функция Патрика. Выбор кратчайших и минимальных ДНФ.

Каждая функция может быть представлена дизъюнктивной нормальной формой различными способами. например, изменить ДНФ можно, если в одно из ее слагаемых ввести фиктивную переменную с помощью сомножителя $x + \overline{x}$. Возникает задача из всех ДНФ найти наиболее простую в том или ином смысле.

Возможны по крайней мере два подхода к оценке сложности ДНФ: по количеству элементарных конъюнкций в ней (это количество называют ***длиной ДНФ***) и по общему количеству литералов, т.е. по сумме длин элементарных конъюнкций, входящих в ДНФ (эту сумму называют ***сложностью ДНФ***). ДНФ называют ***минимальной***, если она имеет наименьшую сложность, и ***кратчайшей***, если она имеет наименьшую длину.

В приведенной терминологии задача состоит в выборе среди всех ДНФ, представляющих данную функцию, наименьших и кратчайших. Решение такой задачи сводится к отбору некоторого ограниченного круга ДНФ, „подозрительных на минимум“, и последующему перебору отобранных ДНФ. Здесь возможны различные приемы. Чтобы описать и объяснить эти

приемы, перейдем к геометрической интерпретации области определения булевой функции — булева куба.

Отдельные элементы, составляющие булев куб $\mathbb{B}^n$, т.е. булевы векторы принято называть ***вершинами куба***. Множество булевых векторов, у которых компоненты заданного набора, например с номерами $i_1, \ldots, i_k$, имеют определенные значения образуют подалгебру булевой алгебры, называемую ***гранью булева куба***. Грань представляет собой булев куб, но меньшей размерности, равной $n - k$, где n — размерность булева куба, k — количество фиксированных номеров. Грань размерности 1 называется ***ребром булева куба***. Кортеж номеров фиксированных компонент, определяющий грань куба называют ***направлением грани***. Грани, имеющие одинаковое направление, не пересекаются. Они называются ***параллельными***. Среди паралельных граней выделим ***соседние***, у которых совпадают значения всех фиксированных компонент, кроме одной.

Грани булева куба удобно обозначать, как слово из трех символов 0, 1 и $*$, где 0 и 1 обозначают фиксированные значения компонент, а $*$ указывает меняющиеся компоненты. Например, слово $00*1$, обозначает ребро четырехмерного булева куба, определяемое условиями $x_1 = 0$, $x_2 = 0$, $x_4 = 0$. Третья компонента, отмеченная звездочкой, варьируется. При таком способе записи размерность грани совпадает с количеством звездочек, грани параллельны, если у них совпадает положение звездочек (например $10*0$ и $01*1$). Грани соседние, если их записи различаются только в одной позиции, в которой для одной грани указано значение 0, а для другой 1 (например $01*0*1$ и $11*0*1$).

Будем говорить, что булева функция f, определенная на $\mathbb{B}^n$, подчиняется булевой функции g, определенной на том же кубе (или f меньше g), если $f(x) \leqslant g(x)$ для любого $x \in \mathbb{B}^n$. В этом случае будем писать $f \leqslant g$. Множество вершин куба, в которых функция f принимает значение 1, будем называть ***уровнем единицы функции***, сами эти вершины называют ***конституентами единицы***. Если $f \leqslant g$, то уровень единицы функции f является подмножеством уровня единицы функции g.

Элементарная конъюнкция принимает значение 1 в том и только в том случае, когда каждый литерал в ней принимает значение 1. Часть переменных может не входить в конъюнкцию, т.е. эти переменные будут фиктивными. Варьируя значения фиктивных переменных, получим грань n-мерного булева куба. Таким образом, уровень единицы любой элементарной конъюнкции есть некоторая грань булева куба.

Любую ДНФ данной функции f можно интерпретировать как набор граней булева куба, объединение которых совпадает с уровнем единицы функции f. При этом каждая элементарная конъюнкция рассматриваемой ДНФ будет подчинена функции f. Любую элементарную конъюнкцию, подчиненную функции f, будем называть ***импликантой***.

Замечание. Далее будем предполагать, что исследуемая функция от n аргументов не меняется, так что можно заранее каждый аргумент функции связать с некоторой переменной, а точнее, с i-м аргументом связывается переменная x_i. Это позволяет не различать булевы функции и формулы, составленные из переменных $x_1, \ldots, x_n$. Именно в этом контексте элементарная конъюнкция трактуется как функция от n аргументов.

Чтобы получить кратчайшую ДНФ заданной функции, надо выбрать минимальное количество импликант (граней булева куба, содержащихся в уровне единицы функции), в совокупности накрывающих уровень 1. Для минимальной ДНФ минимума достигает сумма длин импликант, а для этого надо стремиться увеличить сумму размерностей граней куба, соответствующих элементарным конъюнкциям ДНФ. В самом деле, пусть d_i и n_i, $i = \overline{1, k}$, — длины конъюнкций, составляющих ДНФ, и размерности соответствующих граней булева куба; s — сложность ДНФ; r — сумма размерностей граней, соответствующих конъюнкциям ДНФ. Тогда

$$s = \sum_{i=1}^{k} d_i = \sum_{i=1}^{k} (n - n_i) = kn - r,$$

и мы видим, что увеличение суммы размерностей граней ведет к уменьшению сложности ДНФ.

При небольших размерностях булева куба (до 4) его геометрию можно представить на плоскости с помощью так называемых ***карт Карно***. Представим куб $\mathbb{B}^n$ как декартово произведение двух кубов $\mathbb{B}^k \times \mathbb{B}^l$, где $k, l \leqslant 2$. Такое произведение можно изобразить таблицей, в которой каждая строка соответствует вершине куба $\mathbb{B}^k$, а каждый столбец — вершине куба $\mathbb{B}^l$. Существенным моментом является такой порядок строк и соответственно столбцов, при котором соседним строкам и столбцам отвечают соседние вершины соответствующего булева куба. А именно, при $k, l = 1$, берем естественный порядок 0, 1. при $k, l = 2$ используем порядок 00, 01, 11, 10 (табл. 1.5).

Таблица 1.5

	00	01	11	10
00				
01				
11				
10				

В такой таблице соседние клетки соответствуют соседним вершинам четырехмерного булева куба. Если представить, что эта „шахматная доска“ склеена по горизонтали и вертикали, т.е. соседними, например, являются (00, 11) и (10, 11), а также (01, 00) и (01, 10), то соседние на доске будет в точности соответствовать соседним на кубе.

На карте Карно ребро — это пара соседних (в том числе и „через край“) клеток. Легко изобразить на карте Карно и двумерные грани, состоящие из 4-х вершин. это либо квадрат из двух примыкающих пар клеток, либо одна строка, либо один столбец. Трехмерная грань — это две соседние строки или два соседних столбца.

Таблица 1.6

	000	001	011	010	110	111	101	100
00								
01								
11								
10								

К сожалению, при увеличении размерности такая простая интерпретация исчезает. Например, для пятимерного куба (табл. 1.6), соседние клетки (включая „соседство через край“) будут соответствовать соседним вершинам. Но, кроме того, и другие пары клеток, например (01, 001) и (01, 101), не будучи соседними в таблице, соответствуют соседним вершинам. Причина в том, что на карте Карно у каждой клетки четыре соседних, в то время как в n-мерном кубе каждая вершина имеет n соседних вершин, которые получаются, если в булевом векторе изменить одну компоненту из n возможных.

Рассмотрим два приема, с помощью которых из имеющейся ДНФ можно получить более простую ДНФ (и в смысле длины, и в смысле сложности).

Склейка. Дизъюнктивная нормальная форма данной функции может содержать пару импликант, соответствующих соседним граням булева куба. Но две соседние грани булева куба вместе составляют грань большей размерности. Например, две двумерные грани 01**10 и 01**11 пятимерного куба в совокупности составляют трехмерную грань 01**1*. Замена в ДНФ двух таких импликант одной более короткой приводит к уменьшению и длины, и сложности ДНФ. Такую замену называют ***склейкой***. Импликанты, соответствующие соседним граням, имеют один и тот же набор переменных, причем они различаются только по одной переменной: в одной импликанте стоит сама переменная, а в другой — ее отрицание. Например, грани 01**10 и 01**11 соответствуют элементарным конъюнкциям $\overline{x}_1 x_2 x_4 \overline{x}_5$ и $\overline{x}_1 x_2 x_4 x_5$. Изменение ДНФ выполняется в данном случае в соответствии с тождеством $\overline{x}_1 x_2 x_4 \overline{x}_5 + \overline{x}_1 x_2 x_4 x_5 = \overline{x}_1 x_2 x_4$. Примеры возможных склеек показаны с помощью карты Карно на рис. 1.3.

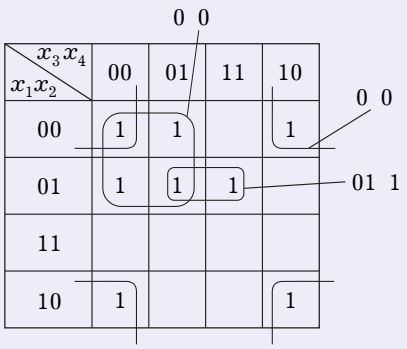


Рис. 1.3

Отталкиваясь от совершенной ДНФ, мы можем сперва склеивать различные пары соседних нульмерных граней (вершин) в ребра, затем в соседние ребра в двумерные грани и т.д. Процесс подобной склейки приведет к выявлению граней максимальной размерности, содержащихся в

уровне единицы функции. Этим граням соответствуют максимальные импликанты функции (максимальные относительно введенного порядка $\leqslant$ на импликантах функции). Такие импликанты называются ***простыми импликантами***. ДНФ данной функции, в которую входят все простые импликанты, называют ***сокращенной ДНФ***.

Отметим, что если исходная ДНФ не является совершенной, то склейка может и не привести к сокращенной ДНФ. Так, если на рис. 1.3 выбрать грани $*0*0$, $01*1$, 0001, 0100, получим покрытие уровня единицы функции. В соответствующей ДНФ $\overline{x}_2\overline{x}_4+\overline{x}_1x_2x_4+\overline{x}_1\overline{x}_2\overline{x}_3x_4+\overline{x}_1x_2\overline{x}_3\overline{x}_4$ нельзя склеить какие-либо грани, но эта ДНФ не является сокращенной, так как два последних слагаемых не являются простыми импликантами.

Избыточные импликанты. В сокращенной ДНФ одна из импликант может накрываться остальными. Такую импликанту можно удалить из формулы, уменьшая и длину, и сложность ДНФ. Рассмотрим, например, функцию, представленную картой Карно на рис. 1.4. У этой функции максимальными являются шесть импликант $1*0$, $10*$, $*01$, $0*1$, $01*$, $*10$. Их сумма даст сокращенную ДНФ. Но из рисунка видно, что, например, импликанта $10*$ накрывается двумя импликантами $1*0$ и $*01$. Значит, эту импликанту можно из ДНФ удалить.

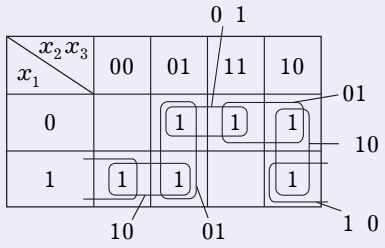


Рис. 1.4

Импликанта в ДНФ, подчиненная сумме остальных импликант этой ДНФ, называется ***избыточной***. Если из сокращенной ДНФ удалить избыточные импликанты, то получим ***тупиковую ДНФ***. По определению, тупиковая ДНФ — это ДНФ, состоящая из простых импликант, ни одна из которых в этой ДНФ не является избыточной. Например, на рис. 1.4 тупиковую ДНФ образуют импликанты $1*0$, $*01$, $01*$.

Основной целью каждого метода минимизации ДНФ является выявление всех тупиковых ДНФ. Именно среди них находится минимальная и кратчайшая формы.

Построение сокращенной ДНФ. Один из методов построения сокращенной ДНФ — это ***метод Квайна — МакКлоски***. Его суть в следующем. В качестве исходной выбирается совершенная ДНФ. На первом этапе проводится склейка импликант совершенной ДНФ в ребра, которые добавляются в ДНФ. После того как проведены все склейки вершин, из ДНФ удаляются все избыточные вершины, т.е. те, которые подчиняются добавленным ребрам. На втором этапе проводится склейка всех ребер в четырехмерные грани, которые добавляются к ДНФ. После проведения всех склеек между ребрами удаляются ребра, подчиненные двумерным граням. На третьем этапе выполняется склейка двумерных граней и удаление избыточных двумерных граней. Процесс останавливается, когда между гранями соответствующей размерности не удается провести ни одной склейки. Результатом этих преобразований и будет сокращенная ДНФ.

Пример 1.6. Рассмотрим функцию, заданную картой Карно на рис. 1.5. Первый этап склейки по методу Квайна приведет к ребрам $1*00$, $10*0$, $*010$, а также четырем ребрам двумерной грани $0**1$ и четырем ребрам двумерной грани $0*1*$. Все исходные импликанты будут удалены как избыточные.

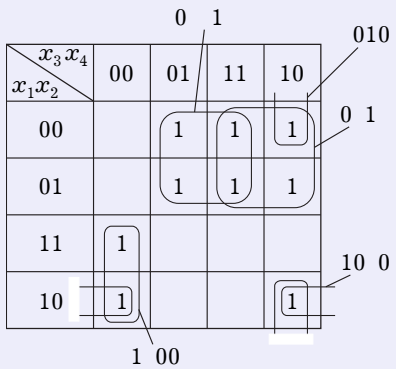


Рис. 1.5

На втором этапе мы получим две двумерные грани $0**1$ и $0*1*$. При этом будут удалены 6 ребер, входящих в эти грани. Поскольку склеить две эти грани нельзя, процесс выявления простых импликант на этом завершится. Сокращенная ДНФ будет содержать пять импликант, соответствующих граням $0**1$, $0*1*$, $1*00$, $10*0$, $*010$. #

Другой метод построения сокращенной ДНФ — ***метод Блейка***, в котором в качестве исходной можно взять любую ДНФ. В этом методе на первом этапе многократно выполняется операция обобщенного склеивания, при которой сумма

$xK_1 + \overline{x}K_2$ заменяется суммой $xK_1 + \overline{x}K_2 + K_1K_2$. Такая операция выполняется до тех пор, пока удается получать новые импликанты вида K_1K_2. После завершения первого этапа выполняется операция поглощения согласно формуле $K_1K_2 + K_2 = K_2$.

Выявление всех тупиковых ДНФ. Создать список всех тупиковых ДНФ можно, используя следующий прием. Обозначим все простые импликанты символами K_1, K_2, …, K_m. Перенумеруем все вершины уровня единицы функции. Пусть первая вершина накрывается склейками K_{i_1}, K_{i_2}, …, K_{i_p}. Тогда элементарная дизъюнкция $K_{i_1} + K_{i_2} + \ldots + K_{i_p}$ задает условие, что первая вершина уровня единицы функции накрывается одной из простых импликант. Составив конъюнкцию из указанных элементарных дизъюнкций по всем вершинам уровня единицы функции, мы получим КНФ, представляющее собой логическое условие того, что весь уровень единицы функции накрыт простыми импликантами. Эта КНФ называют ***функцией Патрика***. Преобразуем КНФ в ДНФ, пользуясь свойствами булевых операций (дистрибутивностью, коммутативностью, идемпотентностью). В ДНФ каждая элементарная конъюнкция будет описывать набор простых импликант, целиком покрывающих уровень единицы функции, т.е. набор, описывающий конкретную тупиковую ДНФ рассматриваемой функции. Можно показать, что таким образом можно выявить все тупиковые ДНФ.

Среди простых импликант рассматриваемой функции есть те, которые входят в любую тупиковую ДНФ. К таким относится любая импликанта, для которой среди накрываемых вершин есть хотя бы одна, не накрываемая никакой другой импликантой. Например, на рис. 1.5 вершина 1100 накрывается единственной простой импликантой 1∗00. Эта импликанта должна входить в любую тупиковую ДНФ. Импликанты, удовлетворяющие описанному условию, называются ***ядровыми импликантами***. Совокупность ядровых импликант — ***ядро*** — постоянная часть всех тупиковых ДНФ. Например, на рис. 1.5 ядро составляют импликанты 0∗∗1, 0∗1∗, 1∗00, а импликанты ∗010 и 10∗0 неядровые. При выявлении тупиковых ДНФ ядровые импликанты и накрываемые ими вершины из функции Патрика можно исключить.

Пример 1.7. У функции, представленной на рис. 1.5 диаграммой Карно, имеется пять простых импликант. Введем обозначения:

$$K_1 = 0{**}1, \quad K_2 = 0{*}1{*}, \quad K_3 = 1{*}00, \quad K_4 = 10{*}0, \quad K_5 = {*}010.$$

Тогда функция Патрика для 9 вершин уровня 1 при их нумерации по строкам будет иметь следующий вид:

$$P = K_1(K_1 + K_2)(K_2 + K_5)K_1(K_1 + K_2)K_2K_3(K_3 + K_4)(K_4 + K_5).$$

Упростив согласно идемпотентности операций и тождествам поглощения ($x(x + y) = x$), получим

$$P = K_1K_2K_3(K_4 + K_5).$$

Используя дистрибутивность и идемпотентность, приходим к ДНФ:

$$P = K_1K_2K_3(K_4 + K_5) = K_1K_2K_3K_4 + K_1K_2K_3K_5.$$

Таким образом, рассматриваемая функция имеет две сокращенные ДНФ, описываемые двумя списками склеек K_1, K_2, K_3, K_4 и K_1, K_2, K_3, K_5. В результате получим

$$D_1 = \overline{x}_1x_4 + \overline{x}_1x_3 + x_1\overline{x}_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_4, \qquad D_2 = \overline{x}_1x_4 + \overline{x}_1x_3 + x_1\overline{x}_3\overline{x}_4 + \overline{x}_2x_3\overline{x}_4.$$

Видно, что у обоих сокращенных ДНФ есть общая часть — первые три конъюнкции. Это ядро, входящее в каждую сокращенную ДНФ. Его можно исключить из рассмотрения, рассматривая условия накрытия только для тех вершин, которые не накрываются склейками ядра. В данном случае это единственная вершина 1010. При этом функция Патрика будет иметь вид

$\hat{P} = K_4 + K_5$. Добавив к каждому слагаемому элементарную конъюнкцию $K_1K_2K_3$, описывающую ядро, получим полную функцию Патрика. #

Ядро данной функции можно получить с помощью ***таблицы Квайна***, в которой каждая строка соответствует одной простой импликанте, а каждый столбец — одной вершине уровня единицы функции. В каждой клетке указывается 1, если импликанта накрывает вершину, и пробел в противном случае Например, составим таблицу Квайна функции, представленной картой Карно на рис. 1.3 (табл. 1.7). Чтобы определить ядро функции, необходимо выявить столбцы, в которых находится только одна единица, и пометить соответствующие строки. Импликанты, соответствующие помеченным строкам, образуют ядро. Например, в табл. 1.7 по одной единице содержится в столбцах, отвечающих вершинам 0001, 0010, 0100, 0111, 1000, 1010 (эти единицы выделены полужирным шрифтом). Поскольку в каждой строке содержится хотя бы одна выделенная единица, все простые импликанты ядровые, т.е. рассматриваемая функция имеет единственную тупиковую ДНФ.

Таблица 1.7

	0000	0001	0010	0100	0101	0111	1000	1010
0∗0∗	1	**1**		**1**	1			
∗0∗0	1		**1**				**1**	**1**
01∗1					1	**1**		

ОГЛАВЛЕНИЕ

**Московский государственный технический университет
имени Н.Э. Баумана**

Факультет «Фундаментальные науки»
Кафедра «Математическое моделирование»

А.Н. Канатников

ДИСКРЕТНАЯ МАТЕМАТИКА

Конспект лекций

Для студентов специальности
«Прикладная математика»

Москва
2006

2. ЛОГИКА ВЫСКАЗЫВАНИЙ

2.1. Алгебра высказываний

Высказывания и их истинностные значения. Логические операции $\vee$, $\wedge$, $\to$, $\sim$, $\neg$. Пропозициональные операции и связки. Пропозициональные формулы: пропозициональные переменные и шаг индукции ($X \vee Y$, $X \wedge Y$, $X \to Y$, $\neg X$). Язык и метаязык. Истинностные функции и пропозициональные формулы. Утверждение: каждая истинностная функция соответствует некоторой пропозициональной формуле.

Напомню, что под высказыванием понимается любое предложение, относительно которого можно сказать истинно оно или нет, т.е. утвердительное предложение. Строго говоря, сказанное не следует рассматривать как настоящее математическое определение — лишь как указание на те реальные объекты, которые могут служить иллюстрацией к точной математической теории.

Формирование предложений в естественном языке указывает на то, что высказывания могут объединяться с помощью связывающих союзов. С математической точки зрения эти союзы реализуют операции над высказываниями. Выделим пять таких операций:

1) дизъюнкция (соответствует союзу „или“);
2) конъюнкция (соответствует союзу „и“);
3) импликация (соответствует фразе типа „если ..., то „);
4) эквиваленция (соответствует фразе типа „... тогда и только тогда, когда ...);
5) отрицание (соответствует союзу „не“).

Первые четыре из этих операций бинарные, последняя унарная. Относительно указанных операций можно сказать лишь, что они формируют новое высказывание. При этом, зная, истинны или ложны исходные высказывания, можно сказать, является ли истинным вновь образованное высказывание. Введенные пять операций мы будем называть ***логическими*** или ***пропозициональными***.

Мы имеем дело с некоторой алгебраической системой, и для нее можно ввести свой математический язык — язык алгебры высказываний. В этом языке из заданного набора символов — алфавита языка — по определенным правилам составляются последовательности, называемые словами или фразами, формулами.

Алфавит языка алгебры высказываний составляют множество ***пропозициональных переменных***, множество *функциональных символов* (символов операций, или ***логических связок***) $\vee$, $\wedge$, $\to$, $\sim$, $\neg$ и множество служебных символов (две круглые скобки). Формулы языка вводятся индуктивно.

База индукции: пропозициональные переменные представляют собой формулы.

Шаг индукции: если X и Y — формулы, то формулами являются $(X \vee Y)$, $(X \wedge Y)$, $(X \to Y)$, $(X \sim Y)$, $(\neg X)$.

Договоримся о следующих обозначениях. Будем обозначать: пропозициональные переменные — строчными латинскими буквами конца алфавита (x, y, z и т.д.); какие-либо формулы — прописными латинскими буквами конца алфавита (X, Y, Z и т.д.). Как и в теории булевых функций, для сокращения количества скобок в формулах договоримся о таком же приоритете операций. Это соглашение не относится к самому языку ми служит лишь для удобства.

В наших рассуждениях будут встречаться формулы, которые относятся к введенному языку, но, кроме того, придется использовать и дополнительные обозначения и символику, чтобы рассуждать не в рамках языка, а о самом языке и его формулах. Так, обозначения переменных — это элемент языка алгебры высказываний, а обозначения формул уже выходят за рамки языка.

Дополнительное соглашение о приоритете операций также выходит за рамки языка. В этом случае говорят о расширении рассматриваемого языка или о ***метаязыке***. На практике язык и метаязык тесно переплетаются, и разделить их не просто.

Слова языка алгебры высказываний, называемые ***пропозициональными формулами***, — лишь цепочки символов, составленные по определенным правилам. Они получают содержательный смысл, если ввести какую-либо *интерпретацию* рассматриваемого языка. Естественная интерпретация — ввести область изменения каждой пропозициональной переменной как множество всех высказываний и сопоставить каждому символу соответствующую логическую операцию. Тогда каждая формула будет определять правило, по которому из некоторого набора исходных высказываний, обозначенных переменными, мы получим новое высказывание. Сама подстановка вместо переменных конкретных высказываний уже выходит за рамки рассматриваемого языка. Отметим, что формулы алгебры высказываний имеют и другие интерпретации. Например, пропозициональные переменные можно рассматривать как булевы переменные, а функциональные символы увязать с соответствующими булевыми операциями. Такая интерпретация позволяет получить исходя из истинности высказываний истинность сложного высказывания. Формула будет определять булеву функцию, которую называют ***истинностной функцией***. Именно истинностные функции и следует рассматривать как главную цель изучения в алгебре высказываний, поскольку она позволяет судить об истинности сложных, запутанных высказываний.

Теорема 2.1. Каждая булева функция является истинностной функцией некоторой формулы алгебры высказываний.

◀ Язык алгебры высказываний совпадает с языком булевой алгебры, построенным на множестве из пяти булевых функций $\vee$, $\wedge$, $\to$, $\sim$, $\neg$. Поскольку это множество содержит стандартный базис в $\mathbb{B}$ и, следовательно, полно, любая булева функция может быть представлена формулой над заданным множеством. Эту формулу можно рассматривать как формулу алгебры высказываний. При этом булева функция будет истинностной функцией указанной формулы. ▶

2.2. Тавтологии и эквивалентность формул

Тавтологии. Формулы выполнимые и опровержимые. Теорема о правиле modus ponens: Если X и $X \to Y$ — тавтологии, то и Y — тавтология. Подстановка $\mathbf{S}(z_1, z_2, \ldots, z_n | Y_1, Y_2, \ldots, Y_n | X)$ (результат подстановки Y_i вместо z_i) — пропозициональная формула. Эквивалентность формул алгебры высказываний. Теорема о заменах. Следствие 1: если X — тавтология, то и $\mathbf{S}(z_1, z_2, \ldots, z_n | Y_1, Y_2, \ldots, Y_n | X)$ — тавтология. Следствие 2: инвариантность эквивалентности относительно логических операций.

Среди формул алгебры высказываний выделяют:

– ***выполнимые***, имеющие значение 1 хотя бы для одного набора значений пропозициональных переменных;

– ***опровержимые***, имеющие значение 0 хотя бы для одного набора значений пропозициональных переменных.

Формула, не являющаяся опровержимой, истинна при любом значении переменных. Такую формулу называют ***тождественно истинной*** или ***тавтологией***. Тавтологии описывают универсальные логические законы. Именно с использованием тавтологий проводится любое математическое доказательство.

Формула, не являющаяся выполнимой, ложна при любом значении переменных. Такую формулу называют ***тождественно ложной*** или ***противоречием***.

Для тавтологий (противоречий) истинностная функция есть константа 1 (0). Для выполнимых формул истинностная функция не равна постоянной 0, а для опровержимых — постоянной 0.

Пример 2.1. Формула $(x \vee (\neg y)) \to z$ является одновременно выполнимой и опровержимой: она истинна, если значением переменной z является истинное высказывание, и ложна, если, например, значениями переменных y и z являются ложные высказывания. Это можно увидеть составив истинностную функцию, которая в данном случае описывается вектором 01110101. Формула $(x \vee (\neg x))$ является тавтологией, а формула $(x \wedge (\neg x))$ — противоречием. #

Следующее утверждение отражает часто используемое на практике умозаключение, называемое modus ponens (модус поненс).

Теорема 2.2. Если формулы X и $X \to Y$ являются тавтологиями, то и формула Y есть тавтология.

◄ Пусть формулы X и Y построены из переменных $z_1, \ldots, z_n$. Выберем для этих переменных какие-либо значения. Тогда об истинности формулы $X \to Y$ можно судить на основании истинности формул X и Y. Анализируя истинностную функцию для импликации, видим, что при истинности X и ложности Y формула $X \to Y$ является ложной. Но по условию теоремы эта формула тождественно истинная, как и формула X. Следовательно, формула Y не может быть ложной при выбранных значениях переменных. Поскольку значения переменных выбирались произвольно, заключаем, что формула Y тождественно истинна, т.е. тавтология. ►

Как и в булевой алгебре, введем понятие *эквивалентных формул* алгебры высказываний — формул, имеющих равные истинностные значения при любых значениях входящих в формулы переменных. Альтернативное определение: формулы X и Y называются эквивалентными, если формула $X \sim Y$ является тавтологией. Нетрудно показать, рассуждая как в последней теореме, что формула $X \sim Y$ является тавтологией тогда и только тогда, когда при любых значениях переменных формулы X и Y одновременно или истинны, или ложны, т.е. истинностные функции этих формул совпадают. Для эквивалентных формул введем обозначение $X \equiv Y$. Итак, $X \equiv Y \Leftrightarrow X \sim Y$ — тавтология. Обратите внимание на три символа эквивалентности в последней фразе. Первый обозначает отношение эквивалентности формул, введенное на множестве формул алгебры высказываний, третий — операцию эквиваленции, а второй — по сути та же эквиваленция, но в утверждении, в котором формулируется свойство самой алгебры высказываний, и ее следует отнести к сфере метаязыка.

В теории булевых функций мы уже использовали некоторые стандартные приемы, приводящие к эквивалентным формулам. Один из них — подстановка. Если мы в формуле заменим одну из подформул другой, то получим новую цепочку символов. Нетрудно доказать индукцией по построению, что это цепочка будет формулой. Заменяемая подформула может встречаться несколько раз. В этом случае говорят о ***вхождениях подформулы***. Замена может выполняться для одного какого-либо вхождения данной подформулы или для всех. Из теории булевых функций вытекает следующий результат.

Теорема 2.3. 1) Если $X \equiv Y$, то при замене всех вхождений какой-либо переменной и в X, и в Y какой-либо формулой Z получим эквивалентные формулы.

2) Если в формуле X заменить одно из вхождений подформулы Y эквивалентной формулой Z, то получим формулу, эквивалентную X.

Под подстановкой будем понимать замену всех вхождений в формулу одной или нескольких переменных некоторыми формулами. Результат подстановки в формулу X вместо переменных $z_1, \ldots, z_n$ формул $Y_1, \ldots, Y_n$ обозначают примерно так: $\mathbf{S}(z_1, \ldots, z_n | Y_1, \ldots, Y_n | X)$.

Следствие 2.1. Если X — тавтология, то и $\mathbf{S}(z_1, \ldots, z_n | Y_1, \ldots, Y_n | X)$ — тавтология.

◄ Можно рассуждать так. Тавтологии — это формулы, эквивалентные, например, формуле $W = x \vee \neg x$. В качестве переменной x можно выбрать ту, которая не входит в формулу X. В силу теоремы, заменив в формулах X и W все вхождения переменных $z_1, \ldots, z_n$ формулами $Y_1, \ldots, Y_n$, мы получим эквивалентные формулы. Но формула W при этом не изменится.

Поэтому вновь построенная формула $\mathbf{S}(z_1,\dots,z_n|Y_1,\dots,Y_n|X)$ будет эквивалентна формуле W, т.е. будет являться тавтологией. ►

Следствие 2.2. Пусть $X \equiv Z$ и $Y \equiv W$. Тогда $(X \vee Y) \equiv (Z \vee W)$, $(X \wedge Y) \equiv (Z \wedge W)$, $(X \to Y) \equiv (Z \to W)$, $(X \sim Y) \equiv (Z \sim W)$, $(\neg X) \equiv (\neg Z)$.

◄ Формулу $(Z \circ W)$, где $\circ$ — одна из логических связок, можно рассматривать как результат замены в формуле $(X \circ Y)$ сперва подформулы X эквивалентной формулой Z, а затем подформулы Y эквивалентной формулой W. Согласно доказанной теореме такая замена приводит к эквивалентной формуле. Аналогичны рассуждения и для отрицания. ►

2.3. Способы получения эквивалентных формул

Эквивалентности на основе свойств логических операций (коммутативность, ассоциативность, идемпотентность, дистрибутивность, поглощение). Эквивалентности на основе взаимосвязей операций. Эквивалентности на основе двойственности

С помощью подстановки можно получать эквивалентные формулы, отталкиваясь от известных свойств логических операций.

Теорема 2.4. Для любых пропозициональных формул X, Y и Z верны следующие эквивалентности: 1) $X \wedge X \equiv X$; 2) $X \wedge Y \equiv Y \wedge X$; 3) $(X \wedge Y) \wedge Z) \equiv (X \wedge (Y \wedge Z)$; 4) $X \vee X \equiv X$; 5) $X \vee Y \equiv Y \vee X$; 6) $(X \vee Y) \vee Z) \equiv (X \vee (Y \vee Z)$; 7) $X \wedge (Y \vee Z) \equiv (X \wedge Y) \vee (X \wedge Z)$; 8) $X \vee (Y \wedge Z) \equiv (X \vee Y) \wedge (X \vee Z)$; 9) $X \wedge (Y \vee X) \equiv X$; 10) $X \vee (Y \wedge X) \equiv X$.

◄ Непосредственно из таблицы для истинностной функции вытекает, что $x \wedge x \equiv x$. Подставив вместо всех вхождений переменной x формулу X, получим эквивалентность $X \wedge X \equiv X$. Остальные эквивалентности доказываются аналогично. ►

Еще один способ получения эквивалентностей — замена одних операций другими по соответствующим формулам. Из теории булевых функций вытекает, что верны следующие эквивалентности:

$$(\neg(\neg x)) \equiv x, \quad (\neg(x \vee y)) \equiv (\neg x) \wedge (\neg y), \quad (x \to y) \equiv ((\neg x) \vee y).$$

Отталкиваясь от этих эквивалентностей можно доказать следующее.

Теорема 2.5. Для любых пропозициональных формул X, Y и Z верны следующие эквивалентности:

1) $(\neg(\neg X)) \equiv X$ (закон двойного отрицания);
2) $(\neg(X \vee Y)) \equiv (\neg X) \wedge (\neg Y))$ (перенос отрицания через конъюнкцию);
3) $(\neg(X \wedge Y)) \equiv (\neg X) \vee (\neg Y))$ (перенос отрицания через дизъюнкцию);
4) $(\neg(X \to Y)) \equiv (X \wedge (\neg Y))$ (перенос отрицания через импликацию);
5) $(X \to Y) \equiv ((\neg X) \vee Y)$ (представление импликации через дизъюнкцию);
6) $(X \to (\neg X)) \equiv (\neg X)$ (закон упрощения);
7) $(X \to Y) \equiv ((\neg Y) \to (\neg X)$ (закон контрапозиции).

◄ Доказывается теорема так же, как и предыдущая. Например, на основании простой эквивалентности $(\neg(\neg x)) \equiv x$, устанавливаемой непосредственно, путем подстановки вместо переменной x формулы X получаем эквивалентность $(\neg(\neg X)) \equiv X$. ►

Понятие двойственности из теории булевых функций переносится на алгебру высказываний. В данном случае речь идет о формулах, содержащих только базовые операции $\vee$, $\wedge$, $\neg$. Для любой такой формулы X двойственная формула X^* получается взаимной заменой операций $\vee$ и $\wedge$.

Переход к двойственной формуле соответствует переходу от булевой функции $f(x)$ к двойственной функции $\overline{f(\overline{x})}$. Понятие двойственных функций позволяет ввести понятие двойственности для любых формул алгебры высказываний, однако для произвольных формул двойственность не является такой простой, как в случае трех базовых операций. Из понятия двойственных функций вытекает, что двойственность — симметричное отношение, т.е. $X^{**} = X$ (здесь знак равенства означает не эквивалентность формул, а их совпадение). Из понятия двойственности функций вытекает и следующая эквивалентность:

$$X^{**} \equiv \mathbf{S}(t_1, \ldots, t_n | \neg t_1, \ldots, \neg t_n | \neg X),$$

где t_1, …, t_n — полный список переменных формулы X.

Каждая булева функция может быть представлена формулой над стандартным базисом. При этом булева функция, не равная тождественно 0, может быть представлена совершенной дизъюнктивной нормальной формой, а булева функция, не равная тождественно 1, — совершенной конъюнктивной нормальной формой. Отсюда вытекает следующее утверждение.

Теорема 2.6. Каждая формула алгебры высказываний, не являющаяся противоречием, имеет эквивалентную ей совершенную ДНФ. Каждая формула алгебры высказываний, не являющаяся тавтологией, имеет эквивалентную ей совершенную КНФ. #

ОГЛАВЛЕНИЕ

**Московский государственный технический университет
имени Н.Э. Баумана**

Факультет «Фундаментальные науки»
Кафедра «Математическое моделирование»

А.Н. Канатников

ДИСКРЕТНАЯ МАТЕМАТИКА

Конспект лекций

Для студентов специальности
«Прикладная математика»

Москва
2006

3. ИСЧИСЛЕНИЕ ВЫСКАЗЫВАНИЙ

3.1. Введение

Об аксиоматическом методе. Современные представления. Исчисления. Теоремы и вывод. Теория и метатеория. Цели формализации в математике. Логика высказываний и ее формализация — теория $\boldsymbol{K}$.

В основе современной математики лежит аксиоматический метод. Суть его в том, что математическая теория строится путем вывода всех утверждений (теорем) из относительно небольшого числа утверждений, которые заранее считаются истинными (это аксиомы).

Именно в таком ключе аксиоматический метод был введен в математику в Древней Греции. В современной математике аксиоматический метод получил дальнейшее развитие. Детальному анализу подверглись не только сами выводы конкретных математических теорий, но и способы доказательств. Однако, чтобы проанализировать приемы математических рассуждений и их обоснованность формализации нужно подвергнуть не только предметную область, т.е. содержательные понятия конкретной математической теории, но и приемы логических умозаключений.

В свете этого полностью формализованная математическая теория выглядит так. Имеется некоторый язык, позволяющий составлять правильные слова-формулы, которые отражают возможные утверждения теории. Есть некоторый набор формул, изначально объявленных истинными (это аксиомы). Кроме того, задан некоторый набор правил преобразования формул, которые позволяют из истинных формул получать новые истинные формулы. Все истинные утверждения теории получаются путем формальных преобразований формул в рамках узаконенных правил преобразований. Цепочка последовательных преобразований называется ***выводом***.

В полностью формализованной теории утрачивается содержательный смысл формул, а все построение теории превращается в манипуляции заданными символами. В силу этого такие теории называют ***исчислениями***.

Построение исчислений не является высшей целью развития математики. На это указывает положение дел в таких классических областях математики, как дифференциальное и интегральное исчисления, геометрия. Эти разделы, несмотря на их многолетнюю историю, по-прежнему излагаются на содержательном, а не формальном, уровне. Дело в том, что формализация — это определенный метод математического исследования, который используется при исследовании теорий на непротиворечивость, наличие зависимых понятий и т.п.

Отметим, что при изложении формальной теории приходится выходить за ее рамки, чтобы проводить содержательный анализ ее результатов. Поэтому вокруг формальной теории возникает другая теория: „учение о формальной теории“, которую называют ***метатеорией***. Метатеория формирует свой язык, расширяющий язык формальной теории, который называется ***метаязыком***. На это было обращено внимание при изложении логики высказываний.

Логика высказываний не является формальной теорией, хотя и имеет собственный язык. Все суждения в логике высказываний оказываются за пределами ее языка и устанавливаются на базе истинностных функций, которые никак не отражаются в языке логики высказываний.

Реально все необходимые выводы логики высказываний можно было бы сделать, не прибегая к ее полной формализации. Но тогда теряется связь получения тех или иных утверждений с реальный процессом умозаключений, который и представляет собой подлинное математическое доказательство. Кроме того, логика высказываний в основном сводится к теории булевых функций, т.е. к исследованию конечных объектов. Поэтому формальная теория логики высказываний — одна из самых простых и в этом смысле удобна как иллюстрация современного подхода к формализации в математике.

Любая теория имеет множество вариантов формализации. В рамках логики высказываний мы остановимся на одном из этих вариантов, который назовем ***теорией K***.

3.2. Основные положения теории N

Язык теории $\boldsymbol{N}$. Аксиомы (их одиннадцать): 1) $X \to (Y \to X)$; 2) $X \to Y \to (X \to (Y \to Z) \to (X \to Z))$; 3) $X \wedge Y \to X$; 4) $X \wedge Y \to Y$; 5) $(X \to Y) \to ((X \to Z) \to (X \to Y \wedge Z))$; 6) $X \to X \vee Y$; 7) $Y \to X \vee Y$; 8) $X \to Z \to (Y \to Z \to (X \vee Y \to Z))$; 9) $X \to Y \to (\neg Y \to \neg X)$; 10) $\neg\neg X \to X$; 11) $X \to \neg\neg X$. Правило вывода $\dfrac{X,\ X \to Y}{Y}$ (modus ponens). Вывод в теории $\boldsymbol{N}$. Вывод из гипотез. Пример: $X \to X$. Дерево вывода.

Язык теории $\boldsymbol{N}$ — это язык алгебры высказываний. В теории $\boldsymbol{N}$ одиннадцать ***схем аксиом***. Схема аксиом отличается от аксиомы тем, что в ней используются не конкретные переменные, а символы подстановки, вместо которых могут подставляться конкретные формулы теории. При выборе вместо символов подстановки конкретных формул мы получаем конкретную аксиому. Отметим, что можно было бы избежать схем аксиом, но тогда придется вводить дополнительные правила вывода, которые позволили бы „размножить“ аксиому.

Итак, сформулируем одиннадцать схем аксиом теории $\boldsymbol{N}$:

1) $X \to (Y \to X)$;
2) $X \to Y \to (X \to (Y \to Z) \to (X \to Z))$;
3) $X \wedge Y \to X$;
4) $X \wedge Y \to Y$;
5) $(X \to Y) \to ((X \to Z) \to (X \to Y \wedge Z))$;
6) $X \to X \vee Y$;
7) $Y \to X \vee Y$;
8) $X \to Z \to (Y \to Z \to (X \vee Y \to Z))$;
9) $X \to Y \to (\neg Y \to \neg X)$;
10) $\neg\neg X \to X$;
11) $X \to \neg\neg X$.

В теории $\boldsymbol{N}$ всего одно правило вывода — правило заключения (modus ponens):

$$\frac{X,\ X \to Y}{Y}.$$

Представленная запись означает, что из двух „правильных“ формул вида X и $X \to Y$ вытекает „правильная“ формула Y.

После введения языка, аксиом и правил вывода формальная теория готова. Дальше можно начинать „игру в слова“ и получать теоремы (т.е. выводимые формулы) этой теории. На этом, собственно, заканчивается формальная часть и начинается неформальная, т.е. метатеория. Основной вопрос: что дает формальная теория в качестве выводимых формул. Отметим, что добросовестный вывод даже простых теорем оказывается весьма трудоемким, и следует прибегать к приему, хорошо известному математикам: использовать уже выведенные теоремы наравне с аксиомами.

Выводом теории $\boldsymbol{N}$ будем называть последовательность формул $X_1, X_2, \ldots, X_n$, в которой каждая формула X_i есть либо аксиома, либо получена из каких-либо предшествующих формул X_k, X_m $(k, m < i)$ по правилу modus ponens. Формула X называется выводимой в теории $\boldsymbol{N}$, если она является конечной формулой некоторого вывода. Этот факт будем обозначать следующим образом: $\vdash X$.

Для нас будет важен условный вывод, или выод из гипотез, при котором некоторые формулы мы предполагаем истинными и на основании этого строим вывод. Такой условный вывод играет промежуточную роль, позволяя рассматривать отдельные части окончательного вывода. Кроме того, условный вывод можно рассматривать как построение вывода из нелогических аксиом, которые присутствуют во всех математических теориях (они характеризуют основные положения теории). Далее большими греческими буквами будем обозначать списки формул теории $\boldsymbol{N}$.

Выводом из гипотез Γ в теории $\boldsymbol{N}$ будем называть последовательность формул, в которой каждая формула есть либо аксиома, либо формула из списка Γ, либо она получена из предшествующих формул по правилу modus ponens. При этом конечная формула X любого вывода из гипотез Γ называется ***выводимой из гипотез*** Γ, что обозначается следующим образом: $\Gamma \vdash X$.

Пример 3.1. Построим вывод формулы $X \to X$, где X — какая-либо формула теории $\boldsymbol{N}$:

1) из схемы 1 получаем аксиому $X \to (X \to X)$;

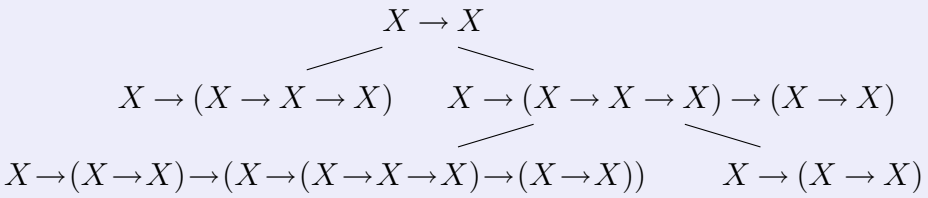
2) из схемы 2, заменяя Z на X и Y на $X \to X$, получаем аксиому $X \to (X \to X) \to (X \to (X \to \to X \to X) \to (X \to X))$,

3) из двух предыдущих формул по правилу вывода $X \to (X \to X \to X) \to (X \to X)$;

4) из схемы 1 при $X \to X$ взамен Y получаем аксиому $X \to (X \to X \to X)$;

5) из двух последних формул по правилу вывода $X \to X$.

Следует заметить, что фактически вывод представляет собой особую структуру — дерево: каждая формула вывода есть либо аксиома или гипотеза (лист дерева, начальный элемент структуры), либо имеет предшественников, из которых она получена по правилу вывода. Для формулы $X \to X$ из последнего примера дерево вывода имеет следующий вид:



3.3. Правила естественного вывода

Теорема о дедукции: если $\Gamma, X \vdash Y$, то $\Gamma \vdash X \to Y$. Структурные правила естественного вывода. Логические правила естественного вывода. Дополнительные правила естественного вывода: присоединение посылки; закон противоречия; двойное отрицание; доказательство от противного; удаление конъюнкции справа. Обобщенное правило введения дизъюнкции.

Одна из целей введения исчисления высказываний — анализ используемой практики построения доказательств. В качестве единственного правила вывода в теории взято правило modus ponens, в то время как построение доказательств на практике использует и многие другие правила: правило исключенного третьего, доказательство от противного и т.п. Оказывается, что все подобные правила можно получить из аксиом исчисления высказываний и правила modus ponens.

Веденный символ $\vdash$ выводимости в исчислении высказываний позволяет строить формулы нового типа $\Gamma \vdash X$, которые на содержательном уровне можно интерпретировать так: „если истинны формулы списка Γ, то истинна формула X. Это формулы метаязыка, позволяющие упростить процесс установления того, выводима данная формула или нет. Само правило modus ponens можно трактовать как формулу $X, X \to Y \vdash Y$. Формулу вида $\Gamma \vdash X$ будем называть ***секвенцией***. Секвенция — это логическая формула, которая может быть истинной или нет.

Чтобы приблизиться к общепринятой практике доказательств, выведем в теории $\boldsymbol{N}$ ряд дополнительных правил, называемых ***правилами естественного вывода***. Следующая теорема дает основополагающее правило естественного вывода, в некотором смысле обращающее правило modus ponens. Фактически эта теорема представляет собой утверждение об эквивалентности символа импликации $\to$ и символа выводимости $\vdash$.

Теорема 3.1 (теорема о дедукции). Если $\Gamma, X \vdash Y$, то $\Gamma \vdash X \to Y$.

◀ Доказательство строится на анализе последовательности вывода. Фактически надо показать, что для любого вывода $Z_1, Z_2, \ldots, Z_n = Y$ из гипотез Γ, X существует вывод из гипотез Γ формулы $X \to Y$. Доказательство проведем индукцией по длине вывода.

При $n = 1$ конечная формула вывода $Z_n = Y$ есть либо аксиома, либо формула из списка Γ, либо формула X (правило modus ponens в данном случае не применялось, так как длина вывода меньше двух). В первом и втором случаях строим последовательность формул Y, $Y \to (X \to Y)$ (аксиома схемы 1), $X \to Y$ (modus ponens), которая является выводом формулы X и списка гипотез Γ. В третьем случае $X = Y$ и формула $X \to Y$ совпадает с выводимой формулой $X \to X$ (см. пример 3.1). Таким образом, при $n = 1$ утверждение доказано.

Предположим, что утверждение доказано для всех формул Y, имеющих вывод из гипотез Γ, X длины менее n. Рассмотрим произвольный вывод $Z_1, Z_2, \ldots, Z_n = Y$. Формула Y есть либо аксиома, либо формула из списка Γ, либо X, либо получена по правилу modus ponens. В первых трех случаях рассуждения те же, что и при $n = 1$ (в выводе можно оставить только последнюю формулу и свести дело к $n = 1$). Рассмотрим случай, когда Y получена по правилу modus ponens из формул Z_k и Z_j. В этом случае одна из формул, например Z_j, есть импликация $Z_k \to Y$. В соответствии с индукционным предположением из гипотез Γ выводимы формулы $X \to Z_k$ и $X \to (Z_k \to Y)$. Объединим два этих вывода, удалив из списка повторения формул, если они есть*. Дополняем полученное объединение следующими тремя формулами:

$X \to Z_k \to (X \to (Z_k \to Y) \to (X \to Y))$ (аксиома схемы 2);

$X \to (Z_k \to Y) \to (X \to Y)$ (modus ponens);

$X \to Y$ (modus ponens).

В результате получаем вывод формулы $X \to Y$ из гипотез Γ. В соответствии с методом математической индукции утверждение теоремы доказано для конечной формулы любого вывода из гипотез Γ, X. ▶

Следующая теорема устанавливает пять правил, называемых ***структурными правилами естественного вывода***. Правила сформулированы не для формул языка алгебры высказываний, а для секвенций. Они позволяют из уже известных выводимостей получать новые. При этом соответствующий вывод на самом деле не строится, а лишь формулируется заключение о существовании такого вывода.

Теорема 3.2. Для любых списков формул Γ и Δ и любых формул X, Y, Z истинны следующие секвенции:

1) $\Gamma, X, \Delta \vdash X$ (закон тождества);

2) $\dfrac{\Gamma \vdash X}{\Gamma, \Delta \vdash X}$ (правило добавления гипотез);

3) $\dfrac{\Gamma, Y, Y, \Delta \vdash X}{\Gamma, Y, \Delta \vdash X}$ (правило сокращения гипотез);

4) $\dfrac{\Gamma, Y, Z, \Delta \vdash X}{\Gamma, Z, Y, \Delta \vdash X}$ (правило перестановки гипотез);

5) $\dfrac{\Delta \vdash Y;\ \Gamma, Y \vdash X}{\Gamma, \Delta \vdash X}$ (правило сечения).

Правила, обозначенные в теореме, — элементарные следствия из определения понятия „вывод из гипотез“ и лишь фиксируют очевидное.

Отметим также очевидный факт, что мы можем произвольно вводить аксиомы в список Γ гипотез или выводить аксиомы из списка, не меняя сути. Первое вытекает из правила до-

*По определению в выводе каждая формула есть либо аксиома, либо гипотеза, либо выводится из предыдущих по правилу modus ponens, т.е. формулы в выводе, строго говоря, могут повторяться. Однако ясно, что повторения можно убрать, не нарушая связей в выводе.

бавления гипотез, а второе утверждение можно получить как формальное следствие правила сечения:

$$\begin{array}{ccc} & \Gamma \vdash X & \\ \swarrow & & \searrow \\ \Gamma, Y \vdash X & & \vdash Y \end{array}$$

Следующая теорема вводит так называемые ***логические правила естественного вывода***, описывающие некоторые общепринятые приемы проведения математических доказательств.

Теорема 3.3. Для любого списка формул Γ и любых формул X, Y, Z верны следующие утверждения:

1) $\dfrac{\Gamma, Y \vdash X}{\Gamma \vdash Y \to X}$ (введение импликации, теорема о дедукции);
2) $\dfrac{\Gamma \vdash X;\ \Gamma \vdash X \to Y}{\Gamma \vdash Y}$ (удаление импликации, modus ponens);
3) $\dfrac{\Gamma \vdash X;\ \Gamma \vdash Y}{\Gamma \vdash X \wedge Y}$ (введение конъюнкции);
4) $\dfrac{\Gamma, X, Y \vdash Z}{\Gamma, X \wedge Y \vdash Z}$ (удаление конъюнкции);
5) $\dfrac{\Gamma \vdash X}{\Gamma \vdash X \vee Y}$ и $\dfrac{\Gamma \vdash X}{\Gamma \vdash Y \vee X}$ (введение дизъюнкции);
6) $\dfrac{\Gamma, X \vdash Z;\ \Gamma, Y \vdash Z}{\Gamma, X \vee Y \vdash Z}$ (удаление конъюнкции, правило разбора случаев);
7) $\dfrac{\Gamma, X \vdash Y;\ \Gamma, X \vdash \neg Y}{\Gamma \vdash \neg X}$ (введение отрицания);
8) $\dfrac{\Gamma \vdash \neg\neg X}{\Gamma \vdash X}$ (удаление отрицания).

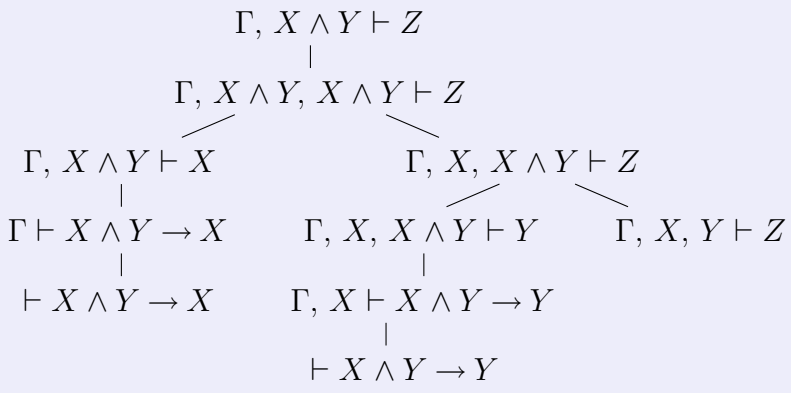
◀ Первые два правила — это теорема о дедукции и правило modus ponens в применении к выводу из гипотез. В самом деле, объединив выводы из Γ формул X и $X \to Y$, получим последовательность, в которой есть эти формулы. Применив modus ponens, получим Y, которую добавим в конец последовательности формул. Получим вывод Y из Γ.

Для доказательства правила 3 для произвольно заданных формул X и Y используем какую-либо аксиому T и аксиому $T \to X \to (T \to Y \to (T \to X \wedge Y))$ схемы 5. Из выводимости формул X и Y из списка Γ следует выводимость формул $T \to X$ и $T \to Y$. Дважды удаляя импликацию в аксиоме схемы 5, получим выводимость $T \to X \wedge Y$. Еще раз удаляя импликацию (T выводима), получаем выводимость $X \wedge Y$. Указанную последовательность шагов можно представить в виде дерева

$$\begin{array}{ccccccc}
 & & & \Gamma \vdash X \wedge Y & & & \\
 & & \swarrow & & \searrow & & \\
 & & \Gamma \vdash T \to X \wedge Y & & & \Gamma \vdash T & \\
 & \swarrow & & \searrow & \swarrow & & \searrow \\
 & \Gamma \vdash T \to Y \to (T \to X \wedge Y) & & & \Gamma \vdash T \to Y & & \vdash T \\
\swarrow & & \searrow & \swarrow & & \searrow & \\
\Gamma \vdash T \to X \to (T \to Y \to (T \to X \wedge Y)) & & & \Gamma \vdash T \to X & & \Gamma, T \vdash Y & \\
| & & & | & & | & \\
\vdash T \to X \to (T \to Y \to (T \to X \wedge Y)) & & & \Gamma, T \vdash X & & \Gamma \vdash Y & \\
 & & & | & & & \\
 & & & \Gamma \vdash X & & &
\end{array}$$

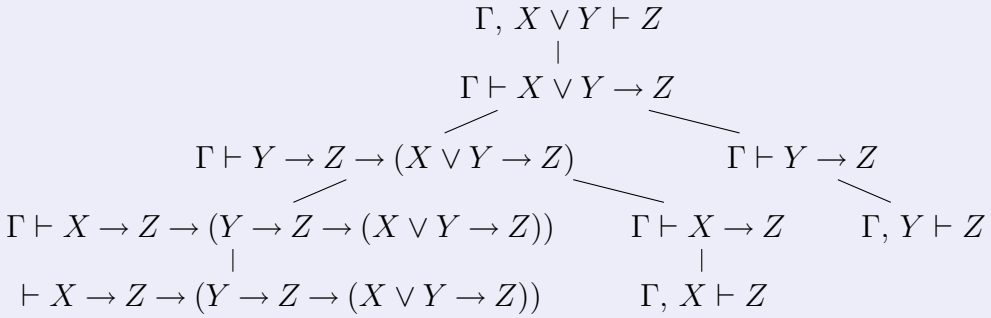
Правило 4 опирается на аксиомы $X \wedge Y \to X$ и $X \wedge Y \to Y$. Из этих аксиом по правилу modus ponens получаем выводимости $X \wedge Y \vdash X$ и $X \wedge Y \vdash Y$. Остается соединить два этих

вывода с выводом $\Gamma, X, Y \vdash Z$ для получения нужного вывода $\Gamma, X \wedge Y \vdash Z$. Соответствующее дерево вывода имеет следующий вид:

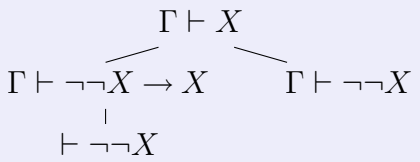


(здесь использовано упрощенное правило 2 вида $\frac{\Gamma \vdash X \to Y}{\Gamma, X \vdash Y}$). Аналогично доказывается правило 5, но с использованием аксиом $X \to X \vee Y$ и $X \to Y \vee X$, полученных по схемам 6 и 7.

Правило 6 опирается на аксиому $X \to Z \to (Y \to Z \to (X \vee Y \to Z))$, полученную по схеме 8 и выводится аналогично правилу 5:

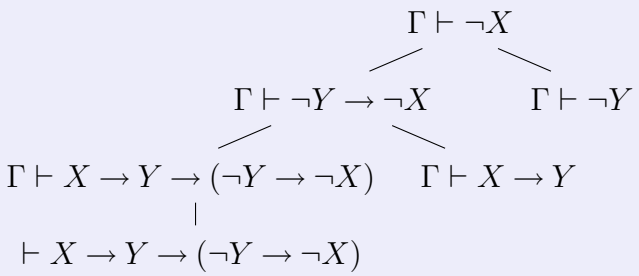


Правило 8 опирается на схему аксиом 10 и доказывается следующим деревом:



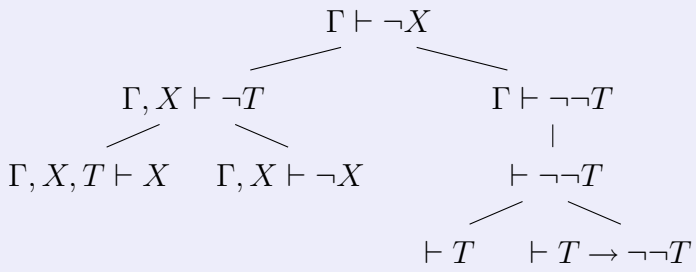
Отметим, что из схемы 11 вытекает правило, обратное к правилу 8: $\frac{\Gamma \vdash X}{\Gamma \vdash \neg\neg X}$.

Остановимся на правиле 7, которое вытекает из схем 9, 10, 11. Непосредственно из схемы аксиом 9 вытекает более слабое правило $\frac{\Gamma, X \vdash Y;\ \Gamma \vdash \neg Y}{\Gamma \vdash \neg X}$:



Чтобы получить нужное правило, необходимо в список Γ добавить аксиому X. Получим правило $\frac{\Gamma, X, X \vdash Y;\ \Gamma, X \vdash \neg Y}{\Gamma, X \vdash \neg X}$, или после сокращения одинаковых гипотез: $\frac{\Gamma, X \vdash Y;\ \Gamma, X \vdash \neg Y}{\Gamma, X \vdash \neg X}$. Остается обосновать правило $\frac{\Gamma, X \vdash \neg X}{\Gamma \vdash \neg X}$, для чего используется схема аксиом 11. Кроме того, как и в случае правила 5 используется какая-либо аксиома T. Эта аксиома вводится в список гипотез,

а затем дважды применяется слабый вариант доказываемого правила:



(3.1)

Теорема полностью доказана. ▶

Как видно по комментариям в формулировке теоремы, логические правила естественного вывода есть двух видов: правила введения, которые позволяют доказывать формулу с данной логической связкой, и правила удаления, которые показывают, как использовать эту формулу для доказательства других формул.

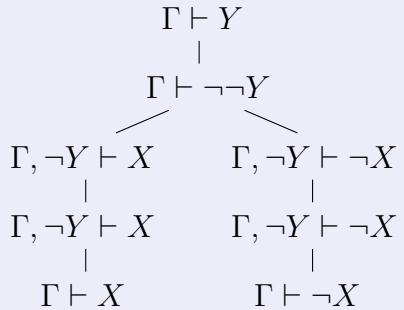
Правила естественного вывода (в первую очередь, теорема о дедукции) позволяют устанавливать выводимость в исчислении высказываний тех или иных формул, не строя конкретного вывода. Достаточно опираться на установленные правила и, как показано, в доказательстве теоремы, строить вывод, но уже других формул — секвенций. Отметим, что набор правил естественного вывода самодостаточен: в этих правилах учтены все схемы аксиом. При установлении истинности секвенций нет нужды прибегать непосредственно к аксиомам. Как станет ясно из дальнейшего изложения, выводимость формулы X равносильна тому, что секвенция $\vdash X$ выводима из правил естественного вывода.

На практике при установлении истинности конкретных секвенций удобно использовать дополнительные правила, вытекающие из правил естественного вывода. Укажем некоторые простейшие следствия правил естественного вывода.

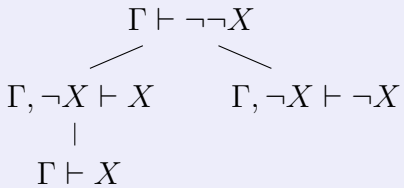
Пример 3.2. а. Правило $\dfrac{\Gamma \vdash X}{\Gamma \vdash Y \to X}$, называемое присоединением посылки, вытекает из правила введения импликации и правила добавления гипотез:

$$\begin{array}{c} \Gamma \vdash Y \to X \\ | \\ \Gamma, Y \vdash X \\ | \\ \Gamma \vdash X \end{array}$$

б. Правило $\dfrac{\Gamma \vdash X;\ \Gamma \vdash \neg X}{\Gamma \vdash Y}$ (закон противоречия) вытекает из правила введения отрицания добавлением гипотез:



в. Правило введения двойного отрицания $\dfrac{\Gamma \vdash X}{\Gamma \vdash \neg\neg X}$ может быть получено из схемы аксиом 11. Его можно также вывести из правил естественного вывода:



г. Полезно правило $\dfrac{\Gamma, X \vdash Y}{\Gamma, \neg Y \vdash \neg X}$ доказательства от противного, которого вытекает из правила введения отрицания:

$$\begin{array}{ccc} & \Gamma, \neg Y \vdash \neg X & \\ \swarrow & & \searrow \\ \Gamma, X, \neg Y \vdash Y & & \Gamma, X, \neg Y \vdash \neg Y \\ | & & \\ \Gamma, X \vdash Y & & \end{array}$$

д. Правила $\dfrac{\Gamma \vdash X \wedge Y}{\Gamma \vdash X}$ и $\dfrac{\Gamma \vdash X \wedge Y}{\Gamma \vdash Y}$ удаления конъюнкции справа вытекают из правила удаления конъюнкции (но также могут быть получены из схем аксиом 6 и 7). Например, первое из этих правил получается применением правила сечения и правила удаления конъюнкции:

$$\begin{array}{ccc} & \Gamma \vdash X & \\ \swarrow & & \searrow \\ \Gamma, X \wedge Y \vdash X & & \Gamma \vdash X \wedge Y \\ | & & \\ \Gamma, X, Y \vdash X & & \end{array}$$

Второе правило удаления конъюнкции справа можно получить аналогично. #

Правила естественного вывода и их следствия позволяют доказать выводимость широкого круга секвенций. Однако отметим, что наиболее тяжело устанавливаются выводимости вида $\Gamma \vdash X \vee Y$. Правило введения дизъюнкции сводит задачу к выводу более сильного утверждения $\Gamma \vdash X$ или $\Gamma \vdash Y$.

Пример 3.3. На практике для доказательства утверждения вида $X \vee Y$ можно рассуждать так: „пусть X не верно, т.е. истинно $\neg X$. Докажем, что тогда истинно Y.“ Этому варианту рассуждений соответствуют правила $\dfrac{\Gamma, \neg X \vdash Y}{\gamma \vdash X \vee Y}$ и $\dfrac{\Gamma, \neg Y \vdash X}{\gamma \vdash X \vee Y}$, которые можно получить и в исчислении высказываний. Для этого достаточно использовать правило введения отрицания и правило доказательства от противного:

$$\begin{array}{ccc} & \Gamma \vdash X \vee Y & \\ & | & \\ & \Gamma \vdash \neg\neg(X \vee Y) & \\ \swarrow & & \searrow \\ \Gamma, \neg(X \vee Y) \vdash X & & \Gamma, \neg(X \vee Y) \vdash \neg X \\ | & & | \\ \Gamma, \neg(X \vee Y) \vdash \neg\neg X & & \Gamma, X \vdash X \vee Y \\ | & & | \\ \Gamma, \neg X \vdash X \vee Y & & \Gamma, X \vdash X \\ | & & \\ \Gamma, \neg X \vdash Y & & \end{array}$$

Это правило назовем ***обобщенным правилом введения дизъюнкции***.

3.4. Глобальные свойства теории N

Выводимые формулы и тавтологии. Непротиворечивость теории $\boldsymbol{N}$. Полнота теории $\boldsymbol{N}$. Разрешимость теории $\boldsymbol{N}$. Независимость аксиом в теории $\boldsymbol{N}$.

Остановимся на описании множества выводимых формул в теории $\boldsymbol{N}$. Нетрудно убедиться в том, что все аксиомы этой теории являются тавтологиями . Например, любая аксиома, полученная по схеме 1, может быть получена подстановкой формул в тавтологию $x \to (y \to x)$, а потому сама является тавтологией (следствие 2.1).

Если формулы X и $X \to Y$ являются тавтологиями, то правило modus ponens приводит к формуле Y, также являющейся тавтологией (теорема 2.1). Это означает, что в любом выводе все формулы являются тавтологиями, поскольку либо являются аксиомами, либо получены из тавтологий по правилу modus ponens (аккуратное доказательство может быть проведено методом математической индукции по длине вывода). Можно также утверждать, что если в списке Γ все формулы — тавтологии, то и все формулы, выводимые из гипотез Γ, также будут тавтологиями.

Теорема 3.4. Любая формула, выводимая в теории $\boldsymbol{N}$, является тавтологией. #

Итак, множество формул, выводимых в теории $\boldsymbol{N}$ содержится в множестве всех тавтологий. На самом деле оба множества совпадают. Чтобы это доказать, установим одно вспомогательное утверждение. Как и в случае булевых функций для произвольной пропозициональной переменной x введем обозначение x^σ, $\sigma \in \mathbb{B}$, полагая, что $x^1 = x$ и $x^0 = \neg x$.

Теорема 3.5. Пусть Y — формула, построенная на переменных $x_1, \ldots, x_n$, $f(\xi_1, \ldots, \xi_n)$ — истинностная функция формулы Y, $\alpha_i \in \mathbb{B}$, $i = \overline{1, n}$, — набор истинностных значений для переменных $x_1, \ldots, x_n$. Если $f(\alpha_1, \ldots, \alpha_n) = 1$, то имеет место выводимость $x_1^{\alpha_1}, \ldots, x_n^{\alpha_n} \vdash Y$. Если $f(\alpha_1, \ldots, \alpha_n) = 0$, то имеет место выводимость $x_1^{\alpha_1}, \ldots, x_n^{\alpha_n} \vdash \neg Y$.

◀ Доказательство строится индукцией по построению формулы. В этом случае базис индукции относится к элементарным формулам, т.е. к переменным. Пусть $Y = x$ и задано истинностное значение α. При $\alpha = 1$ имеем $x^\alpha = x$, $Y = x$, и севенция $x^\alpha \vdash Y$ истинна, поскольку и левая, и правая части секвенции — одна и та же формула. При $\alpha = 0$ имеем $x^\alpha = \neg x$, $\neg Y = \neg x$, и истинной является секвенция $x^\alpha \vdash \neg Y$.

Чтобы доказать шаг индукции, необходимо в предположении, что утверждение верно для формул Z и W, установить его истинность для формул $(Z \vee W)$, $(Z \wedge W)$, $(Z \to W)$ и $(\neg Z)$. Пусть $z_1, \ldots, z_n$ — общий список переменных формул Z и W.

Рассмотрим случай $Y = Z \wedge W$. Для заданного набора α значений $\alpha_1, \ldots, \alpha_n$ введем обозначение $\Gamma = x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}$. Предположим, что на наборе α истинностные функции f_Z и f_W принимают значение 1. Тогда по индуктивному предположению имеем $\Gamma \vdash Z$ и $\Gamma \vdash W$, откуда по правилу введения конъюнкции получаем выводимость $\Gamma \vdash Y$, так как $Y = Z \wedge W$. Пусть одна из истинностных функций, например f_Z, принимает значение 0. Тогда имеет место выводимость $\Gamma \vdash \neg Z$. Используя правила естественного вывода, получаем

$$\begin{array}{ccc} & \Gamma \vdash \neg(Z \wedge W) & \\ \swarrow & & \searrow \\ \Gamma, Z \wedge W \vdash \neg Z & & \Gamma, Z \wedge W \vdash Z \\ | & & | \\ \Gamma, Z, W \vdash \neg Z & & \Gamma, Z, W \vdash Z \\ | & & \\ \Gamma \vdash \neg Z & & \end{array}$$

В результате получена выводимость $\Gamma \vdash \neg Y$, поскольку $\neg Y = \neg(Z \wedge W)$.

Рассмотрим случай $Y = Z \vee W$. Опять выберем набор значений α и введем, как и выше, обозначение Γ. Если одна из функций f_Z f_W на наборе α имеет значение 1, например $f_Z(\alpha) = 1$, то по индуктивному предположению имеет место выводимость $\Gamma \vdash Z$. Отсюда сразу получаем выводимость $\Gamma \vdash Z \vee W = Y$ в силу правила введения дизъюнкции. Осталось рассмотреть вариант $f_Z(\alpha) = f_W(\alpha) = 0$. В этом варианте $f_Y(\alpha) = f_Z(\alpha) \vee f_W(\alpha) = 0$ и, значит, нужно установить выводимость $\Gamma \vdash \neg Y$ исходя из индуктивных предположений $\Gamma \vdash \neg Z$ и $\Gamma \vdash \neg W$. Имеем:

$$\begin{array}{ccccc} & & \Gamma \vdash \neg(Z \vee W) & & \\ & \swarrow & & \searrow & \\ & \Gamma, Z \vee W \vdash U & & \Gamma, Z \vee W \vdash \neg U & \\ \swarrow & & \searrow & \swarrow & \searrow \\ \Gamma, Z \vdash U & & \Gamma, W \vdash U & \Gamma, Z \vdash \neg U & \Gamma, W \vdash \neg U \end{array}$$

В результате требуемая секвенция оказадась сведена к четырем однотипным севенциям, в которых выбор формулы U (соответственно $\neg U$) не имеет значения. Истинность всех четырех севенций устанавливается по единому сценарию с помощью закона противоречия. Рассмотрим, например, первую из них:

$$\begin{array}{ccc} & \Gamma, Z \vdash U & \\ \swarrow & & \searrow \\ \Gamma, Z \vdash Z & & \Gamma, Z \vdash \neg Z \\ & & \downarrow \\ & & \Gamma \vdash \neg Z \end{array}$$

Рассмотрим случай $Y = Z \to W$. Выделим вариант $f_W(\alpha) = 1$. В этом варианте $f_Y(\alpha) = 1$ и нужно установить выводимость $\Gamma \vdash Z \to W$ исходя из $\Gamma \vdash W$. Это верно в силу правила присоединения посылки. Пусть $f_Z(\alpha) = 1$, $f_W(\alpha) = 0$. Тогда $f_Y(\alpha) = 0$, и нужно установить выводимость $\Gamma \vdash \neg(Z \to W)$ исходя из $\Gamma \vdash Z$ и $\Gamma \vdash \neg W$. Это можно сделать следующим образом:

$$\begin{array}{ccccc} & & \Gamma \vdash \neg(Z \to W) & & \\ & \swarrow & & \searrow & \\ & \Gamma, Z \to W \vdash W & & & \Gamma, Z \to W \vdash \neg W \\ & \swarrow & \searrow & & \downarrow \\ \Gamma, Z \to W \vdash Z & & \Gamma, Z \to W \vdash Z \to W & & \Gamma \vdash \neg W \\ \downarrow & & & & \\ \Gamma \vdash Z & & & & \end{array}$$

Пусть $f_Z(\alpha) = 0$, $f_W(\alpha) = 0$. Тогда $f_Y(\alpha) = 1$, и требуется установить $\Gamma \vdash Z \to W$ исходя из $\Gamma \vdash \neg Z$ и $\Gamma \vdash \neg W$. Из первой секвенции добавлением гипотезы получаем $\Gamma, Z \vdash \neg Z$, что с очевидной секвенцией $\Gamma, Z \vdash Z$ по закону противоречия приводит к выводимости $\Gamma, Z \vdash W$ и далее по правилу введения импликации к $\Gamma \vdash Z \to W$.

Рассмотрим случай $Y = \neg Z$. Если $f_Z(\alpha) = 1$, то $f_Y(\alpha) = 0$ и $\Gamma \vdash Z$. Введением отрицания получаем $\Gamma \vdash \neg\neg Z = \neg Y$. Аналогично при $f_Z(\alpha) = 0$ имеем $f_Y(\alpha) = 1$ и $\Gamma \vdash \neg Z$, что равносильно $\Gamma \vdash Y$. ►

Теорема 3.6. Любая тавтология Y выводима в теории $\boldsymbol{N}$.

◄ Пусть формула Y построена из переменных $x_1, \ldots, x_n$. Поскольку Y — тавтология, ее истинностная функция на любом булевом векторе α принимает значение 1. Значит, для любых значений $\alpha_1, \ldots, \alpha_n$ имеет место выводимость $x_1^{\alpha_1}, \ldots, x_n^{\alpha_n} \vdash Y$.

Обозначим $\Gamma_2 = x_2^{\alpha_1}, \ldots, x_n^{\alpha_n}$. Имеют место выводимости $x_1, \Gamma_2 \vdash Y$ и $\neg x_1, \Gamma_2 \vdash Y$. По правилу удаления конъюнкции находим $x_1 \vee \neg x_1, \Gamma_2 \vdash Y$. Используя выводимость $\vdash x_1 \vee \neg x_1$, по правилу сечения получаем $\Gamma_2 \vdash Y$.

Повторяя рассуждения, последовательно убираем из списка гипотез переменные x_2, x_3 и т.д. В конечном счете получим выводимость $\vdash Y$, что равносильно утверждению теоремы. ►

Полученное описание выводимых в теории $\boldsymbol{N}$ формул позволяет установить такие свойства этой теории, как непротиворечивость, полноту и разрешимость. Установление этих свойств — цель, преследуемая при формализации любой теории.

Под непротиворечивостью формальной теории понимают отсутствие в этой теории такой формулы X, для которой выводимыми являются и сама формула X, и ее отрицание $\neg X$. Иногда дают такое определение непротиворечивости теории: теория непротиворечива, если в ней существует невыводимая формула. Формально утверждение о выводимости любой формулы сильнее, чем утверждение о выводимости какой либо пары X и $\neg X$ (из первого утверждения очевидно вытекает второе). Однако в данном случае оба подхода дают одно и то же: если в теории выводимы и X, и $\neg X$, то по закону противоречия выводима любая формула.

Теорема 3.7. Теория $\boldsymbol{N}$ является непротиворечивой.

◄ Выводимыми в теории $\boldsymbol{N}$ являются только тавтологии. Если формула X выводима в теории $\boldsymbol{N}$, то X — тавтология. Но тогда $\neg X$ является противоречием и не относится к числу выводимых формул. ►

Под полнотой формальной теории понимают свойство, согласно которому не существует такой невыводимой формулы X, добавление которой к аксиомам приводит к непротиворечивой теории. Другими словами, в такой теории Для каждой формулы X выводима либо сама формула, либо ее отрицание. Наша теория не полна в этом смысле. Действительно, если, добавив в качестве аксиомы некоторую формулу X, мы найдем вывод формулы $\neg X$, то, с иной точки зрения, мы получим вывод $\neg X$ из гипотезы X в теории $\boldsymbol{N}$, т.е. $X \vdash \neg X$. Значит, согласно дереву (3.1), имеет место выводимость $\vdash \neg X$. Однако, как доказано, такое возможно только, если X — противоречие. Следовательно, добавление любой формулы, не являющейся тавтологией или противоречием, дает непротиворечивую теорию.

Впрочем, если в теорию ввести дополнительную схему аксиом, построенную на формуле, не являющейся тавтологией, то подставляя в схему вместо символов подстановки определенные формулы, можно получить противоречие. Это противоречие, с одной стороны, является аксиомой, поскольку получено из схемы аксиом, а с другой стороны, его отрицание есть тавтология, выводимая в теории $\boldsymbol{N}$. Пусть например, схема аксиом построена на формуле, представленной истинностной функцией $f(x_1, \ldots, x_n)$. Так как формула — не тавтология, существует набор значений α_1, $\ldots$, α_n аргументов функции, на котором она принимает значение 0. Каждую переменную x_j заменим формулой $x_j \vee \neg x_j$, если $\alpha_j = 1$, и формулой $x_j \wedge \neg x_j$. Полученная формула, построенная по схеме аксиом, будет противоречием.

При построении исчисления высказываний (и далее исчисления предикатов) различают полноту в широком смысле, означающую сказанным выше, и полноту в узком смысле, означающую, что выводимыми являются все тавтологии. В узком смысле теория $\boldsymbol{N}$ полна. Впрочем, отсутствие полноты в широком смысле легко устраняется простой модификацией формальной теории: достаточно вместо каждой схемы аксиом записать обычную аксиому, заменив места подстановки переменными и добавить еще одно правило вывода, заключающееся в подстановке вместо переменных произвольных формул. Отметим, что свойством полноты обладают очень простые математические теории: согласно теореме Геделя о неполноте любая достаточно содержательная теория не является полной, поскольку в рамках такой теории удается построить такую формулу Z, что ни эта формула, ни ее отрицание не являются выводимыми.

Под разрешимостью теории понимают наличие алгоритма, который для любой формулы позволяет установить за конечное число элементарных операций, выводима эта формула или нет. В данном случае в качестве такого алгоритма может рассматривать процедуру построения истинностной функции и проверки ее на наличие значений 0. Отсутствие таких значений означает, что функция является постоянной, а формула — тавтологией, т.е. выводимой в теории $\boldsymbol{N}$. Указанная процедура выполняется за конечное число алгебраических операций и дает однозначный ответ, выводима формула или нет. В этом смысле теория $\boldsymbol{N}$ является разрешимой теорией.

Еще один вопрос, связанный с построением формальной теории — независимость аксиом. Аксиома формальной теории не зависит от остальных аксиом этой теории, если эта аксиома не является выводимой формулой в теории, которая получается из исходной удалением указанной аксиомы. Если ни одна из аксиом формальной теории не является логическим следствием остальных, то говорят о ***независимой системе аксиом***.

Система аксиом теории $\boldsymbol{N}$ является независимой в том смысле, что удалив одну из схем аксиом теории, мы не сможем вывести ни одну из формул, получаемой в рамках этой схемы аксиом. Как можно доказать утверждения подобного рода. Напомним, что в формальной теории формулы теряют всякий содержательный смысл; например, знак импликации может означать какую угодно бинарную операцию на множестве возможных значений переменных, а сами переменные могут быть связаны с любыми математическими объектами. Придание смысла символам формальной теории называют интерпретацией этой теории. В данном случае множество всех высказываний как область изменения переменных и естественный логический смысл символов операций — это одна из возможных интерпретаций исчисления $\boldsymbol{N}$.

Интерпретация формальной теории строится в рамках какой-либо другой математической

теории. Надежность интепретации определяется надежностью той теории, в которой интепретируется исчисление. Так, суждения о непротиворечивости, полноте и разрешимости исчисления $\boldsymbol{N}$ получены в рамках так называемой теории булевых функций, т.е. теории функций с областью изменения переменных и функций 0 и 1. В этом смысле подобные суждения носят относительный характер. В математике считают абсолютно надежными теории, связанные с конечными множествами. В этом смысле утверждения о непротиворечивости, полноте и разрешимости исчисления $\boldsymbol{N}$ являются абсолютными. Утверждение о независимости системы аксиом тоже будет абсолютным, если оно будет получено на базе какой-либо конечной интерпретации.

Теорема 3.8. Схемы аксиом исчисления $\boldsymbol{N}$ не зависят друг от друга.

◀ Наиболее просто доказать независимость схем, начиная с 3-й, так как все эти схемы используют, кроме импликации, еще одну операцию, которая участвует лишь в двух других схемах аксиом. Можно ограничиться двухэлементной областью изменения переменных и стандартной интепретацией операций, кроме одной. В табл. 3.1 приведены схемы аксиом и новая интерпретация одной из операций. При этой интерпретации указанная схема аксиом при некоторых значениях входящих в нее формул принимает значение 0, в то время как остальные аксиомы имеют постоянное истинностное значение 1. Изменение интерпретации не затрагивает правила modus ponens, так что при удалении указанной схемы аксиом мы получаем теорию, в которой все выводимые формулы — тавтологии, но некоторые формулы, получаемые из указанной схемы, выводимыми не являются. Это и означает, что указанная схема аксиом не зависит от остальных.

Независимость первых двух схем доказать сложнее, поскольку они касаются импликации, затрагивающей все схемы. Ответ можно найти, рассмотрев в качестве области изменения переменных множество из трех целых чисел $\{0,\ 1,\ 2\}$ и выбрав следующие интерпретации операций: $x \wedge y = \min\{x,\ y\}$, $x \vee y = \max\{x,\ y\}$, $\neg x = 2 - x$. Для импликации потребуем выполнения условия, что $x \to y = n$ тогда и только тогда, когда $x \leqslant y$. При поставленных условиях схемы 3, 4, 6, 7, 10, 11 будут давать формулы с тождественным значением n, а в силу условия, наложенного на импликацию, значение Y будет тождественное n при $X \equiv n$ и $X \to Y \equiv n$. Недоопределенность импликации можно использовать для получения нужных значений первых двух схем аксиом и для обеспечения тождественного значения n схем 5, 8, 9.

Положим $x \to y = 0$ при $x > y$. Тогда схема $X \to (Y \to X)$ будет иметь значение 0 при $X = 1$, $Y = 2$, в то время как остальные аксиомы будут давать тождественное значение 2. Действительно, в схеме 2 исключаем случай $X > Y$ или $X \leqslant Z$, так как тогда она имеет вид $0 \to W$ или $W \to 1$. Значит, $Z < X \leqslant Y$ и схема имеет вид $2 \to (X \to 0 \to 0)$. Но $X > Z \geqslant 0$, значит, $X \to 0 = 0$ и $2 \to (X \to 0 \to 0) = 2 \to (0 \to 0) = 2 \to 2 = 2$. В схеме 5 исключаем случай $X > Y$ или $X > Z$, когда она имеет значение 2. Но тогда $X \leqslant \min\{Y,\ Z\}$ и $X \to Y \vee Z = 2$. Поэтому $X \to Y \to (X \to Z \to (X \to Y \wedge Z)) = X \to Y \to (X \to Z \to 2) = 2$. В схеме 8 исключаем случай $X > Z$ или $Y > Z$. Тогда $\max\{X,\ Y\} \leqslant Z$, $X \vee Y \to Z = 2$ и схема 8 имеет тождественное значение 2. В схеме 9 исключаем случай $X > Y$. Но тогда $X \leqslant Y$, $\neg Y \leqslant \neg X$ и формула имеет вид $2 \to 2$.

Положим $x \to y = 1$ при $x > y$. Тогда схема 8 будет иметь значение 1, например, при $X = 1$, $Y = 2$, $Z = 0$. В схеме 1 исключаем случай $Y \leqslant X$. Тогда $X < Y \leqslant 2$, а значит, $X \leqslant 1$ и $X \to (Y \to X) = X \to 1 = 2$. Схемы 5, 8, 9 проверяются так же, как выше. ▶

Таблица 3.1

№	Схема аксиом	Интерпретируемая операция	№	Схема аксиом	Интерпретируемая операция
3	$X \wedge Y \to X$	$x \wedge y = y$	8	$X \to Z \to (Y \to Z \to (X \vee Y \to Z))$	$x \vee y = 1$
4	$X \wedge Y \to Y$	$x \wedge y = x$	9	$X \to Y \to (\neg Y \to \neg X)$	$\neg x = x$
5	$X \to Y \to (X \to Z \to (X \to Y \wedge Z))$	$x \wedge y = 0$	10	$\neg\neg X \to X$	$\neg x = 1$
6	$X \to Y \vee Z$	$x \vee y = z$	11	$X \to \neg\neg X$	$\neg x = 0$
7	$Y \to Y \vee Z$	$x \vee y = y$			

ОГЛАВЛЕНИЕ

**Московский государственный технический университет
имени Н.Э. Баумана**

Факультет «Фундаментальные науки»
Кафедра «Математическое моделирование»

А.Н. Канатников

ДИСКРЕТНАЯ МАТЕМАТИКА

Конспект лекций

Для студентов специальности
«Прикладная математика»

Москва
2006

4. АЛГЕБРА ПРЕДИКАТОВ

4.1. Предикаты и кванторы

Ограниченность исчисления высказываний. Переменные высказывания. Субъект и предикат переменного высказывания. Понятие предиката в математической логике. Арность предиката. Контуры будущей теории. Кванторы как символы агрегирования. Выражения двух видов.

Построенное исчисление высказываний в реальности оказывается довольно бедной теорией. Оно позволяет строить и анализировать различные комбинации высказываний. Однако в математике встречаются предложения, которые, вообще говоря, не являются высказываниями. Например, нельзя сказать, истинно ли утверждение „число x положительное“. Ответ зависит от того, какое это число. Здесь мы сталкиваемся с уточнением того, что такое суждение. В каждом суждении присутствуют два компонента — субъект и предикат. Субъект — это понятие, отражающее предмет, о котором идет речь. В предложении субъект связывается с подлежащим. Предикат же можно уподобить сказуемому, поскольку этот структурный элемент выражает свойства, приписываемые предметам или отрицаемые у них. В суждениях „Роза красная“, „Гитара семиструнная“ субъектами являются „роза“ и „гитара“, а предикатами — „красная“ (все то, что называется красным) и „семиструнная“ (все то, что называется семиструнным).

В математике этой трактовке суждения отвечает „высказывание с параметрами“, т.е. суждение, которое становится высказыванием при конкретизации параметров. Такие суждения называют предикатами. С математической точки зрения предикат можно интерпретировать как некое отображение, которое каждому набору входящих в него параметров ставит в соответствие высказывание. Количество участвующих в предикате параметров называется его ***арностью***.

Понятие предиката не вкладывается в язык исчисления высказываний. Требуется построение более широкой теории, требующей и расширения языка теории. Что должен охватывать такой язык помимо высказываний и логических связок?

Во-первых, если в формулах алгебры высказываний встречаются переменные, областью действия которых является множество высказываний, то в формулах любой математической теории есть переменные, область действия которых связана не с высказываниями, а с объектами, изучаемыми в этой теории. В математическом анализе переменные числовые, в геометрии переменные обозначают точки, прямые, плоскости. Такие переменные называют ***предметными***, а область их действия — ***предметной областью***. В теории могут участвовать предметные переменные разного рода, поскольку описывают объекты разных классов (например, точки и прямые в геометрии).

Во-вторых, в ряде математических теорий есть установленные символы для обозначения некоторых стандартных объектов (например, число π). В отличие от переменных эти символы имеют конкретное значение. Их называют константами.

В третьих, в каждой математической теории используются функции, причем разные, к которым можно отнести и операции над объектами математической теории. Функции можно рассматривать двояко: как отображение и как связь между переменными. С точки зрения формальной записи функция — это символ, который исходя из некоторой совокупности переменных формирует новую переменную (точнее, новый объект, имеющий неопределенное значение).

Наконец, в четвертых, в этом языке должны быть отражены и такие символы, как кванторы. Кванторы — специальные символы (или фразеологические штампы), которые позволяют из не-

определенного предложения построить правильное высказывание, истинность которого устанавливается однозначно. Кванторы можно рассматривать как символы агрегирования, которые из совокупности возможных значений формируют одно значение. Таковы символы предела, интеграла, суммы. В логике используют два квантора: всеобщности $\forall$ и существования $\exists$. Запись $\forall x P(x)$ формирует высказывание, являющееся истинным, если предикат $P(x)$ является истинными для любого возможного значения предметной переменной x. Запись $\exists x P(x)$ формирует высказывание, истинное, если предикат $P(x)$ имеет значение истины хотя бы для одного возможного значения предметной переменной x.

Введение в язык символов агрегирования приводит к тому, что вхождение переменных в формулы может играть две разных роли. В формуле $x + y$ переменным x и y можно присвоить конкретные значения, в результате чего конкретное значение получит все выражение. В формуле $\forall x P(x)$ переменной x нельзя присвоить конкретное значение, поскольку формула подразумевает использование целого множества значений переменной, а не отдельно взятого значения (поэтому и квантор — символ агрегирования). В первом случае говорят о ***свободном вхождении переменной***, а во втором — о ***связанном вхождении***. Одна и та же переменная в выражение может входить и свободно, и связанно. Например, в выражении $n + \sum_{n=1}^{3} a_n$ переменная n входит свободно (первое вхождение) и связанно (второе и третье).

В дальнейшем изложении мы не будем использовать предметных символов агрегирования, ограничившись только двумя кванторами, так как предметные символы агрегирования усложняют язык, но не добавляют чего-то существенного в формальную теорию как таковую. Отметим, что „за бортом“ остаются обозначения множеств и связи множеств и элементов. Если в формальной теории нет средств для обозначения множеств, ее называют теорией 1-го порядка. При введении множеств отдельных элементов получаем теорию 2-го порядка, множества множеств приводят к теории 3-го порядка. И так далее.

Отметим, что новая теория будет содержать выражения двух видов. Первые не означают какое-то высказывание, они описывают лишь действия, выполняемые над объектами теории. Например, выражение $x + y$ играет роль новой предметной переменной, значения которой однозначно определяются значениями x и y. Напротив, выражение $x + y = 0$ выражает суждение, которое может быть истинным или ложным при заданных значениях неизвестных. Разделяя два типа выражений, первые будем называть ***термами***, а вторые ***формулами***. Таким образом, значением терма является некоторый объект теории (например, число), в то время как значением формулы является высказывание.

4.2. Логико-математические языки

Язык как совокупность четырех множеств (S, C, F, P). Предметные переменные, константы и их сортность. Функциональные и предикатные символы, их тип. Предикатные символы арности 0 как пропозициональные переменные. Индуктивное построение термов и формул. Функциональная сложность терма, логическая сложность формулы. Выражения. Правильные формулы: $A(x)$, $B(x, y)$, C, $(\forall x A(x))$. Комментарии по вариантам построения логико-математических языков. Односортные языки. Свободные и связанные вхождения переменных. Предложения (замкнутые термы и формулы). Соглашения о сокращенной записи формул.

Под логико-математическим языком мы будем понимать совокупность $(V,\ C,\ F,\ P)$ из четырех не более чем счетных множеств, элементы которых называют ***символами***. Первое множество — это множество V ***предметных переменных***. Предметные переменные могут иметь разные ***сорта***. Конкретно мы считаем, что есть еще пятое множество S — ***множество сортов*** и задано отображение $\mathrm{sort}_V\colon V \to S$. Значение отображения sort_V на переменной $x \in V$ и есть сорт этой переменной. Каждому сорту $\pi \in S$ соответствует множество V_π предметных переменных этого сорта, являющееся полным прообразом $\mathrm{sort}_V^{-1}(\pi)$ элемента $\pi \in S$.

Элементы множества C называются ***константами***. Каждая константа относится к определенному сорту $\pi \in S$, т.е. задано отображение $\mathrm{sort}_C\colon C \to S$ множества констант в множество сортов.

Элементы множества F — это ***функциональные символы***, каждый из которых характеризуется своим типом, определяемым кортежем $(\pi_0, \pi_1, \ldots, \pi_n) \in S^{n+1}$ сортов (целое число $n > 0$ называется ***арностью функционального символа***).

Элементы множества P — это ***предикатные символы***, каждый из которых характеризуется своим типом, определяемым кортежем сортов $(\pi_1, \ldots, \pi_n) \in S^n$ (и в этом случае $n \geqslant 0$ — ***арность предикатного символа***).

Названия элементов четырех множеств указывают на роль, которые они будут играть в содержательной интерпретации языка. Однако этот содержательный смысл не играет роли в формальной теории. Главное, каким образом из символов составляются правильные слова языка.

В логико-математическом языке два вида правильных слов. Это термы и формулы. Начнем с индуктивного определения термов.

База индукции. Любая переменная или константа сорта π является термом сорта π (элементарные термы).

Шаг индукции. Если $t_1, t_2, \ldots, t_n$ — термы сортов $\pi_1, \ldots, \pi_n$ соответственно, f — функциональный символ типа $(\pi_0, \pi_1, \ldots, \pi_n)$, то $f(t_1, t_2, \ldots, t_n)$ — терм сорта π_0.

Из этого определения нетрудно понять, что термы служат для записи выражений в предметной области (например, многочленов в полях).

Функциональной сложностью терма называется количество функциональных символов в нем. Индуктивно эту характеристику можно определить так: сложность элементарного терма равна 0; сложность терма $f(t_1, t_2, \ldots, t_n)$ равна сумме сложностей аргументов плюс один, т.е.

$$|f(t_1, t_2, \ldots, t_n)| = |t_1| + t_2| + \ldots + |t_n| + 1,$$

где знак модуля обозначает функциональную сложность терма.

Подобное индуктивное определение может использоваться не только для определения понятий, но и для доказательства различных утверждений. Например, пусть все элементарные термы обладают свойством P; из того, что термы $t_1, t_2, \ldots, t_n$ обладают свойством P, следует, что терм $f(t_1, t_2, \ldots, t_n)$ также обладает свойством P. Тогда все термы обладают свойством P.

Определим теперь формулы.

База индукции. Если $t_1, t_2, \ldots, t_n$ — термы сортов $\pi_1, \ldots, \pi_n$ соответственно, p — предикатный символ типа $(\pi_1, \ldots, \pi_n)$, то $p(t_1, t_2, \ldots, t_n)$ — формула (такую формулу будем называть элементарной).

Шаг индукции. Если X_1, X_2 — формулы, то $(X_1 \to X_2)$, $(X_1 \wedge X_2)$, $(X_1 \vee X_2)$, $\neg X_1$ — формулы. Если X — формула, x — предметная переменная, то $\forall x X$ и $\exists x X$ — формулы.

В то время как термы представляют собой записи последовательностей операций над объектами предметной области, формулы обозначают те или иные утверждения о свойствах этих объектов. Совокупность всех термов в рассматриваемом языке обозначим Tm, совокупность всех формул — Fm, а совокупность и тех, и других — Expr. Элементы Expr будем называть ***выражениями***.

Отметим, что допускаются предикатные символы арности 0, которые являются элементарными формулами. Множество таких символов в сочетании с логическими связками образует подмножество языка, эквивалентное языку алгебры высказываний. Поэтому предикатные символы арности 0 можно рассматривать как пропозициональные переменные.

Для формул может быть введена характеристика, называемая ***логической сложностью***, аналогичная функциональной сложности термов, но определяемая количеством предикатных символов в формуле.

Замечание. 1. На самом деле понятие логико-математического языка можно вводить различными способами, не меняя содержательной сути. Так, предикатные символы можно рассматривать как особый род функциональных символов, для которых сорт π_0 есть „высказывание“. Аналогично можно было бы отказаться от переменных языка, рассматривая их как функциональные символы арности 0. В результате дело можно было бы свести к двум множествам: C и F.

Отметим также, что множество V на самом деле можно трактовать как семейство множеств, в котором каждое множество — это множество переменных одного сорта. Чтобы обозначить сорт переменной, достаточно указать множество, которому эта переменная принадлежит.

Выбор конкретного варианта определения диктуется субъективными факторами, а также тем, что должно присутствовать нечто, трактуемое как сорт высказываний.

2. Совокупность всех переменных, констант, функциональных и предикатных символов плюс логические связки, кванторы, скобки и запятая составляет алфавит логико-математического языка. В данном случае алфавит оказывается счетным множеством. Однако нетрудно дело свести к конечному алфавиту, если считать, что элементы логико-математического языка на самом деле являются словами другого языка (языка нижнего уровня). Так, для всех натуральных чисел можно получить обозначения с помощью двух символов 0 и 1, используя позиционную форму записи. Однако этот механизм находится за пределами интересов математической логики. Поэтому мы и считаем идентификаторы переменных, функций и т.п. простейшими неделимыми элементами языка — символами. #

Среди логико-математических языков выделим ***односортные языки***, в которых все предметные переменные имеют один сорт. Таков язык, например, арифметики. Напротив, язык линейной алгебры является двусортным (числа и векторы).

В формулах $\forall x\, X$ или $\exists x\, X$ первый символ называется ***квантором*** (далее, если несущественно, какой именно квантор, мы будем использовать обозначение ∇). Комбинация из квантора и переменной называется ***кванторной приставкой***, а формула X, следующая за кванторной приставкой — ***областью действия квантора***. Допускается, что переменная кванторной приставки (***кванторная переменная***) вообще не имеет вхождений в области действия этой приставки.

Появление кванторов приводит к тому, что предметные переменные формулы по своему положению могут иметь разные свойства. В языке высказываний была введена такая операция, как подстановка вместо переменных других переменных или формул. Однако при наличии кванторов произвольная подстановка может привести к неправильной формуле. Например, В формуле $\forall x\,(x > 0)$ нельзя переменную x заменить произвольной формулой: слово $\forall(x+1)\,((x+1) > 0)$ не является формулой, потому что по индуктивному правилу образования формул сразу за квантором должна следовать переменная, а не служебный символ или константа. Содержательный смысл формулы при такой подстановке также теряется.

Будем говорить, что вхождение предметной переменной x в формулу X является ***связанным***, если оно есть часть кванторной приставки или находится в области действия кванторной приставки с той же переменной. В противном случае вхождение переменной называется ***свободным***. В терме все вхождения каждой переменной свободные.

В одной и той же формуле переменная x может иметь и свободные, и связанные вхождения. Например, в формуле $p_1(x, (\forall x\, p_2(x)))$ три вхождения переменной x, причем первое вхождение свободное, а два других связанные. Если переменная x имеет свободные вхождения в формулу X, то ее называют ***свободной переменной*** этой формулы. Отметим, что определенное вхождение переменной может быть связанным в формуле и свободным в некоторой подформуле. Например, предикатный символ, в один из аргументов которого входит переменная, является подформулой, а в элементарной формуле вхождение любой переменной свободное. Еще один пример. В формуле

$$(\forall \boldsymbol{x} p(\boldsymbol{x}, y) \wedge q(x)) \to (\exists \boldsymbol{x} r(\boldsymbol{x}, \boldsymbol{x}))$$

полужирным шрифтом помечены связанные вхождения переменных, а светлым — свободные.

Свободные переменные, входящие в терм или формулу, называются ***параметрами*** этого терма или формулы. Если формула не имеет свободных переменных, то она называется ***замкнутой*** или ***предложением***.

При записи формул будем использовать сокращения скобок на основе приоритета операций. Наивысшим приоритетом обладают кванторные приставки. Затем идет отрицание, далее дизъюнкция и конъюнкция. Наинизшим приоритетом обладает импликация. Мы будем также использовать сокращение $(X \sim Y)$ для формулы $((X \to Y) \wedge (Y \to X))$, причем символу $\sim$ (эквиваленция) припишем самый низкий приоритет (ниже, чем у импликации). Кроме того, некоторые из функциональных символов арности 2, будем употреблять в инфиксной форме (например, символы алгебраических операций). Все эти соглашения используются для более удобной записи и с теоретическими построениями не связаны. От подобных соглашений всегда можно избавиться, модфицировав соответствующим образом терм или формулу. Разумеется, эти элементы можно было бы включить в формальный язык, но это в теоретическом плане ничего существенного не добавит, хотя заметно усложнит изложение.

4.3. Переименования и подстановки

Роль свободных и связанных переменных. Переименование. Коллизия переменных при переименовании. Отношение конгруэнтности ($\approx$). Подстановка, свободная для данного выражения. Сохранение конгруэнтности при подстановке. Свойство чистоты переменных. Лемма о чистоте переменных.

С понятием связанной переменной мы встречаемся уже в курсе математического анализа Обозначения предела, определенного интеграла дают примеры связывания переменных. Из того же курса известно, что связанную переменную можно изменять, не меняя смысла выражения. Однако это в общем верное правило может приводить к неверным результатам. Например, если в выражении $\lim\limits_{x\to 0} \cos(x^2+y^2)$ заменить связанную переменную x переменной y, получим формулу $\lim\limits_{y\to 0} \cos(y^2+y^2)$, имеющую совсем другой смысл. В этом случае говорят, что при подстановке произошла коллизия переменных.

Одновременная замена всех вхождений переменной x в кванторной приставке[*] ∇x и в области действия этой приставки на переменную y того же сорта называется ***переименованием***. ***Коллизия переменных при переименовании*** в формуле $\nabla x\, X$ — ситуация, когда при переименовании переменной x в переменную y в X или в одной из ее подформул есть переменная, которая из свободной превращается в связанную. Например, в формуле $\forall x\, p(x,y,z)$ переменные y и z являются свободными. Но при переименовании x в y получим $\forall y\, p(y,y,z)$, где только одна переменная z осталась свободной. Может быть и другая ситуация. В формуле $\forall x\, p_1(x,(\forall y\, p_2(x,y)),z)$ оба вхождения переменной y связанные. Переменная x в формулу входит связанно (раз есть квантор), но в подформулу $\forall y\, p_2(x,y)$ эта переменная уже входит свободно. Но переименование x в y переводит y в разряд связанной переменной в этой подформуле: $\forall y\, p_1(y,(\forall y\, p_2(y,y)),z)$. В этом случае также произошла коллизия переменных.

Можно сказать по-другому. Назовем переменную x в формуле Z ***свободной переменной*** для переменной y, если никакие вхождения y в Z, являющиеся свободными для Z или одной из ее подформул, не попадают в область действия кванторных приставок с переменной x. Коллизия переменных при переименовании — попытка заменить кванторную переменную x на y в то время как x не является свободной для y.

Мы будем говорить, что две формулы X и X' ***конгруэнтны*** (являются вариантами одна другой, обозначение $X \approx X'$), если одна получена из другой с помощью правильного, т.е. без коллизии переменных, переименования. Точное определение этого понятия можно дать индукцией по построению формулы:

1) элементарная формула конгруэнтна только себе самой;

[*]Напомним, что символ ∇ обозначает любую из двух кванторных приставок.

2) если $Y \approx Y'$, $Z \approx Z'$, то $\neg Y \approx \neg Y'$ и $Y \nabla Z \approx Y' \nabla Z'$, где ∇ — одна из трех двуместных логических связок;

3) если $X = \nabla x\, Y$, $X' = \nabla y\, Z$, то $X \approx X'$, если x и y одного сорта и для любой переменной z того же сорта, вообще не входящей ни в Y, ни Z, имеем $Y_z^x \approx Z_z^y$, где W_u^v есть результат замены в W всех свободных вхождений переменной v на переменную u.

Отметим, что конгруэнтность — это отношение эквивалентности (симметричное, рефлексивное и транзитивное). Таким образом, множество всех формул разделяется на классы попарно конгруэнтных формул. Хотя формально конгруэнтные формулы различаются, нет оснований такое различие учитывать. С математической точки зрения здесь осуществляется переход от понятия „формула“ к более общему понятию „класс конгруэнтных формул“.

Под подстановкой θ мы будем понимать символическую запись вида

$$\theta = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ t_1 & t_2 & \dots & t_n \end{pmatrix},$$

в которой $x_1, x_2, \dots, x_n$ — конечный набор (предметных) переменных, а $t_1, t_2, \dots, t_n$ — набор термов, причем сорт терма t_i совпадает с сортом переменной x_i. Такую запись мы всегда можем трактовать как запись отображения, определенного на конечном подмножестве множества всех предметных переменных языка, в множество Tm всех термов языка, которое сохраняет сортность термов. Допускается, что приведенная таблица может быть пустой.

Далее через $T\theta$ будем обозначать результат подстановки θ в выражение (т.е. терм или формулу) T, т.е. результат замены в T всех свободных вхождений каждой переменной x_i из подстановки θ соответствующим термом t_i. Отметим, что при этом некоторые переменные x_i могут не входить свободно в T. В этом случае в отношении x_i не происходит никаких замен, соответствующий столбец в подстановке θ можно опустить.

В алгебре высказываний подстановка — один из способов преобразования формулы в эквивалентную. При этом на подстановку не накладывалось никаких ограничений. Ситуация меняется с появлением кванторов. При подстановке в формуле с кванторами смысл формулы может заметно измениться. Поэтому не все подстановки пригодны с точки зрения логики. Например, формула[**] $x + y = z$ с натуральными переменными x, y, z есть предикат от трех переменных. Формула $\exists y\,(x + y = z)$ есть предикат уже от двух переменных x и z, логическим эквивалентом которого является формула $x < z$. Заменим в этой формуле свободную переменную x термом $x \cdot y$, надеясь на то, что получим предикат $x \cdot y < z$. Однако формула $\exists y\,(x \cdot y + y = z)$ имеет совсем другой смысл. Здесь возникла та же проблема, что и при переименовании — коллизия переменных: при замене переменной x термом $x \cdot y$ переменная y, входящая в терм свободно, оказалась в области действия кванторной приставки и стала связанной. Чтобы избежать искажения, достаточно было предварительно „убрать“ из формулы связанную переменную y с помощью переименован ия а уж затем выполнить подстановку: $\exists u\,(x \cdot y + u = z)$. При таком порядке действий действительно получим предикат $x \cdot y < z$.

Подстановку θ назовем ***свободной для выражения*** T, если для любой переменной x из области определения $\mathrm{dom}\,\theta$ подстановки θ (т.е. встречающейся в первой строке матрицы), никакое свободное вхождение x в T не попадает в область действия никаких кванторных приставок с переменными, свободно входящими в $\theta(x)$.

В частности, если ни один из параметров формул $\theta(x)$ не является кванторной переменной в T, то θ свободна для T. Это будет так в случае, когда каждая из формул $\vartheta(x)$ вовсе не имеет параметров, т.е. является замкнутой. Такая подстановка называется ***константной***.

Если подстановка θ не является свободной для T, то, как показывает рассмотренный выше пример, можно, не меняя логической сути формулы, выполнить переименование переменных в T, переходя к конгруэнтной формуле T', для которой θ уже будет свободной. Этот подход

[**]Отмечу, что любое алгебраическое уравнение является не термом, а формулой: его можно рассматривать как утверждение, касающееся указанных переменных, при этом решение уравнения представляет собой выявление области истинности этой формулы.

позволяет обеспечить универсальность подстановки, но, правда, пожертвовав однозначностью: результат будет лишь с точностью до когруэнтности.

Теорема 4.1. Если $X' \approx X''$, а θ свободна для X' и X'', то $X'\theta \approx X''\theta$.

◀ Доказательство базируется на индукции по построению формулы и на индуктивном определении конгруэнтности. Если $X' \approx X''$, то формулы X либо обе элементарные, либо обе получены с помощью логической связки, либо обе построены присоединением кванторной приставки. В первом случае X' и X'' совпадают, а значит, формулы $X'\theta$ и $X''\theta$ также элементарны и совпадают, т.е. $X' \approx X''$.

Пусть, например, $X' = U' \vee V'$ и $X'' = U'' \vee V''$, причем $U' \approx U''$, $V' \approx V''$. По индуктивному предположению $U'\theta \approx U''\theta$ и $V'\theta \approx V''\theta$. Следовательно, $U'\theta \vee V'\theta \approx U''\theta \vee V''\theta$ и

$$X'\theta = (U' \vee V')\theta = U'\theta \vee V'\theta \approx U''\theta \vee V''\theta = (U'' \vee V'')\theta = X''\theta.$$

Аналогичны рассуждения для других логических связок.

Пусть $X' = \nabla x U'$, $X'' = \nabla y U''$. Выберем переменную z, не входящую ни в формулы U', U'', ни в формулы подстановки θ. Тогда, согласно условию $X' \approx X''$, имеем $(U')^x_z \approx (U'')^y_z$. Согласно индуктивному предположению, $(U')^x_z\theta \approx (U'')^y_z\theta$. Удалим из θ столбцы с переменными x и y, если они есть. Получим подстановку $\overline{\theta}$. В области значений этой подстановки переменные x и y не встречаются, поскольку $\overline{\theta}$, как и θ, свободна для X' и X''. Из равенства $(U')^x_z\theta \approx (U'')^y_z\theta$ вытекает $(U')^x_z\overline{\theta} \approx (U'')^y_z\overline{\theta}$, а так как $(U')^x_z\overline{\theta} = (U'\overline{\theta})^x_z$ и $(U'')^y_z\overline{\theta} = (U''\overline{\theta})^y_z$, то $(U'\overline{\theta})^x_z \approx (U''\overline{\theta})^y_z$. Следовательно, $\nabla x(U'\overline{\theta}) = \nabla y(U''\overline{\theta})$ и $\nabla x(U'\theta) = \nabla y(U''\theta)$, потому что под квантором кванторная переменная все равно не заменяется. ▶

Будем говорить, что формула X обладает ***свойством чистоты переменных***, если, во-первых, никакая переменная кванторной приставки не входит в X свободно, и во-вторых, разные кванторные приставки содержат разные переменные.

Следующая лемма позволяет утверждать, что в любой формуле T можно устроить переименование, в результате которого данная подстановка θ станет свободной для переименованной формулы.

Лемма 4.1 (о чистоте переменных). Для любой формулы X и любого множества S переменных существует формула Y, когруэнтная X, обладающая свойством чистоты переменных, в которой нет связанных переменных, принадлежащих S.

◀ Доказательство проводится индукцией по построению формулы. Для атомарных формул утверждение тривиально, поскольку такие формулы вообще не имеют связанных переменных. Пусть задано множество S и формула $X = Y\nabla Z$, где ∇ — одна из двуместных логических связок. Предположим, что относительно формул Y и Z уже доказано утверждение леммы с произвольным множеством S. Пусть S_X — множество всех переменных (и свободных, и связанных), входящих в X. Существует формула Y', конгруэнтная Y, обладающая свойством чистоты переменных и не имеющая связанных переменных из множества $S \cup S_X$. Обозначим через S_Y множество всех переменных, входящих в Y'. Существует формула X', конгруэнтная X, обладающая свойством чистоты переменных и не имеющая связанных переменных из множества $S \cup S_Y$. Формула $X'\nabla Y'$ обладает свойством чистоты переменных (любая кванторная переменная, например, в X' не совпадает со „своими“ переменными, т.е. другими кванторными переменными формулы X', а по построению и с „чужими“ переменными, составляющими множество S_Y). Кроме того, в построенной формуле нет связанных переменных, попадающих в S. Для одноместной связки $\neg$ утверждение теоремы очевидно.

Пусть $X = \nabla x\, Y$, где ∇ — один из кванторов. Сперва надо „вывести“ новую кванторную переменную за пределы S. Для этого выбираем новую переменную u, не принадлежащую S и не входящую никаким образом в Y. Подстановка $\binom{x}{u}$ свободна и мы получаем формулу $\nabla u\, Y^x_u$, конгруэнтную X. Если мы теперь заменим Y^x_u конгруэнтной формулой Y' со свойством чистоты

переменных и без переменных из S, то получим конгруэнтную Y формулу $\nabla u\, Y'$ со свойством чистоты переменных и без переменных из S. ►

В соответствии с доказанной леммой, чтобы выполнить в X правильную подстановку θ, необходимо переименовать X в X' так, что среди переменных X' не будут встречаться переменные термов $\theta(x)$. Тогда подстановка θ будет свободной для X', и мы можем реализовать эту подстановку.

4.4. Семантика логико-математического языка

Интерпретация логико-математического языка. Предметная область. Интерпретация предметных констант. Интерпретация функциональных и предикатных символов. Интерпретация логических связок и кванторов. Расширение языка введением символов реальных объектов. Оценка и оцененные выражения.

Сами по себе термы и формулы логико-математического языка ничего не значат. Это лишь средство для обозначения или записи реальных объектов. Чтобы термы и формулы приобрели какой-то содержательный смысл, необходимо связать их с реальными объектами из какой-либо области математики, т.е. определить ***интерпретацию логико-математического языка***, или построить ***модель языка***. Для этого мы должны предусмотреть несколько вещей.

Во-первых, необходимо придать смысл предметным переменным, определив их возможные значения. Мы назовем ***носителем*** (или ***предметной областью***) языка Ω отображение, которое каждому сорту $\pi \in S$ ставит в соответствие некоторое непустое множество D_π. Множество D_π представляет собой носитель сорта π или, другими словами, область изменения переменных сорта π.

Во-вторых, следует придать смысл предметным константам. Естественно каждую константу $c \in C$ сорта π связать с некоторым элементом $\widetilde{c} \in D_\pi$ из носителя сорта π, т.е. задать отображение $\widetilde{C}\colon C \to \bigcup\limits_{\pi\in S} D_\pi$, сохраняющее сортность.

В-третьих, нужно каждому функциональному символу f типа $(\pi_0,\ \ldots,\ \pi_n)$ поставить в соответствие конкретное отображение $\widetilde{f}\colon D_{\pi_1} \times \ldots \times D_{\pi_n} \to D_{\pi_0}$, т.е. задать некоторое отображение $\widetilde{F}\colon f \mapsto \widetilde{f}$.

В-четвертых, следует каждому предикатному символу p типа $(\pi_1,\ \ldots,\ \pi_n)$ поставить в соответствие n-местный предикат того же вида, т.е. отображение, которое каждому элементу множества $D_{\pi_1} \times \ldots \times D_{\pi_n}$ ставит в соответствие высказывание. Впрочем, поскольку нас интересует лишь истинность формул, можно ограничиться булевым вариантом и символу p поставить в соответствие отображение $\widetilde{p}\colon D_{\pi_1} \times \ldots \times D_{\pi_n} \to \mathbb{B}$. Отдельно упомянем предикатные символы арности 0. Это пропозициональные переменные, которые в случае булевой интерпретации можно заменить двумя булевыми константами.

В-пятых, следует придать смысл всем логическим связкам и кванторам. Мы будем считать, что они имеют общепринятый смысл и что их интерпретация фиксирована.

Все сказанное означает, что интерпретация — предписание, которое символам языка ставит в соответствие настоящие объекты. Говорят, что модель языка определяет его ***семантику***.

Если задана интерпретация M языка Ω, то каждый терм сорта π определяет отображение, которое каждому набору значений переменных ставит в соответствие объект сорта π, т.е. элемент множества D_π. Другими словами, терм получает значение, если известны значения всех его параметров. Значение терма легко определить индукцией по построению.

Обычно придание переменным терма некоторых значений рассматривается как некоторый формальный акт подстановки в терм вместо свободных переменных других объектов. Вообще говоря, использовать в выражении конкретные объекты не очень удобно. Во-первых, это выводит за пределы рассматриваемого языка. Во-вторых, подстановка в выражение „неизвестно

чего“ может привести к неоднозначностям или другим казусам, поскольку так называемые „реальные“ объекты вполне могут оказаться словами другого языка.

Указанный формальный подход можно реализовать следующим образом. Введем в обиход дополнительный набор констант, каждая из которых соответствует реальному объекту. Более точно, выберем множество $\underline{C}$ символов, равномощное $D = \bigcup\limits_{\pi\in S} D_\pi$ и биективное отображение $\widehat{C}\colon D \to \underline{C}$, устанавливающее взаимно однозначное соответствие между объектами $a \in D_\pi$ и символами $\underline{a}$ того же сорта. В формулах будем использовать не сами объекты, о сопоставленные им символы. В результате получим некоторое расширение $\Omega(D)$ исходного языка Ω.

Теперь мы можем в рамках расширенного языка выполнить формальную подстановку вида

$$\theta = \begin{pmatrix} x_1 & x_2 & \dots x_n \\ \underline{a_1} & \underline{a_2} & \dots \underline{a_n} \end{pmatrix}.$$

Это константная подстановка, называемая ***оценкой языка*** Ω. Если θ содержит все переменные терма T, то терм $T\theta$ оказывается замкнутым выражением (без параметров) и будет иметь конкретное значение. В этом случае θ называется ***оценкой для выражения*** T, а само выражение $T\theta$ называется ***оцененным***. В общем оцененное выражение — это любое замкнутое выражение языка $\Omega(D)$.

Все сказанное переносится и на формулы языка Ω. Значением оцененной формулы является ложь или истина. При этом говорят, что оцененная формула истинна (ложна) в модели M. Подчеркнем, что сами по себе формулы не имеют какого-либо значения. Истинность возникает, когда конкретную интерпретацию получают все функциональные и предикатные символы, а формула не содержит параметров (или все они заменены константами в результате оценки).

Пример 4.1. В качестве примера логико-математического языка рассмотрим язык элементарной арифметики. Этот язык имеет лишь один сорт переменных, которые мы будем обозначать x, y, z, … Единственную константу обозначим 0. Три функциональных символа f, g, h, из которых первый одноместный, два остальных двуместные. Единственный двуместный предикатный символ обозначим p.

Для функциональных и предикатного символов введем альтернативные бинарные нотации:
$f(x) \equiv x^+$ (переход к следующему натуральному числу);
$g(x,y) \equiv (x+y)$ (операция сложения);
$h(x,y) \equiv (x\cdot y)$ (операция умножения);
$p(x,y) \equiv (x=y)$ (формальное равенство).

Для введенного языка рассмотрим две модели. Первую обозначим ω. Носителем этой модели является множество $\mathbb{N}_0$ целых неотрицательных чисел. Символу 0 поставим в соответствие число нуль. Функциональным символам припишем соответствующие операции над целыми числами. Наконец, равенство $(x=y)$ будет истинным тогда и только тогда, когда два числа x и y совпадают.

Рассмотрим терм $(x+y)\cdot z + x^+$. Этому терму мы можем давать различные оценки. Например, при оценке $\begin{pmatrix} x & y & z \\ \underline{1} & \underline{2} & \underline{3} \end{pmatrix}$ получим терм $(\underline{1}+\underline{5})\cdot\underline{3}+\underline{1}^+$, который в ω имеет значение 20. Формула $\exists y\,(x+y=z)$ при оценке $\begin{pmatrix} x & z \\ \underline{3} & \underline{5} \end{pmatrix}$ дает истинную оцененную формулу $\exists y\,(\underline{3}+y=\underline{5})$. В то же время при оценке $\begin{pmatrix} x & z \\ \underline{5} & \underline{3} \end{pmatrix}$ получим ложную формулу $\exists y\,(\underline{5}+y=\underline{3})$.

Носителем второй модели, которую обозначим R, является множество $\mathbb{R}$ действительных чисел. Функциональным символам опять сопоставим сложение, умножение и добавление единицы. Формула $x=y$ выражает совпадение действительных чисел.

Вообще говоря, в качестве интерпретации рассматриваемого языка можно выбрать любое множество D, два любых отображения типа $f, g\colon D^2 \to D$, отображение типа $h\colon D \to D$ и отображение $p\colon D \to \mathbb{B}$. Все возможно. Но при этом меняется смысл термов и формул. Например, в модели ω истинна формула $x+y=y+x$, отражающая арифметический закон. Она сохраняется в модели R, но теряет силу в других моделях.

4.5. Логические законы

Формулы общезначимые, выполнимые, опровержимые, тождественно ложные. Пропозициональные тавтологии. Тавтологии с кванторами. Пример: $\neg\exists x\neg X \to \forall x\, X$ — тавтология; $\forall x\,\exists y\, p(x,y) \to \exists y\,\forall x\, p(x,y)$ — нет. Основные логические законы: а) де Моргана (пронесение через отрицание); б) пронесение кванторов через бинарные связки (8 законов); в) обобщенное пронесение кванторов через дизъюнкцию и конъюнкцию (4 закона); г) добавление кванторов или изменение порядка (5 законов); д) законы конгруэнтности.

Среди формул языка Ω есть такие, которые остаются истинными при любой интерпретации и при любой оценке. Такие формулы называют ***тавтологиями*** или ***общезначимыми формулами***. То, что X — тавтология, будем записывать так: $\models X$. Например, формула $p \to (q \to p)$ истинна в любой модели, поскольку вообще не имеет параметров и в любой модели автоматически становится оцененной, тождественно истинной формулой. В то же время есть формулы, которые являются тождественно истинными для данной модели, но не являются тождественно истинными в другой модели. Например, формула $x \cdot y = y \cdot x$ в модели ω тождественно истинна. Однако, если в качестве модели взять множество квадратных матриц (операции x^+ можно поставить в соответствие добавление единичной матрицы) с естественной интерпретацией двуместных функциональных символов, формула $x \cdot y = y \cdot x$ уже не будет тождественно истинной. Если формула X является истинной в данной модели M при любой оценке (т.е. тождественно истинна в M), но не является истинной в некоторой другой модели при некоторой оценке, то эта формула выражает ***закон в модели*** M, истинность X в модели M будем обозначать $M \models X$.

Поскольку нульарные предикатные символы не имеют параметров, их значение не меняется в рамках заданной модели. Поэтому множество всех нульарных предикатных символов в сочетании с логическими связками образует подмножество рассматриваемого языка, идентичное языку алгебры высказываний. Следовательно, любой тавтологии из алгебры высказываний соответствует тавтология языка Ω. Например, $\models \neg A \vee B \to \neg(A \wedge \neg B)$, так как это калька с тавтологии из алгебры высказываний. Такие тавтологии будем называть ***пропозициональными***. В них нет ни предметных переменных, ни кванторов.

Кроме пропозициональных тавтологий в языке Ω есть и другие тавтологии, порожденные кванторами. Например $\neg\exists x\neg X \to \forall x\, X$ — тавтология. Чтобы это показать, выберем произвольную модель M и в ней произвольную оценку θ. Поскольку переменная x не входит свободно в X, можно считать, что x не входит и в θ. Проделаем подстановку по всем правилам в формуле. В результате придем к $M \models \neg\exists x\neg(X\theta) \to \forall x\,(X\theta)$. Импликация связывает две оцененные формулы. Для доказательства ее истинности можно предположить, что левая часть импликации истинна, т.е. $M \models \neg\exists x\neg(X\theta)$. Надо доказать, что $M \models \forall x\,(X\theta)$, т.е. для всякого объекта $a \in D_\pi$ верно $M \models (X\theta)^x_{\underline{a}}$. Предполагая противное, заключаем, что существует такой объект $a \in D_\pi$, что формула $(X\theta)^x_{\underline{a}}$ в M ложна. Но тогда $M \models \neg(X\theta)^x_{\underline{a}}$. Следовательно $M \models \exists x\,\neg(X\theta)$. Но это противоречит предположению $M \models \neg\exists x\neg(X\theta)$.

Пример 4.2. Покажем, что формула $\forall x\,\exists y\, p(x,y) \to \exists y\,\forall x\, p(x,y)$ не является логическим законом. Для этого необходимо придумать такую модель M, что $M \models \forall x\,\exists y\, p(x,y)$, но формула $\exists y\,\forall x\, p(x,y)$ в модели M не является истинной (отметим, что рассматриваемые формулы замкнуты, а потому в любой модели автоматически являются оцененными).

В качестве модели выберем ω — модель элементарной арифметики. Предикатному символу $p(x,y)$ поставим в соответствие отношение $x < y$ на множестве натуральных чисел. Тогда формула $\forall x\,\exists y\, p(x,y)$ будет означать, что в множестве натуральных чисел с отношением порядка $x < y$ нет максимального элемента, а формула $\exists y\,\forall x\, p(x,y)$ будет иметь противоположный смысл, что в множестве натуральных чисел есть наибольший элемент. Ясно, что первая формула истинна, а вторая нет.

Две формулы X и Y называются ***логически эквивалентными***, $X \equiv Y$, если формула $X \sim Y$ есть логический закон (напомним, что запись $X \sim Y$ — это сокращение формулы

$(X \to Y) \wedge (Y \to X)$). Можно сказать иначе: две формулы логически эквивалентны, если в любой модели и при любой оценке они обе или истинны, или ложны.

Теорема 4.2. Если $X \equiv Y$, то $\forall x X \equiv \forall x Y$ и $\exists x X \equiv \exists x Y$.

◄ Пусть x, x_1, x_2, …, x_n — полный набор всех параметров формул X и Y, в котором в качестве первой выбрана переменная x. При произвольно выбранной интепретации языка и при указанном порядке переменных формулы X и Y порождают булевы функции $f_X(x_1, x_2, \ldots, x_n)$ и $f_Y(x_1, x_2, \ldots, x_n)$. Эквивалентность формул означает, что эти функции совпадают. Но тогда

$$\min_x f_X(x, x_1, \ldots, x_n) = \min_x f_Y(x, x_1, \ldots, x_n) \quad \text{и} \quad \max_x f_X(x, x_1, \ldots, x_n) = \max_x f_Y(x, x_1, \ldots, x_n).$$

Вычисление минимума или максимума функций f_X, f_Y равносильно навешиванию квантора $\forall$ или $\exists$ с переменной x на формулы X, Y. Это указывает на то, что пары формул $\forall x X$ и $\forall x Y$, $\exists x X$ и $\exists x Y$ эквивалентны в выбранной модели. Так как модель выбиралась произвольно, заключаем, что эти пары формул логически эквивалентны. ►

Приведем некоторые логические законы. Остановимся лишь на тех, которые выходят за рамки пропозициональных.

Законы де Моргана:

1. $\neg(\forall x\, X) \sim \exists x\, (\neg X)$.
2. $\neg(\exists x\, X) \sim \forall x\, (\neg X)$.

Доказательства общезначимости этих формул аналогичны доказательству общезначимости формулы $\neg\exists x \neg X \to \forall x\, X$, рассмотренной выше.

Одностороннее пронесение кванторов (здесь X не содержит свободно переменной x):

3. $X \wedge \forall x\, Y(x) \sim \forall x\, (X \wedge Y(x))$.
4. $X \vee \forall x\, Y(x) \sim \forall x\, (X \vee Y(x))$.
5. $X \wedge \exists x\, Y(x) \sim \exists x\, (X \wedge Y(x))$.
6. $X \vee \exists x\, Y(x) \sim \exists x\, (X \vee Y(x))$.
7. $X \to \forall x\, Y(x) \sim \forall x\, (X \to Y(x))$.
8. $X \to \exists x\, Y(x) \sim \exists x\, (X \to Y(x))$.
9. $\forall x\, Y(x) \to X \sim \exists x\, (Y(x) \to X)$.
10. $\exists x\, Y(x) \to X \sim \forall x\, (Y(x) \to X)$.

Рассмотрим, например, формулу 3. Выберем произвольную модель и в ней произвольную оценку θ. Поскольку x в формулу не входит свободно, считаем, что и в θ переменная x не выходит. Получим оцененную формулу $X\theta \wedge \forall x\, Y\theta \sim \forall x\, (X\theta \wedge Y\theta)$, которая является истинной в том и только в том случае, когда обе части эквиваленции имеют одно и то же истинностное значение. Если левая часть импликации истинна, то формулы $X\theta$ и $\forall x\, Y\theta$ обе истинны. Значит, при любом $\underline{a}$ формула $(Y\theta)^x_{\underline{a}}$ является истинной. Следовательно, истинна формула $X\theta \wedge (Y\theta)^x_{\underline{a}}$. Отсюда вытекает, что формула $\forall x(X\theta \wedge (Y\theta))$ в правой части эквиваленции истинна. Нетрудно увидеть, что приведенные рассуждения можно провести в обратном порядке и тем самым показать, что из истинности правой части эвиваленции вытекает истинность левой части.

Отметим, что доказанная формула 3 имеет естественную интрпретацию при выборе конкретной модели:

$$\min\{f_X,\ \min_x f_Y\} = \min_x \min\{f_X,\ f_Y\}.$$

Это верно при условии, что функция f_X не зависит от переменной x.

Дополнительные законы пронесения кванторов:

11. $\forall x\, X(x) \wedge \forall x\, Y(x) \sim \forall x\, (X(x) \wedge Y(x))$.
12. $\exists x\, X(x) \vee \exists x\, Y(x) \sim \exists x\, (X(x) \vee Y(x))$.
13. $\exists x\, (X(x) \wedge Y(x)) \to \exists x\, X(x) \wedge \exists x\, Y(x)$.
14. $\forall x\, X(x) \vee \forall x\, Y(x) \to \forall x\, (X(x) \vee Y(x))$.

Рассмотрим, например, формулу 11. Выбрав произвольную модель и в ней произвольную оценку θ (считаем, что она не содержит x), получим оцененную формулу $\forall x\, X\theta \wedge \forall x\, Y\theta \sim \sim \forall x\,(X\theta \wedge Y\theta)$. Истинность левой части означает, что истинны формулы $\forall x\, X\theta$ и $\forall x\, Y\theta$. Следовательно, для любой константы $\underline{a}$ истинны $(X\theta)^x_{\underline{a}}$ и $(Y\theta)^x_{\underline{a}}$, откуда заключаем, что истинна формула $(X\theta)^x_{\underline{a}} \wedge (Y\theta)^x_{\underline{a}}$. Поэтому истинна $\forall x\,(X\theta \wedge Y\theta)$. Наоборот, если формула $\forall x\,(X\theta \wedge Y\theta)$ истинна, то для любой константы $\underline{a}$ истинна формула $(X\theta)^x_{\underline{a}} \wedge (Y\theta)^x_{\underline{a}}$. Делаем вывод, что для любой $\underline{a}$ истинна формула $(X\theta)^x_{\underline{a}}$ и для любой $\underline{a}$ истинна формула $(Y\theta)^x_{\underline{a}}$. Поэтому истинны формулы $\forall x\, X\theta$ и $\forall x\, Y\theta$, а значит, и формула $\forall x\, X\theta \wedge \forall x\, Y\theta$.

Логические законы, связанные добавлением кванторов или изменением их порядка:

15. $\forall x\, X \to X$.
16. $X \to \exists x\, X$.
17. $\exists x \exists y X \sim \exists y \exists x X$.
18. $\forall x \forall y X \sim \forall y \forall x X$.
19. $\exists x \forall y X \to \forall y \exists x X$.

Формулы 15 и 16 можно интерпретировать как утверждение, что для функции $u(x)$ имеет место неравенство

$$\min_x u(x) \leqslant u(x) \leqslant \max_x u(x). \tag{4.1}$$

Формулы 17 и 18 ассоциируются с правилом перестановки максимумов и минимумов по разным переменным, например:

$$\max_x \max_y u(x,y) = \max_y \max_x u(x,y),$$

а формула 19 — отражение неравенства

$$\max_x \min_y u(x,y) \leqslant \min_y \max_x u(x,y).$$

И в самом деле, $u(x,y) \leqslant \max_x u(x,y)$. Следовательно, $\min_y u(x,y) \leqslant \min_y \max_x u(x,y)$. В результате справа — константа, а слева — функция переменного x, т.е. $\min_y \max_x u(x,y)$ есть верхняя грань функции $\min_y u(x,y)$. Значит, имеет место неравенство (4.1).

Логические законы, связанные с конгруэнтностью (формула X не имеет свободных вхождений переменной y):

20. $\forall x\, X \sim \forall y\, X^x_y$.
21. $\exists x\, X \sim \exists y\, X^x_y$.

Эти логические законы указывают на то, что конгруэнтные формулы логически эквивалентны. Впрочем, эквивалентность конгруэнтных формул лучше доказывать непосредственно, опираясь на индуктивное определение конгруэнтности. Иначе придется доказывать, что конгруэнтная формула получается последовательным переименованием, что интуитивно ясно, но требует формальных рассуждений.

Что касается сформулированных логических законов, то можно рассуждать так (на пример квантора всеобщности). Выбираем произвольную модель и в ней произвольную оценку θ. При этом, поскольку ни x, ни y в формулу $\forall x X$ не входят свободно, можно считать, что и в оценку θ эти переменные не входят. Если переменная x не входит в X свободно, то наличие квантора не играет роли: формула $X\theta$ оказывается оцененной. Если же x входит в X свободно, то формула $X\theta$ будет содержать единственный параметр x. Истинность $\forall x(X\theta)$ означает, что для любого объекта $\underline{a}$ формула $(X\theta)^x_{\underline{a}}$ оказывается истинной. Но если мы предварительно выполним подстановку $(X\theta)^x_y$), а затем подстановку $\binom{y}{\underline{a}}$, то получим ту же формулу $(X\theta)^x_{\underline{a}})$. Она окажется истинной. Аналогичны рассуждения при ложности формулы $\forall x(X\theta)$. Тем самым доказано, что в выбранной модели формулы $\forall x X$ и $\forall y X^x_y$ при любой оценке одновременно истинны или ложны. Значит, они эквивалентны, а формула $\forall x X \sim \forall y X^x_y$ есть логический закон.

4.6. Замены

Понятие формального предиката. Замена элементарной формулы формальным предикатом (индуктивное определение). Теорема о замене элементарных подформул. Правило замены эквивалентным. Пример: эквивалентность $r(x)\vee\forall z\,(\neg\forall x\,q(x,z))$ и $r(x)\vee\forall z(\exists x\,\neg q(x,z))$. Принцип двойственности. Нильпотентность. Эквивалентность двойственных формул.

Речь идет о получении аналога теоремы 2.3. У нас нет самостоятельных пропозициональных переменных. Их роль играют предикатные символы арности нуль. Разумеется, теорема 2.3 остается верной при замене предикатного символа арности нуль какой-либо формулой. Но как быть, если нужно заменить предикатный символ, имеющий параметры? Таким образом, речь идет об обобщении теоремы 2.3 на случай замены в формуле произвольной элементарной подформулы. Механизм замены должен сохранять эквивалентность формул.

При замене какого-либо вхождения элементарной подформулы $p(t_1,\ldots,t_n)$ (аргументы в этой подформуле — какие-либо термы) некоторой формулой X следует установить соответствие между переменными, входящими в элементарную подформулу, и переменными, входящими в X. Такое соответствие можно организовать, вводя понятие ***формального предиката***. Под формальным предикатом мы будем понимать слово вида $x_1x_2\ldots x_nX$. Такое слово означает лишь, что задана какая-то формула X и упорядоченный набор переменных x_1, x_2, …, x_n — больше ничего. Эти переменные и все их вхождения в X в рамках формального предиката будем считать связанными. Кортеж сортов $(\pi_1,\ \pi_2,\ \ldots,\ \pi_n)$ „объявленных“ переменных $x_1, x_2, \ldots, x_n$ будем называть типом формального предиката.

Чтобы в формуле Y заменить элементарную подформулу $p(t_1,\ldots,t_k)$, необходимо построить формальный предикат $U = x_1x_2\ldots x_nX$ того же типа, что и предикатный символ p, затем в формуле X выполнить подстановку $X^{x_1,\ldots,x_n}_{t_1,\ldots,t_n}$ и результат подстановки подставить в формулу Y взамен подформулы $p(t_1,\ldots,p_n)$.

Подобный подход будет более понятным, если учесть, что в любой модели формула задает предикат, роль аргументов которого играют переменные формулы. Задавая формальный предикат, мы выбираем некоторую совокупность переменных (записываем их перед формулой) и упорядочиваем. Эти отобранные переменные фиксируются, остальные остаются свободными. При подстановке вместо элементарной формулы мы должны на место „объявленных аргументов“ предиката подставить аргументы элементарной формулы и в таком виде подставить формулу вместо элементарной. Например, пусть элементарная формула имеет вид $p(t_1,t_2,\ldots,t_n)$. Формальный предикат $x_1x_2\ldots x_nX$ следует в любой модели интерпретировать как предикат $\widetilde{p}(x_1,\ldots,x_n,z_1,\ldots,z_m)$, в котором первые n аргументов по типам соответствуют аргументам предиката p. В результате мы меняем в исходной формуле элементарную подформулу $p(t_1,t_2,\ldots,t_n)$ на подформулу $\widetilde{p}(t_1,\ldots,t_n,z_1,\ldots,z_m)$. Отобранные элементы формального предиката играют роль подстановочных мест и заменяются аргументами элементарной подформулы. Оставшиеся переменные формулы X без изменений переходят в изменяемую формулу.

Указанный алгоритм подстановки можно описать с помощью индукции по построению формулы. Пусть задан формальный предикат $U = x_1x_2\ldots x_nX$ и Y обозначает исходную формулу. Результат $Y(p\,||\,U)$ подстановки в Y формального предиката U типа $(\pi_1,\ \pi_2,\ \ldots,\ \pi_n)$ вместо элементарной подформулы $p(t_1,\ldots,t_n)$ с предикатным символом p того же типа определим следующим образом.

1. Если Y — атомарная формула $q(r_1,\ldots,r_m)$, то при q, отличном от p полагаем $Y(p\,||\,U) = Y$, а если $Y = p(t_1,\ldots,t_n)$, то полагаем $Y(p\,||\,U) = X\theta$, где $\theta = \begin{pmatrix} x_1 & x_2 & \ldots & x_n \\ t_1 & t_2 & \ldots & t_n \end{pmatrix}$.

2. Если $Y = V\nabla W$, где ∇ — одна из трех двуместных логических связок, то $Y(p\,||\,U) = V(p\,||\,U)\nabla W(p\,||\,U)$. Если $Y = \neg V$, то $Y(p\,||\,U) = \neg V(p\,||\,U)$.

3. Если $Y = \nabla z\,V$, где ∇ — один из кванторов, то возможны два случая. В первом случае, когда U не содержит свободных вхождений z или V не содержит вхождений предикатного символа p, полагаем $Y(p||U) = \nabla z\,V(p||U)$. Во втором случае, когда z имеет свободные вхождения

в U, а V имеет вхождения p, выбираем новую переменную u (т.е. не встречающуюся ни в Y, ни в U) и полагаем $Y(p \,||\, U) = \nabla u\,(V_u^z)(p \,||\, U)$.

Обратим внимание на то, что в соответствии с описанным индуктивным правилом при замене меняются все вхождения данного предикатного символа. Последний пункт регулирует вхождение свободных переменных формального предиката. Снова действует правило: никакое свободное вхождение переменной после преобразования не должно оказаться связанным. Если такое получается, соответствующую кванторную переменную надо изменить.

Теорема 4.3. Если $X \equiv Y$, то и $X(p \,||\, x_1 \dots x_k Z) \equiv Y(p \,||\, x_1 \dots x_k Z)$. Если $Y \equiv Z$, то и $X(p \,||\, x_1 \dots x_k Y) \equiv X(p \,||\, x_1 \dots x_k Z)$.

◀ Первое утверждение сводится к следующему: если U — тавтология, то и $U(p \,||\, x_1 \dots x_k Z)$ — тавтология (достаточно в качестве U взять формулу $X \sim Y$). То, что U — тавтология, означает, что при любой интерпретации и любой оценке формула U истинна. Интерпретация, в частности, обеспечивает конкретное содержание предикатного символа p, а формальный предикат $x_1 x_2 \dots x_k Z$ превращается в некоторый предикат $q(x_1, \dots, x_k, y_1, \dots, y_l)$. При замене происходит замена одного предиката другим, при этом появляются свободные переменные y_1, …, y_l, которые мы можем зафиксировать с помощью оценки. В результате дело фактически сводится к замене предикатной буквы p на предикатную букву q. Такую замену можно интепретировать как изменение интерпретации буквы p. При изменении интерпретации формула остается истинной. Значит, и после замены формула остается истинной.

Доказательство второго утверждения аналогично. Замена предикатного символа p одним из формальных предикатов можно интерпретировать как выбор интепретации символа p. Так как формулы Y и Z эквивалентны, то и первая, и вторая замены дают одну и ту же интерпретацию символу p. Следовательно и истинностное значение двух формул будет одинаковым. ▶

Следствие 4.1. Если $Y \equiv Z$, то после замены в формуле X одного из вхождений подформулы Y формулой Z получим формулу X', эквивалентную X.

◀ Пусть x_1, x_2, …, x_n — полный список переменных, входящих в формулы Y и Z. Рассмотрим формулу Γ, полученную заменой подформулы X элементарной формулой $p(x_1, x_2, \dots, x_n)$. Тогда $X = \Gamma(p \,||\, x_1 \dots x_k Y)$ и $X' = \Gamma(p \,||\, x_1 \dots x_k Z)$. Согласно теореме эти формулы эквивалентны. ▶

Утверждение следствия известно как ***правило замены эквивалентным***.

Пример 4.3. Покажем, что формулы $r(x) \vee \forall z\,(\neg \forall x\, q(x,z))$ и $r(x) \vee \forall z(\exists x\, \neg q(x,z))$ эквивалентны.

Обе формулы можно рассматривать как две подстановки в формулу $X = r(x) \vee \forall z\, p(z)$ двух эквивалентных формул $V = \neg \forall x\, q(x,z)$ и $W = \exists x\, \neg q(x,z)$. Сформируем два формальных предиката zV и zW. В результате получим $X(p \,||\, zV) = r(x) \vee \forall z\,(\neg \forall x\, q(x,z))$ и $X(p \,||\, zW) = r(x) \vee \forall z(\exists x\, \neg q(x,z))$. В соответствии с последним логическим законом эти формулы эквивалентны.

Отметим, что было бы неверно в качестве шаблона использовать, например, формулу $X_1 = r(x) \vee \forall z\, p$, опуская в заменяемом символе связанную переменную z. В этом случае в соответствии с индуктивным правилом мы получили бы формулы $X_1(p \,||\, zV) = r(x) \vee \forall z_1\,(\neg \forall x\, q(x,z))$ и $X_1(p \,||\, zW) = r(x) \vee \forall z_1(\exists x\, \neg q(x,z))$, которые тоже эквивалентны, но отличаются от того, что мы хотели бы получить.

На формулы алгебры предикатов, не содержащие импликации, распространяется принцип двойственности. Формулу X^*, двойственную данной формуле X определим индуктивно:

1) если X — элементарная формула, то $X^* = X$;
2) если $X = \neg U$, то $X^* = \neg U^*$;
3) если $X = U \wedge V$, то $X^* = U^* \vee V^*$;
4) если $X = U \vee V$, то $X^* = U^* \wedge V^*$;
5) если $X = \forall x U$, то $X^* = \exists x U^*$;
6) если $X = \exists x U$, то $X^* = \forall x U^*$.

Непосредственно из определения следует, что преобразование двойственности нильпотентно: $X^{**} = X$. Следующее утверждение связывает преобразование двойственности с отношением эквивалентности. Кроме того, верно следующее утверждение. Назовем внутренним отрицанием $\check{X}$ формулы X ее преобразование, при котором каждая элементарная подформула $p(t_1, t_2, \ldots, t_n)$ заменяется формальным предикатом $x_1 x_2 \ldots x_n \neg p(x_1, x_2, \ldots, x_n)$.

Теорема 4.4. Формула X^*, двойственная формуле X, эквивалентна формуле $\neg\check{X}$.

◀ Доказательство проводится индукцией по построению формулы. Для элементарных формул внутреннее отрицание совпадает с обычным отрицанием $\neg$, так что $\neg\check{X} = X$, но в то же время по определению $X^* = X$.

Если $X = U \wedge V$ и уже доказано, что $U^* \equiv \neg\check{U}$, $V^* \equiv \neg\check{V}$, то согласно определению двойственной формулы и правилу замены эквивалентным

$$X^* = U^* \vee V^* \equiv \neg\check{U} \vee \neg\check{V} \equiv \neg(\check{U} \wedge \check{V}) = \neg\check{X}.$$

Аналогично утверждение доказывается для дизъюнкции и отрицания.

Если $X = \forall x U$ и уже доказано, что $U^* \equiv \neg\check{U}$, то

$$X^* = \exists x U^* \equiv \exists x \neg\check{U} = \neg\forall x \check{U} = \neg\check{X}.$$

Аналогичны рассуждения для кванторной приставки $\exists x$. ▶

Теорема 4.5. Эквивалентность $X \equiv Y$ равносильна эквивалентности $X^* \equiv Y^*$.

◀ Доказательство в силу нильпотентности достаточно провести в одну сторону. При этом достаточно заметить, что если $X \equiv Y$, то $\check{X} \equiv \check{Y}$ и $\neg X \equiv \neg Y$. Поэтому в силу предыдущей теоремы

$$X^* \equiv \neg\check{X} \equiv \neg\check{Y} \equiv Y^*. \qquad ▶$$

4.7. Упрощение формул

Приведение бескванторной формулы к ДНФ и КНФ. Предваренная нормальная форма. Теорема о приведении. Пример: $X = \forall x \neg\exists y\, p(x,y) \to \forall x\,(q(x) \to \neg\exists y\, p(x,y))$. Замечание о порядке вынесения кванторов.

На основании логических законов можно преобразовывать формулы логико-математического языка, упрощая или добиваясь получения формул определенной структуры.

Например, если в формуле отсутствуют кванторы (бескванторная формула), то ее можно привести к ДНФ или КНФ. Такое преобразование всегда можно проводить в рамках языка логики высказываний, рассматривая любую элементарную подформулу как пропозициональную переменную.

Теорема 4.6. Всякая бескванторная формула эквивалентна некоторой ДНФ и некоторой КНФ.

Разумеется, подобные преобразования возможны и для формул, содержащих кванторы. Наличие кванторов лишь усложняет сам процесс, но не отменяет его.

Кванторы не мешают преобразованиям, если они в формуле собраны вместе. Формулу вида $\nabla x_1 \nabla x_2 \ldots \nabla x_n X$, где X — бескванторная формула, называют ***предваренной*** или ***пренексной***. В частности, любая бескванторная формула является предваренной. Предваренной (пренексной) формой данной формулы X называют любую предваренную формулу Y, логически эквивалентную X. Часть формулы Y без кванторных приставок называют ***матрицей предваренной формы***.

Теорема 4.7 (о предваренной форме). Любая форма имеет предваренную форму.

◄ Для доказательства нужно исходную формулу X заменить формулой X' со свойством чистоты переменных, а затем, применяя логические законы одностороннего пронесения кванторов, „вытащить“ все кванторы за пределы формулы.

Аккуратно доказательство следует проводить методом индукции по построению формулы. В самом деле, утверждение очевидно для элементарных формул (они все бескванторные). Пусть $X = V \circ W$, где $\circ$ — одна из двуместных логических связок, и формулы V и W имеют предваренные формы: $V \equiv \nabla v_1 \dots \nabla v_m\, V'$, $W \equiv \nabla w_1 \dots \nabla w_k\, W'$, где V' и W' бескванторные. Полагая, что X обладает свойством чистоты переменных, заключаем, что все переменные v_i и w_j различны, V' не содержит переменных w_j, а W' — переменных v_i. Тогда с помощью законов одностороннего пронесения кванторов получаем

$$X \equiv (\nabla v_1 \dots \nabla v_m\, V') \circ (\nabla w_1 \dots \nabla w_k\, W') \equiv \nabla v_1 \dots \nabla v_m\, \nabla w_1 \dots \nabla w_k\, (V' \circ W').$$

Здесь мы воспользовались также правилом замены подформулы эквивалентной.

Если формула X имеет вид $X = \neg V$, рассуждения аналогичны. Наконец, в случае $X = \nabla z\, V$ достаточно заменить V предваренной формой, чтобы получить предваренную форму X. ►

Пример 4.4. Рассмотрим формулу $X = \forall x\, \neg\exists y\, p(x,y) \to \forall x\, (q(x) \to \neg\exists y\, p(x,y))$. Она не обладает свойством чистоты. Используя переименование с помощью двух новых переменных u и v, получим конгруэнтную (а потому эквивалентную) формулу

$$X' = \forall x\, \neg\exists y\, p(x,y) \to \forall u\, (q(u) \to \neg\exists v\, p(u,v)).$$

Применяя одностороннее пронесение кванторов, последовательно получаем:

$$\begin{aligned}
&\forall x\, \neg\exists y\, p(x,y) \to \forall u\, (q(u) \to \neg\exists v\, p(u,v)) \equiv \\
\equiv\ &\forall x \forall y\, \neg p(x,y) \to \forall u\, (q(u) \to \forall v\, \neg p(u,v)) \equiv \\
\equiv\ &\forall x \forall y\, \neg p(x,y) \to \forall u \forall v\, (q(u) \to \neg p(u,v)) \equiv \\
\equiv\ &\exists x \exists y\, \big(\neg p(x,y) \to \forall u \forall v\, (q(u) \to \neg p(u,v))\big) \equiv \\
\equiv\ &\exists x \exists y \forall u \forall v\, \big(\neg p(x,y) \to (q(u) \to \neg p(u,v))\big).
\end{aligned}$$

Замечание. Вынесение кванторов может проходить в разном порядке. Следовательно, и в предваренной форме кванторные приставки могут идти в разном порядке: предваренная форма определена неоднозначно. В матрице предваренной формы все связки идут в том же порядке и без изменений. Она получается из исходной формулы удалением всех кванторных приставок.

ОГЛАВЛЕНИЕ

**Московский государственный технический университет
имени Н.Э. Баумана**

Факультет «Фундаментальные науки»
Кафедра «Математическое моделирование»

А.Н. Канатников

ДИСКРЕТНАЯ МАТЕМАТИКА

Конспект лекций

Для студентов специальности
«Прикладная математика»

Москва
2006

5. ИСЧИСЛЕНИЕ ПРЕДИКАТОВ

5.1. Построение теории P

Язык алгебры предикатов как расширение языка алгебры высказываний. Аксиомы исчисления предикатов: схемы аксиом исчисления высказываний плюс 4 аксиомы с кванторами: 11) $\forall x\, X \to X_t^x$; 12) $\forall x\,(Y \to X(x)) \to (Y \to \forall x\, X(x))$; 13) $X_t^x \to \exists x\, X$; 14) $\forall x\,(X(x) \to Y) \to (\exists x\, X(x) \to Y)$. Правила вывода: 1^* $\frac{X,\ X \to Y}{Y}$; 2^* $\frac{X}{\forall x\, X}$. Вывод и вывод из гипотез. Структурное ограничение. Секвенции.

Исчисление предикатов строится как расширение исчисления высказываний. Основная цель та же: формализовать доказательство различного рода утверждений. Исчисление предикатов — составная часть любой формальной теории, отвечающая за логическую часть теории. Если говорить об алгебре предикатов как таковой, уместно ограничиться простейшим вариантом логико-математического языка, оставляя за кадром детали предметной области. Для нас несущественно, как образуются термы, поскольку они целиком относятся к предметной области. Мы под предметной переменной можем понимать любой терм. Кроме того, для нас не является существенным, сколько сортов предметных переменных есть в теории. Можем считать, что один сорт. В то же время мы не можем совсем игнорировать предметные переменные, поскольку именно с ними связано функционирование кванторов.

Учитывая это, будем строить теорию $\boldsymbol{P}$, языком Ω которой является односортный логико-математический язык с пустым множеством констант и пустым множеством функциональных символов. При этом каждый терм этого языка будет элементарным, т.е. предметной переменной единственного сорта. Поскольку подстановка в формуле связана с предметными переменными, то в нашем языке подстановка есть замена одних предметных переменных другими.

Упрощается и интерпретация рассматриваемого языка. Достаточно задать единую предметную область для всех предметных переменных и указать отображение, которое n-арному предикатному символу ставит в соответствие булеву функцию от n переменных, а в случае 0-арного предикатного символа символ 0 или 1.

Зададим аксиомы исчисления предикатов в виде некоторого набора схем аксиом. Первые десять схем повторяют схемы исчисления высказываний:

1) $X \to (Y \to X)$;
2) $X \to Y \to (X \to (Y \to Z) \to (X \to Z))$;
3) $X \wedge Y \to X$;
4) $X \wedge Y \to Y$;
5) $(X \to Y) \to ((X \to Z) \to (X \to Y \wedge Z))$;
6) $X \to X \vee Y$;
7) $Y \to X \vee Y$;
8) $X \to Z \to (Y \to Z \to (X \vee Y \to Z))$;
9) $X \to Y \to (\neg Y \to \neg X)$;
10) $\neg\neg X \to X$;
11) $X \to \neg\neg X$.

Еще четыре схемы связаны с кванторами (ниже X_t^x — правильная подстановка терма t вместо переменной x, Y не содержит свободных вхождений переменной x):

12) $\forall x\, X \to X_t^x$;
13) $\forall x\,(Y \to X(x)) \to (Y \to \forall x\, X(x))$;
14) $X_t^x \to \exists x\, X$;
15) $\forall x\,(X(x) \to Y) \to (\exists x\, X(x) \to Y)$.

В дополнение к правилу modus ponens добавляется правило обобщения:

1^* $\dfrac{X,\ X \to Y}{Y}$;

2^* $\dfrac{X}{\forall x\, X}$.

Как и в исчислении высказываний выводом называем последовательность формул языка, в которой каждая формула либо аксиома, либо получена по одному из правил вывода из предшествующих формул. Также вводим понятие вывода из гипотез Γ, в котором в последовательности формул могут находиться формулы из списка Γ. Однако для вывода из гипотез введем дополнительное структурное требование: если формула $\forall x\, X$ получена по правилу обобщения, то во всех гипотезах, используемых для вывода этой формулы, переменная x не должна иметь свободных вхождений. Указанное структурное требование не касается аксиом. Поэтому в выводе без гипотез его можно не учитывать.

Смысл сформулированного структурного ограничения становится понятным, если учесть, что вывод должен сохранять истинность формул. Правило модус поненс сохраняет истинность формул при фиксированной модели и фиксированной оценке, а именно: если θ — оценка формулы $X \to Y$ в данной модели M, то из истинности $X\theta$ и $(X \to Y)\theta$ следует истинность $Y\theta$. Правило обобщения нарушает фиксированность оценки при выводе.

Впрочем, структурное ограничение касается лишь частного вывода. Мы могли бы это ограничение не учитывать, но тогда нам придется считаться с ним при „сшивке“ частных выводов в окончательный вывод без гипотез. Поясним сказанное. Выводимость из X формулы $\forall x\, X$ следует увязать с выводимостью формулы $X \to \forall x\, X$, т.е. с тем, что эта формула есть тавтология. Но если переменная x входит в X свободно, то при некоторой интерпретации для некоторого объекта a формула $X^x_{\underline{a}}$ будет ложной, а для некоторого объекта b формула $X^x_{\underline{b}}$ — истинной. В этом случае $\forall x\, X$ в соответствии с интерпретацией всеобщности будет ложной формулой при любой оценке, а это приведет к ложности $X \to \forall x\, X$. Однако эта формула будет тавтологией, если X не имеет свободных вхождений x.

Мы сохраняем запись $\Gamma \vdash X$, означающую, что X выводима из гипотез Γ. Эту запись называем ***секвенцией***.

Пример 5.1. Пусть переменная y не имеет свободных вхождений в X. Тогда $\vdash \forall x\, X \to \to \forall y\, (X^x_y)$. Действительно:

1) $\forall x\, X \to X^x_y$ (схема аксиом 14);
2) $\forall y\, (\forall x\, X \to X^x_y)$ (по правилу обобщения из 1);
3) $\forall y\, (\forall x\, X \to X^x_y) \to (\forall x\, X \to \forall y\, X^x_y)$ (схема аксиом 12);
4) $\forall x\, X \to \forall y\, X^x_y$ (по правилу модус поненс из 3 и 4);

5.2. Правила естественного вывода

Теорема о дедукции. Правила естественного вывода исчисления высказываний. Новые правила:

р1) $\frac{\Gamma \vdash X}{\Gamma \vdash \forall y\, X^x_y}$ (введение общности);
р2) $\frac{\Gamma \vdash \forall x\, X}{\Gamma \vdash X^x_t}$ (удаление общности);
р3) $\frac{\Gamma \vdash X^x_t}{\Gamma \vdash \exists x\, X}$ (введение существования);
р4) $\frac{\Gamma, X \vdash Y}{\Gamma, \exists y\, X^x_y \vdash Y}$ (удаление существования).

Пример: $\exists x\, X \to \neg\forall x(\neg X)$.

Как и в случае исчисления высказываний, на практике вывод строится с помощью правил естественного вывода. Сохраняются структурные правила естественного вывода, так как они не связаны с сутью самого вывода. Сохраняется теорема о дедукции. Однако в доказательство этой теоремы необходимо внести некоторые коррективы.

Теорема 5.1. Если $\Gamma, X \vdash Y$, то $\Gamma \vdash X \to Y$.

◀ Доказательство теоремы, как и в исчислении высказываний, проводится индукцией по построению формулы. Рассуждения в основном повторяют доказательство теоремы из исчисления высказываний. Дополнительно надо лишь рассмотреть случай, когда конечная формула вывода получена по правилу обобщения.

Итак, пусть в выводе $X_1, X_2, \ldots, X_i = X, \ldots, X_j, \ldots, X_n$ конечная формула имеет вид $X_n = \forall x X_j$. Предположим в выводе формулы X_j формула X не используется. Тогда $\Gamma \vdash X_j$ и

$\Gamma \vdash \forall x\, X_j$. С помощью аксиомы $\forall x\, X_j \to (X \to \forall x\, X_j$ с помощью правила заключения получаем $\Gamma \vdash X \to \forall x\, X_j$. Если формула X используется в выводе формулы X_j, то, согласно структурному ограничению, формула X не имеет свободных вхождений переменной x. Согласно индуктивному предположению из вывода $\Gamma, X \vdash X_j$ вытекает вывод $\Gamma \vdash X \to X_j$, откуда по правилу обобщения (ни X, ни X_j не содержат свободных вхождений x) получаем $\Gamma \vdash \forall x\,(X \to X_j)$. Присоединив аксиому $\forall x\,(X \to X_j) \to (X \to \forall x\, X_j)$, получаем вывод формулы $X \to \forall x\, X_j = X \to Y$. ▶

Теорема о дедукции позволяет без изменений перенести в исчисление предикатов все восемь логических правил естественного вывода, а вслед за ними и дополнительные правила естественного вывода: доказательства этих правил не используют правила обобщения. Кроме того, добавляются следующие четыре правила с кванторами (в этих правилах x не имеет свободных вхождений в формулы списка Γ и Y; если y отлично от x, то y не входит свободно в X):

р1) $\dfrac{\Gamma \vdash X}{\Gamma \vdash \forall y\, X_y^x}$ (введение общности); р3) $\dfrac{\Gamma \vdash X_t^x}{\Gamma \vdash \exists x\, X}$ (введение существования);

р2) $\dfrac{\Gamma \vdash \forall x\, X}{\Gamma \vdash X_t^x}$ (удаление общности); р4) $\dfrac{\Gamma, X \vdash Y}{\Gamma, \exists y\, X_y^x \vdash Y}$ (удаление существования).

При выводе формулы р1 учтем, что формулы списка Γ не содержат переменной x:

$$\dfrac{\dfrac{\Gamma \vdash \forall x X}{\Gamma \vdash X} \qquad \dfrac{\vdash \forall y(\forall x X \to X_y^x) \to (\forall x X \to \forall y X_y^x)}{\Gamma \vdash \forall x X \to \forall y X_y^x}}{\Gamma \vdash \forall y X_y^x} \qquad \dfrac{\dfrac{\dfrac{\Gamma \vdash \forall y(\forall x X \to X_y^x)}{\vdash \forall y(\forall x X \to X_y^x)}}{\vdash \forall x X \to X_y^x}}{}$$

Формула р2 непосредственно следует из аксиомы 12:

$$\dfrac{\Gamma \vdash \forall x X \qquad \vdash \forall x X \to X_t^x}{\Gamma \vdash X_t^x}$$

Аналогично выводится и формула р3:

$$\dfrac{\Gamma \vdash X_t^x \qquad \vdash X_t^x \to \exists x X}{\Gamma \vdash \exists x X}$$

Наиболее длинный вывод получается для правила р4:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, X \vdash Y}{\Gamma \vdash X \to Y}}{\Gamma \vdash \forall x(X \to Y)} \qquad \dfrac{\dfrac{\forall x(X \to Y) \vdash X_y^x \to Y}{\forall x(X \to Y) \vdash \forall y(X_y^x \to Y)}}{\forall x(X \to Y) \vdash \forall x(X \to Y)} \qquad \vdash \forall x(X \to Y) \to (X \to Y)_y^x}{\Gamma \vdash \forall y(X_y^x \to Y)} \qquad \vdash \forall y(X_y^x \to Y) \to (\exists y X_y^x \to Y)}{\Gamma \vdash \exists y X_y^x \to Y}}{\Gamma, \exists y X_y^x \vdash \exists y X_y^x} \qquad \dfrac{\Gamma, \exists y X_y^x \vdash \exists y X_y^x \to Y}{}}{\Gamma, \exists y X_y^x \vdash Y}$$

Использование правил естественного вывода такое же, как и в случае исчисления высказываний.

Пример 5.2. Выведем $\exists x\, X \to \neg\forall x(\neg X)$. Имеем цепочку преобразований „снизу вверх“:

$$\exists x\, X \to \neg\forall x(\neg X) \;\Leftarrow\; \exists x\, X \vdash \neg\forall x(\neg X) \;\Leftarrow\; X_y^x \vdash \neg\forall x(\neg X) \;\Leftarrow\; \forall x(\neg X) \vdash \neg X_y^x.$$

Последняя секвенция вытекает из правила p2.

5.3. Глобальные свойства теории P

Лемма об истинности при фиксированной оценке и ее следствие.

Введенное нами понятие вывода из гипотез сохраняет истинность формул при фиксированной модели и фиксированной оценке.

Лемма 5.1. Пусть Γ — список формул $X_1, X_2, \ldots, X_m$ и $\Gamma \vdash Y$. Если для данной модели M и данной оценки θ формул $X_1, X_2, \ldots, X_m, Y$ в M истинны $X_1\theta, X_2\theta, \ldots, X_m\theta$, то и $Y\theta$ в M истинна.

◀ Доказательство проводится индукцией по длине вывода. Утверждение очевидно, если вывод содержит одну формулу, которая в этом случае есть либо аксиома (являющаяся логическим законом), либо гипотеза X_j. Но тогда $Y\theta$ и $X_j\theta$ — одно и то же.

Предположим утверждение доказано для всех выводов длины менее k. Рассмотрим произвольный вывод $Z_1, Z_2, \ldots, Z_k$ из гипотез Γ длины k с конечной формулой $Y = Z_k$. Формула Y есть либо логический закон, либо гипотеза, либо получена по правилу модус поненс, либо по правилу всеобщности. В первых двух случаях рассуждения те же, что и при $k = 1$. В третьем случае в выводе есть формулы Z_j и $Z_l = Z_j \to Y$. В силу предположения индукции формулы $Z_j\theta$ и $(Z_j \to Y)\theta$ истинны в M. В этом случае правило модус поненс гарантирует истинность в M формулы $Y\theta$.

Если Y получена по правилу обобщения из формулы Z_j для некоторого j, то $Y = \forall x Z_j$. В силу структурного требования формула Z_j получена из гипотез $X_{i_1}, \ldots, X_{i_s}$, каждая из которых не имеет свободных вхождений переменной x. Для всякого объекта a того же сорта, что и x из истинности формул X_i следует истинность $(X_i)_{\underline{a}}^x$ (это то же самое, что и X_i). В силу индуктивного предположения заключаем, что $M \models ((Z_j)_{\underline{a}}^x)\theta$, а в силу интерпретации квантора всеобщности это равносильно истинности $(\forall x\, X)\theta$, т.е. $Y\theta$. ▶

Следствие 5.1. Если $\Gamma \vdash X$ и все формулы в Γ являются логическими законами, то и X есть логический закон. В частности, если $\vdash X$, то X — логический закон.

ОГЛАВЛЕНИЕ

**Московский государственный технический университет
имени Н.Э. Баумана**

Факультет «Фундаментальные науки»
Кафедра «Математическое моделирование»

А.Н. Канатников

ДИСКРЕТНАЯ МАТЕМАТИКА

Конспект лекций

Для студентов специальности
«Прикладная математика»

Москва
2006

6. АЛГОРИТМЫ НА ГРАФАХ

6.1. Введение

Определение графа и орграфа. Связь с отношениями. Вершины и ребра (дуги). Изображение и представления графов. Подграфы. Остовные подграфы. Порожденные подграфы. Отношение порядка на подграфах и максимальные подграфы. Связность и компоненты связности. Гомоморфизм и изоморфизм графов. Инварианты. Степени вершин. Группа автоморфизмов графов.

Графом (неориентированным графом) называют пару множеств $(V,\ E)$, где произвольное (конечное) множество, элементы которого называются ***вершинами*** графа, а E — множество неупорядоченных пар (т.е. двухэлементных подмножеств) в множестве V. Элементы множества V называются ***ребрами***. ***Ориентированный граф*** — это пара множеств $(V,\ E)$, где V — множество вершин, а E — множество упорядоченных пар, называемых дугами графа.

Орграф всегда можно интерпретировать как отношение, заданное на множестве вершин, и наоборот, любое отношение имеет интерпретацию как орграф. Неориентированный граф можно рассматривать как специального вида орграф, в котором каждое ребро равнозначно паре взаимно обратных дуг. В этом контексте неориентированный граф связывается с симметричным иррефлексивным отношением: определение неупорядоченной пары как двухэлементного множества не допускает наличия замкнутых ребер, т.е. ребер с одинаковыми концами. В то же время дуги с совпадающими началом и концом — петли — допускаются.

Впрочем, подобные ограничения — всего лишь условность. На практике используются неориентированные графы с петлями, а также графы, в которых две вершины могут быть связаны несколькими ребрами или дугами одного направления (мультиграфы).

Для представления графов используют разные подходы. Например, каждый граф можно описать матрицей, в которой каждая строка соответствует одной вершине, а каждый столбец — одному ребру. Элемент c_{ij} этой матрицы равен 1, если i-я вершина является концом j-го ребра (инцидентна ребру), и 0 в остальных случаях. Аналогично для ориентированного графа c_{ij} принимает значение 1, если i-я вершина является начальной для j-й дуги, значение -1, если i-я вершина является конечной для j-й дуги, и 0 в остальных случаях. Если j-я дуга является петлей, то в этом столбце все элементы равны нулю. Указанная матрица называется ***матрицей инциденций***.

В матрице инциденций орграфа сумма элементов столбца равна нулю. Сумма элементов i-й строки равна разности количества выходящих из вершины дуг и количество входящих в нее дуг. Две эти характеристики называются соответственно ***полустепенью исхода*** и ***полустепенью захода***. Для неориентированного графа сумма элементов столбца равна двум, а сумма элементов строки равна количеству ребер, инцидентных вершине, которое называется ***степенью вершины***.

Обычно матрица инциденций имеет большие размеры и на практике используется редко. Чаще используют другой вариант матричного представления графа. Для неориентированного графа с n вершинами составим квадратную матрицу порядка n, элемент c_{ij} которой равен 1, если i-я и j-я вершины связаны ребром, и 0 в остальных случаях. Для орграфа составляется аналогичная матрица, в которой $c_{ij} = 1$, если есть дуга из i-й вершины в j-ю, и $c_{ij} = 0$ в остальных случаях. Указанная матрица называется ***матрицей смежности***.

В информационных системах представление графа напрямую зависит от поддерживаемых типов данных и способов их представления в электронной памяти. Один из распространенных способов опирается на структуры данных, называемые списками. Список представляет собой

массив произвольно расположенных в памяти элементов, связанных друг с другом посредством ссылок. Различают однонаправленные списки, в которых i-й элемент списка содержит ссылку (адрес) на следующий $(i+1)$-й элемент списка, и двунаправленные списки, в которых каждый элемент содержит ссылки на следующий и предыдущий элементы. Чаще всего элементы списка одинаковы по структуре, но это необязательно.

На основе списков можно использовать следующий подход к представлению орграфа из n вершин. Составляется массив из n элементов, описывающих вершины орграфа. Этот массив называют ***массивом лидеров***, хотя в действительности он может быть списком, а не массивом. Каждый элемент массива лидеров содержит ссылку на список, называемый ***списком смежности***. Каждый элемент списка смежности i-го элемента массива лидеров отражает вершину орграфа, в которую из i-й вершины ведет дуга. Этот способ может использоваться и для описания неориентированных графов.

Пусть $G = (V,\ E)$ — граф (орграф). Граф (орграф) $G_1 = (V_1,\ E_1)$ называется ***подграфом*** графа G, если $V_1 \subset V$ и $E_1 \subset E$. Подграф собственный, если одно из включений строгое. Если $V = V_1$ (множество вершин одно и то же), то G_1 называется ***остовным подграфом*** графа G.

Пусть $G = (V,\ E)$ — граф (орграф) и $V_1 \subset V$. В множестве E выберем множество E_1 всех тех ребер (дуг), оба конца которых попадают в V_1. Пара $(V_1,\ E_1)$ составляет подграф графа (орграфа) G, который называют ***подграфом, порожденным множеством*** V_1.

Пусть G_1 — подграф графа (орграфа) G, а G_2 — подграф G_1. Тогда G_2 будет подграфом и G. В этой ситуации мы называем G_2 подчиненным G_1 и пишем $G_2 \subset G_1$. В результате на множестве всех подграфов G возникает отношение порядка. Имея в виду это отношение, мы для любой совокупности подграфов графа (орграфа) G можем говорить о максимальном подграфе, как о подграфе, который не содержится ни в каком подграфе той же совокупности (кроме самого себя). В ряде случаев понятия „наибольший подграф“ и „максимальный подграф“ совпадают.

Последовательность $v_0, v_1, \ldots, v_n$ вершин графа называется ***цепью***, если для любого $i = \overline{1,\ n}$ вершины v_{i-1} и v_i соединены ребром (являются смежными). Цепь можно интерпретировать и как последовательность ребер, в которой любые два ребра смежные (правда, надо оговорить, что стыковочные вершины данного ребра с левым и правым соседями должны быть разные). Цепь простая, если в ней все вершины разные. Цепь $v_0, v_1, \ldots, v_n$, в которой $v_0 = v_n$ (концы совпадают) называется циклом. Хотя обычно фиксируется начальная точка цикла, удобно считать, что в качестве таковой может быть любая вершина в последовательности, т.е. мы не различаем, например, циклы $v_0, v_1, \ldots, v_n, v_0$ и $v_1, \ldots, v_n, v_0, v_1$. Цикл простой, если в нем все вершины разные. Правда, следует исключить из понятия „простой цикл“ последовательность вида v_0, v_1, v_0, в которой одно и то же ребро проходится дважды.

Для орграфов вводятся аналогичные понятия ***пути*** $v_0, v_1, \ldots, v_n$, в котором для любого $i = \overline{1,\ n}$ из вершины v_{i-1} в вершину v_i ведет дуга, и контура — пути, у которого начало и конец совпадают, а также простого пути и простого контура.

Граф G связный, если любые две его вершины можно соединить цепью. На произвольном графе можно построить ***отношение достижимости***, содержащее все пары вершин, которые можно соединить цепью. Легко показать, что на неориентированном графе это отношение является отношением эквивалентности. Классы эквивалентности по этому отношению называются ***компонентами связности***. Каждая компонента связности представляет собой связный подграф. Связный граф содержит одну компоненту связности, несвязный — несколько. Компоненту связности можно также охарактеризовать как максимальный связный подграф.

Для орграфов используют три варианта понятия связности. Орграф сильно связный, если для любой пары $(v,\ w)$ вершин есть путь из v в w и есть путь из w в v. Орграф связный, если для любой пары $(v,\ w)$ его вершин есть путь из v в w или из w в v. Орграф слабо связный, если ***ассоциированный неориентированный граф***, получающийся заменой каждой дуги ребром, является связным. Из сильной связности вытекает связность, а из последней — слабая связность. Для каждого из трех видов связности можно ввести соответствующее отношение: от-

ношение сильной достижимости, отношение достижимости и отношение слабой достижимости. Из трех отношений два являются отношениями эквивалентности. Классы эквивалентности по отношению сильной достижимости называются компонентами сильной связности или ***бикомпонентами***. Понятие компоненты связности также используют, имея в виду максимальный связный подграф орграфа.

Пусть $G_1 = (V_1,\ E_1)$ и $G_2 = (V_2,\ E_2)$ — два графа. Отображение φ: $V_1 \to V_2$, удовлетворяющее условию $\{v,\ w\} \in E_1 \Rightarrow \{\varphi(v),\ \varphi(w)\} \in E_2$, называется ***гомоморфизмом графов***. Если гомоморфизм G_1 на G_2 устанавливает взаимно однозначное соответствие между парами множеств V_1 и V_2, E_1 и E_2, его называют ***изоморфизмом графов***.

Аналогично вводятся понятия гомоморфизма и изоморфизма орграфов. Изоморфные графы (орграфы) рассматриваются как идентичные: их различия связаны со способом реализации, но не с внутренней структурой.

Установить изоморфизм двух графов сложной структуры непросто: эффективного алгоритма решения такой задачи нет. Конечно, существует конечное число отображений из V_1 в V_2, и можно проверить все эти отображения. Однако на практике это неприемлемо: объем вычислений с увеличением размера графа растет как факториал и очень быстро оказывается неподъемным.

Во многих случаях можно установить неизоморфность двух графов, сравнивая их характеристики, определяемые внутренней структурой, а точнее, такие характеристики, которые совпадают у любых двух изоморфных графов. Такие характеристики называют ***инвариантами***. Первыми инвариантами являются количество вершин и количество ребер. Другим важным инвариантом является ***вектор степеней вершин***, который представляет собой упорядоченный список степеней вершин. Можно привести и другие инварианты: количество петель, максимальную длину простой цепи (пути). Некоторые инварианты будут приведены позже в связи с анализом конкретных задач. Здесь же отметим еще один инвариант. ***автоморфизм*** графа — это изоморфизм графа на себя. Множество всех автоморфизмов графа непусто, так как включает в себя тождественное отображение множества вершин, и относительно операции композиции отображений образует группу. Она называется ***группой автоморфизмов***. У изоморфных графов группы автоморфизмов изоморфны.

6.2. Деревья

Неориентированное дерево. Критерии: n вершин и $n-1$ ребро (при наличии связности; единственность цепи, соединяющей произвольные две вершины. Ориентированное дерево. Остовное дерево (остов). Пример орграфа без остова. Максимальный остовный лес и его построение с помощью алгоритма поиска в глубину. Алгоритм построения всех остовных деревьев графа. Понятие кратчайшего остовного дерева. Алгоритм Краскала и алгоритм Прима. Задача Штейнера.

Связный ациклический (т.е. без циклов) неориентированный граф называется ***неориентированным деревом*** (далее просто деревом).

Орграф называется ***ориентированным деревом*** (ордеревом), если выполняются условия:

1) существует выделенная вершина v_0, имеющая полустепень захода 0 (корень ордерева);
2) все некорневые вершины имеют полустепень захода 1;
3) орграф не имеет контуров (т.е. является ***бесконтурным***).

Рассмотрим произвольную вершину ордерева. Если это не корень, то для нее есть предшествующая вершина, из которой в рассматриваемому вершину ведет дуга. У предшествующей есть свой предшественник. Эта цепочка рано или поздно прервется. Но это может произойти, только если очередная вершина есть корень. Таким образом, в ордереве в любую вершину из корня ведет путь, причем этот путь единственный (неединственность означает, что есть вершины, полустепень захода которых больше единицы).

Среди вершин ордерева есть такие, степень исхода которых равна нулю. Такие вершины называют ***листьями***. Путь, ведущий из корня в лист не может быть дальше продолжен и по отношению включения является максимальным, т.е. не является частью другого пути.

Все вершины ордерева распадаются на уровни. Глубиной вершины называется длина пути в эту вершину из корня. Один уровень составляют вершины одной глубины. Все дуги графа ведут с какого-либо уровня на следующий. Отметим, что в ордереве из n вершин имеется $n-1$ дуга, поскольку каждой вершине можно поставить в соответствие дугу, входящую в эту вершину. Это соответствие распространяется на все вершины кроме корня. Таких вершин $n-1$. Стало быть, и дуг тоже $n-1$.

Перейдем теперь к случаю неориентированного дерева, имеющего n вершин. Зафиксируем одну из вершин v_0 этого дерева, назвав ее ***корнем***. Тогда каждая вершина графа связывается с корнем единственной простой цепью, длина которой представляет собой ее глубину. Единственность цепи из корня в вершину позволяет, как и в случае ордерева, разделить вершины графа на уровни в соответствии с длиной цепи, связывающей вершину с корнем: единственная вершина уровня 0 — корень v_0, вершины уровня 1 — вершины, смежные с v_0, вершины второго уровня — те, которые соединяются с корнем двузвенной цепью и т.д.

Все ребра дерева ведут с одного уровня на предшествующий или последующий уровень. Любая вершина v имеет ребро, соединяющее эту вершину с вершиной предыдущего уровня, и притом единственное. В самом деле, соединим v с корнем v_0 простой цепью. Любое ребро, инцидентное v, есть либо последнее звено указанной цепи, либо ведет на следующий уровень, поскольку увеличивает длину цепи. Припишем каждому ребру направление в сторону увеличения уровня вершины. Тогда дерево превратится в ордерево с корнем v_0.

Теорема 6.1. Следующие условия эквивалентны:

- граф является деревом;
- граф связен и имеет $n-1$ ребро, где n — количество вершин;
- любые две вершины графа можно соединить простой цепью, и притом единственным способом.

◀ Если граф — дерево, то в силу его связности любые две вершины можно соединить цепью. Существование двух различных цепей, связывающих две вершины v и w, равносильно существованию цикла, составленного из этих двух цепей (или их частей, если цепи частично совпадают). Наоборот, любой цикл можно разделить на две непересекающиеся цепи, соединяющие две вершины. Поэтому третье условие равносильно условию, что граф — дерево.

В дереве выберем произвольную вершину в качестве корня. Тогда дерево превращается в ордерево, как это описано выше. В ордереве каждой вершине, кроме корня, можно поставить в соответствие входящую в эту вершину дугу. Такое соответствие оказывается биективным. Следовательно, количество ребер в дереве должно быть $n-1$.

Докажем, что любой связный граф с n вершинами и $n-1$ ребром есть дерево. Доказательство проведем индукцией по количеству вершин. Утверждение очевидно, например, при $n=2$. Пусть утверждение доказано для всех графов с $n-1$ вершиной. Выберем произвольный связный n-вершинный граф с $n-1$ ребром. У этого графа есть вершина v степени 1. Действительно, в противном случае, если все степени вершин не меньше 2, общая сумма степеней не менее $2n$, а значит, у графа не менее n ребер. Удалим из графа вершину степени 1 и единственное ребро, инцидентное этой вершине. Получим граф из $n-1$ вершины и $n-2$ ребер. Этот граф связный, поскольку удаленная вершина не может быть внутренней вершиной простой цепи, а следовательно, цепь соединяющая две вершины, не проходит через удаленную вершину. В соответствии с предположением индукции он есть дерево. Тогда и исходный граф есть дерево, поскольку не имеет циклов: удаляемое ребро с тупиковым концом не может быть звеном какого-либо цикла. ▶

Пусть G — связный граф. Всякий остовный подграф графа G, являющийся деревом, называется ***остовным деревом***, или ***остовом***.

Теорема 6.2. Всякий связный неориентированный граф имеет остов.

◄ Рассмотрим множество всех подграфов графа G, являющихся деревьями. Это множество конечно и упорядочено по включению. Следовательно, оно имеет максимальный элемент. Покажем, что максимальное поддерево в G является остовным подграфом. Пусть T — подграф в G, являющийся деревом и содержащий $k < n$ вершин. Среди вершин графа G, не включенных в T, можно выбрать вершину v, смежную с одной из вершин дерева T. В самом деле достаточно выбрать цепь, первый конец которой находится в T, а второй — за пределами T. Тогда первая вершина, не попадающая в T будет смежна с предшествующей, относящейся к T. Итак, существует вершина v, не попадающая в T, связанная некоторым ребром $\{v, w\}$ с некоторой вершиной из T. Добавив к T вершину v и ребро $\{v, w\}$, мы получим подграф G, не имеющий циклов, так как добавляемое ребро не может быть звеном цикла.

Итак, мы показали, что любой подграф-дерево, содержащий $k < n$ вершин, не может быть максимальным. Значит, максимальный подграф-дерево содержит n вершин и является остовным. ►

Неориентированным лесом называется граф, любая компонента связности которого является деревом. Лес — это просто ациклический граф.

Следствие 6.1. Любой граф имеет максимальный остовный лес.

◄ Построим для каждой компоненты связности остов. Объединение всех этих остовов будет остовным подграфом-лесом данного графа. Предположив, что он не максимальный, мы приходим к выводу, что к нему можно добавить ребро графа G, сохраняющее свойство ацикличности. Но это ребро будет относиться к одной из компонент связности, и его добавление будет означать, что к остову компоненты можно добавить ребро, сохраняя ацикличность. Но это противоречит максимальности остова. ►

Замечание. Нетрудно увидеть, что любой максимальный лес распадается на компоненты, являющиеся остовами компонент графа. В самом деле, рассуждая как и выше можно показать, что максимальный лес является остовным. Выбрав подграф леса, порожденный множеством вершин какой-либо компоненты графа G, получим максимальный лес для этой компоненты. Но нетрудно увидеть, что максимальность обеспечивает связность, т.е. это будет не максимальный лес этой компоненты, а дерево.

Доказательство существования остова у связного графа (остовного леса в общем случае) на самом деле не является конструктивным и непригодно для построения остова. Чтобы построить остов, можно использовать один из способов обхода всех вершин графа, называемый ***поиском в глубину***. Суть его можно уяснить на такой задаче: просмотреть все каталоги и подкаталоги данного жесткого диска. Структура каталогов жесткого диска имеет древовидную структуру. Например, мы хотим определить суммарную емкость диска, занятую файлами. Задачу можно решить так: начинаем просматривать корневой каталог. Если очередной элемент каталога является подкаталогом, переходим в этот подкаталог и начинаем просматривать его, выбирая все файлы и суммируя их длину. Если подкаталог полностью просмотрен, мы поднимаемся в родительский подкаталог и продолжаем его просмотр с того элемента, на котором он был прерван. Процесс просмотра в целом будет завершен, когда мы закончим просмотр корневого каталога.

Описанная стратегия обхода всех вершин дерева и называется поиском в глубину. В действительности требование, чтобы анализируемый граф был деревом, несущественно: поиск в глубину можно реализовать на любом графе.

Опишем более детально поиск в глубину, предполагая, что граф задан массивом лидеров и списками смежности. Алгоритм обхода вершин будет следующий. В процессе работы мы будем использовать пометки в массиве лидеров, выделяющие уже просмотренные вершины, механизм стека для возврата назад. При старте в массиве лидеров все вершины непомеченные, стек пуст.

1. Из массива лидеров выбираем первую непомеченную вершину как текущую. Если таких нет, переходим к пункту p6. Иначе переходим к следующему пункту.

2. Помечаем текущую вершину, выбираем ее список смежности, устанавливаем в нуль указатель в списке смежности и переходим к следующему пункту.

3. Увеличиваем значение указателя на единицу и проверяем не закончился ли список смежности текущей вершины. Если нет, переходим к следующему пункту. Если закончился, переходим к пункту p5.

4. В списке смежности в соответствии с указателем выбираем очередную вершину. Если эта вершина помечена, пропускаем ее и переходим к пункту p3. Если это непомеченная вершина, записываем в стек номер текущей вершины и значение указателя для нее. В качестве текущей устанавливаем выбранную вершину и переходим к пункту p2.

5. Если стек пуст, переходим к пункту p1. Иначе выбираем из стека номер записанной вершины, делая ее текущей, и номер указателя для ее списка смежности. Затем переходим к пункту p3.

6. Процесс заканчиваем.

При проведении этого алгоритма выделяется часть ребер, по которым проводится переход к новой текущей вершине, которые в совокупности с пройденными вершинами образуют подграф анализируемого графа. Отметим, что каждое из таких ребер проходится ровно один раз в прямом направлении (пункт p2) и один раз в обратном (пункт p5). Если учитывать только прямой ход, то каждая вершина будет однозначно связана с одним входным ребром. Исключение составляет стартовая вершина, вход в которую выполняется на основании пункта p1. При повторном возврате к этому пункту все вершины будут помечены, поскольку граф связный. Поэтому количество ребер в подграфе на единицу меньше количества вершин. Следовательно, этот граф — дерево, причем остовное.

Итак, рассмотренный алгоритм поиска в глубину позволил построить остов связного графа. Однако роль алгоритма поиска в глубину на самом деле гораздо выше, чем поиск остова графа. Он является основой для широкого спектра алгоритмов, решающих различные задачи анализа графов.

Связный ориентированный граф может не иметь остовного ордерева хотя бы потому, что у него есть пара вершин с полустепенью захода 0. Тогда и у любого подграфа будет не менее двух вершин с полустепенью захода 0, а это уже противоречит структуре дерева. Можно рассматривать подграфы-ордеревья данного орграфа и среди них есть максимальные. Просто максимальный подграф-дерево не будет остовным. Если мы применим алгоритм поиска в глубину, мы получим не ордерево (из-за пункта p1), а ориентированный лес. Так что утверждение теоремы остается в силе, но в ослабленном варианте.

6.3. Остов графа наименьшего веса

Поскольку остов неориентированного графа определен неоднозначно, возникает задача выбора остова, оптимального в том или ином смысле. Например, граф представляет собой сеть дорог. Возникает задача прокладки маршрута наименьшей длины, проходящего через данные населенные пункты. Математически подобного рода задача ставится следующим образом. Пусть дан размеченный (взвешенный) граф, т.е. граф, каждому ребру которого приписан вес. Требуется выбрать остов наименьшего веса.

Прежде чем рассматривать конкретные алгоритмы, представим себе задачу перечисления всех остовов связного графа. Процедуру можно представить как последовательное наращивание ребер строящегося дерева. Если выбраны k ребер исходного графа, из которых нельзя составить цикла, мы получаем подграф, являющийся неориентированным лесом. К этому лесу можно добавить еще одно ребро из оставшихся, убедившись, что новое ребро не порождает циклов с остальными. Процедура продолжается до тех пор, пока не наберется $n-1$ ребро. В результате

мы получим граф, содержащий $n-1$ ребро и не имеющий циклов. Если это был бы лес, то количество ребер в нем было бы равно количеству вершин минус количество компонент. Количество $n-1$ может получиться только в том случае, когда подграф остовный и и имеет одну компоненту связности, т.е. является деревом.

На каждом шаге наращивания поддерева существует несколько возможностей добавления очередного ребра. В результате на каждом шаге наше решение разветвляется, и мы фактически получаем так называемое дерево решений. Листья этого дерева, т.е. вершины степени 1, представляют собой конкретные решения. Совокупность всех листьев есть совокупность всех остовов рассматриваемого графа.

Поиск остова наименьшего веса можно представить как оптимальный спуск из корня дерева остовов в лист. На каждом шаге следует принимать решение, какое ребро из допустимых следует выбрать. Особенность этой задачи состоит в том, что оптимальный выбор ребра на каждом шаге гарантирует оптимальность окончательного результата.

В этой процедуре удобно считать, что наращиваемый подграф изначально включает все вершины, а пошаговая процедура состоит в последовательной склейке компонент подграфа-леса с помощью ребер исходного графа. Тогда стартовый граф представляет собой n изолированных вершин. На каждом шаге процесса наращивания добавляется одно ребро и количество компонент уменьшается на одну. На конечном $(n-1)$-м шаге количество добавленных ребер станет $n-1$, а количество компонент сократится до одной.

Рассмотрим два непересекающихся подграфа $T_1 = (V_1,\ E_1)$ и $T_2 = (V_2,\ E_2)$ графа G. Введем для них характеристику

$$\Delta = \Delta(T_1, T_2) = \min_{v_1, v_2} c(v_1; v_2),$$

где $c(v_1; v_2)$ — вес ребра $\{v_1,\ v_2\}$, а минимум берется по всем тем ребрам $\{x,\ y\}$ графа G, для которых $v_1 \in V_1$, $v_2 \in V_2$.

Теорема 6.3. Пусть подграф-лес T, состоит из компонент T_1, T_2, …, T_k и подчинен некоторому остову наименьшего веса. Из компонент T_2, T_3, …, T_k выберем такую компоненту T_j, на которой $\Delta(T_1, T_j)$ минимально, причем эта величина реализуется на ребре γ^*, соединяющем некоторую вершину компоненты T_1 с некоторой вершиной компоненты T_j. Тогда подграф „T плюс γ^*“ подчинен некоторому остову наименьшего веса.

◀ Рассмотрим остовное дерево T' наименьшего веса, содержащее T. В T' есть цепь α, соединяющая концы ребра γ^*. Если само это ребро не входит в T', то его добавление к графу дает цикл $\gamma + \alpha$. В цепи α есть первое ребро γ, выходящее за пределы компоненты T_1 (начиная с того конца цепи, который находится в T_1). По условию выбора вес ребра γ не меньше веса ребра γ^*. Удаление ребра γ разрывает образовавшийся цикл и сохраняет связность T'. Следовательно, замена γ на γ^* приводит к остовному дереву T'', отличающемуся от T' всего одним ребром. Поскольку $c(\gamma^*) \leqslant c(\gamma)$, вес остова T'' не превышает вес остова T' и является наименьшим, т.е. T'' — остов наименьшего веса. ▶

Доказанная теорема является теоретической основой нескольких алгоритмов выбора остова наименьшего веса. Отметим, что принцип выбора, сформулированный в теореме, обладает некоторой свободой: в качестве компоненты T_1 можно выбирать любую из имеющихся. Произвольность такого выбора означает, что и конкретный алгоритм построения остова можно реализовать по-разному. Рассмотрим два таких алгоритма.

Алгоритм Краскала. В этом алгоритме минимизация выполняется по обоим компонентам T_1 и T_j. На самом деле нет нужды перебирать всевозможные пары компонент и определять на какой паре величина Δ минимальна. Можно просто перебрать все ребра графа и среди них выбрать ребро наименьшего веса, которое, во-первых, не входит в T, а во-вторых, не образует цикла с уже имеющимися в T ребрами. Тогда это будет ребро, соединяющее какие-то две компоненты графа T.

Кратко алгоритм Краскала можно описать так.

1. Начинаем с остовного графа T без ребер.

2. Все ребра графа упорядочиваем по возрастанию веса.

3. Выбираем очередное ребро из упорядоченного списка и проверяем, образует ли оно цикл с уже отобранными. Если да, пропускаем его и повторяем пункт. Если нет, переходим к следующему пункту.

4. Проверяем, если отобрано $n-1$ ребро, то процесс прекращаем. Иначе возвращаемся к предыдущему пункту.

В изложенной стратегии два узких места: сортировка ребер по возрастанию и проверка на появление циклов при добавлении очередного ребра. Первая проблема сводится к выбору эффективного алгоритма сортировки. Отметим, что полная сортировка всех ребер на самом деле не нужна. Отбираться будет $n-1$ ребро, в то время как полный граф содержит гораздо больше ребер: $n(n-1)/2$. Сортировку можно свести к выбору на каждом шаге наилучшего из оставшихся ребер. Эта проблема показывает, что алгоритм Краскала наиболее эффективен для графов с небольшим числом ребер (того же порядка, что и число вершин).

Вторая проблема может решаться с помощью специальной системы пометок вершин графа. Поскольку добавляемое ребро должно связывать вершины разных компонент остовного подграфа T, то и проверять надо, принадлежат ли концы ребра разным компонентам. Достаточно каждой вершине приписать специальный ранг, указывающий на компоненту, к которой относится вершина. При выборе очередного ребра отбрасываем варианты, у которых концы имеют одинаковый ранг. Если выбрано ребро, у которого концы имеют ранги r_1 и r_2, то надо просмотреть все вершины и поменять все ранги r_2 на r_1 (или наоборот). Тогда слияние двух компонент в одну будет отражаться в рангах вершин.

Указанную процедуру проверки на циклы можно сделать более эффективной, если в пометку вершин включать не только ранг, но и дополнительную информацию, позволяющую по вершине восстановить все поддерево.

Алгоритм Прима. В этом алгоритме дерево T_1 фиксировано, остальные компоненты до конца остаются одиночными вершинами. Технически эту стратегию можно реализовать следующим образом. Каждой вершине x на текущем шаге соответствует метка, содержащая вершину y дерева T_1, ближайшую к x, и вес β ребра, соединяющего y с x. Если вершина x не имеет ребер, связывающих ее с T_1, тот ей приписывается специальная метка $(0,\ \infty)$. На k-м шаге просматриваются все вершины, не относящиеся к T_1 и среди них выбирается вершина x_k, для которой вес β_k наименьший. Ребро, соединяющее x_k с T_1, заносится в список отобранных ребер. Затем для всех вершин $x \notin T_1$ вес β сравнивается с весом ребра $\{x_k,\ x\}$ и если последнее меньше, метка вершины x обновляется. Процедура останавливается, когда будет отобрано $n-1$ ребер или n вершин.

6.4. Задача о путях в размеченном графе

Отношение достижимости и матрица достижимости. Граф как отношение (отношение смежности) и транзитивное замыкание. Понятие о размеченном графе. Сведение задачи к вычислению итерации матрицы в полукольце. Алгоритм Дейкстры. Алгоритм Флойда.

Граф $G = (V,\ E)$, у которого каждой вершине приписан вес, называется ***размеченным графом***. К размеченным графам может относиться сеть дорог, каждая из которых имеет оценку качества и протяженности. Для такой сети актуальна задача поиска оптимального маршрута. Размеченный граф может обозначать финансовые потоки между различными составляющими бизнес--процесса, информационные потоки в компьютерной сети и т.п.

Задача о путях в размеченном графе может формулироваться в трех вариантах:

- найти кратчайшую цепь, связывающую две заданные вершины графа;
- для заданной вершины v_0 построить кратчайшие цепи, соединяющие v_0 со всеми остальными вершинами графа;

• построить кратчайшие цепи для всех пар вершин графа.

Задача может ставиться как для неориентированных, так и для ориентированных графов. Под весом цикла (пути) понимается сумма весов всех ребер (дуг), входящих в цепь (путь). Однако важно то, что понятие „сумма“ может меняться. Фактически весами в графе могут быть элементы любого полукольца или кольца. Например, как частный случай поставленной задачи можно рассматривать задачу построения матрицы достижимости для данного графа. Достаточно разметить граф над полукольцом $\mathbb{B}$, причем под суммой понимать произведение в $\mathbb{B}$. Тогда вес каждой цепи будет равен 1, а матрица весов кратчайших путей будет совпадать с матрицей достижимости.

Требование, чтобы веса графа были элементами полукольца, вообще говоря, не является обязательным. Достаточно обеспечить алгоритм, позволяющий по весам звеньев цепи однозначно определить вес всей цепи.

В результате решения общей задачи мы для каждой пары вершин v и w получаем значение наименьшего веса цепи (пути), соединяющей v и w, которое называется ***стоимостью***. Из стоимостей можно составить ***матрицу стоимостей***. Процесс решения задачи состоит в вычислении матрицы стоимостей.

Отметим, что наличие в графе, размеченном над кольцом $\mathbb{R}$, циклов (контуров) отрицательного веса может привести к тому, что задача не будет иметь решения. В этом случае необходимо исключить из рассмотрения такие циклы или по крайней мере ограничить число проходов по ним.

Решение первого и второго вариантов задачи как часть входит в решение третьего варианта. Выделение их как самостоятельных преследует цель выбора специальных более экономичных алгоритмов для их решения.

Остановимся на первом варианте задачи.

Алгоритм Дейкстры. Этот алгоритм рассчитан на случай, когда в графе все веса неотрицательные. Он позволяет для заданной вершины v_0 ориентированного графа построить стоимости $D[v]$ для каждой пары вершин v_0 и v. Алгоритм сводится к построению последовательности $S_1, S_2, \ldots, S_n$ множеств вершин, в которой множество S_{k+1} получается из S_k добавлением одной вершины, и вычислению для каждого множества S_k множества $D_k[v]$ стоимостей, определяемых только по тем путям из v_0 в v, которые не выходят за пределы множества S_k (за исключением конечной вершины v). Начальное множество S_1 состоит из единственной вершины v_0, а массив стоимостей $D_1[v]$ — из весов дуг, связывающих вершину v_0 с вершиной v. При отсутствии дуги, связывающей v_0 с v, полагаем $D_1[v] = \infty$. На каждом шаге алгоритма к множеству S_k добавляется вершина $w = v_{k+1} \notin S_k$, для которой значение $D_k[w]$ является наименьшим. Затем вычисляется массив $D_{k+1}[v]$ с помощью следующей процедуры пересчета. Если $v \in S_{k+1}$, то $D_{k+1}[v] = D_k[v]$. Если же $v \notin S_{k+1}$, то в качестве $D_{k+1}[v]$ выбирается наименьшее из значений $D_k[v]$ и $D_k[v_{k+1}] + \omega(v_{k+1}, v])$, где $\omega(v, w)$ — вес дуги, соединяющей вершины v и w.

Через n шагов алгоритма Дейкстры множество S_k будет содержать все вершины графа, а массив $D_k[v]$ — стоимости пар вершин v_0, v. При реализации на практике массивы $D_k[v]$ можно совместить. При этом на k-м шаге общий массив $D[v]$ будет содержать значения $D_k[v]$. При пересчете значения $D[v]$ для вершин $v \in S_{k+1}$ не изменяются, пересчитываются лишь те, для которых $v \notin S_{k+1}$.

Как видим, алгоритм Дейкстры ориентирован на решение второго варианта задачи о путях. Однако его легко модифицировать для решения и первого варианта задачи: достаточно процесс остановить в тот момент, когда в качестве вершины v_{k+1} будет выбрана заданная конечная вершина пути.

Почему алгоритм Дейкстры приводит к кратчайшему пути. Выбор на каждом шаге точки V_{k+1}, путь к которой (из обследованных в рамках множества S_k) является кратчайшим, позволяет сразу объявить этот путь кратчайшим и во всем графе. Это вытекает из следующих соображений. Пусть на k-м шаге в вершину w ведет кратчайший путь веса c^*, т.е. $D_k[w] = c^*$

и минимальна для всех вершин вне S_k (рис. 6.1). Рассмотрим другой путь, ведущий в w через некоторую вершину v. По условиям выбора $c = D_k[v] \geqslant c^*$. Учитывая, что часть пути из v в w имеет неотрицательный вес, заключаем, что вес альтернативного пути не меньше, чем тот, по которому был вычислен вес $D_k[v]$. Как видим, явно используется условие неотрицательности весов графа. Отметим, что это условие автоматически выполнено, если граф размечен над идемпотентным полукольцом. Можно также условие неотрицательности весов заменить условиекм монотонности: вес любого пути не должен быть меньше веса любой своей части.

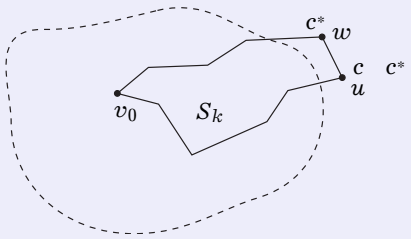


Рис. 6.1

Алгоритм Дейкстры по своей сути — процедура перебора путей в графе для отбора среди них оптимальных. Эту процедуру легко модифицировать так, чтобы можно было получать не только веса оптимальных путей, но и сами эти пути, т.е. пути наименьшего веса. Достаточно в качестве значений массива $D[v]$ рассматривать не стоимости пар вершин, а векторы, первой компонентой которых является вес оптимального пути (стоимость), а остальные компоненты описывают цепочку вершин, представляющую собой оптимальный путь из вершины v_0 в вершину v. На k-м шаге при вычислении $D_{k+1}[v]$ мы либо сохраняем вектор $D_k[v]$, либо к первой компоненте вектора $D_k[v_{k+1}]$ добавляем вес $\omega(v_{k+1}, v])$, а к концу этого вектора присоединяем дугу, соединяющую вершины v_{k+1} и v.

Задача о путях в полукольцах. Во многих случаях в задаче о путях можно считать, что метки дуг (ребер) являются элементами некоторого полукольца. Выбор полукольца позволяет соответствующим образом интерпретировать понятие оптимальности пути.

Считаем, что весом пути является произведение (в полукольце) всех весов дуг, составляющих этот путь, а стоимостью пар вершин — сумма весов всех путей, соединяющих эти вершины. Рассмотрим задачу вычисления матрицы стоимости.

Обозначим через Ω матрицу весов графа. Тогда с учетом правила перемножения матриц заключаем. что Ω^2 содержит стоимости пар вершин, вычисленные по всем путям длины 2, аналогично степень Ω^k содержит стоимости пар вершин, вычисленные по всем путям длины k. Поскольку стоимость пары вершин определяется как сумма весов по всем путям, соединяющим эти вершины, заключаем, что матрица стоимости C определяется формулой

$$C = \Omega^* = E + \Omega + \Omega^2 + \dots$$

Здесь Ω^* — итерация матрицы в рассматриваемом полукольце. Отметим, что в силу конечности графа любой путь длины больше n содержит цикл (контур) и содержит в себе как часть другой путь. Вес пути — произведение всех дуг. Следовательно, часть пути имеет больший вес и при суммировании поглощается. Это значит, что $\Omega^{n+1} \leqslant \Omega^n$. Поэтому

$$C = \Omega^* = E + \Omega + \Omega^2 + \dots \Omega^n.$$

Если граф размечен обычными числовыми весами, достаточно в качестве умножения полукольца рассмотреть сложение действительных чисел, а в качестве сложения полукольца — минимум двух действительных чисел. Чтобы эта операция имела нейтральный элемент, достаточно ввести в рассмотрение вес ∞, обозначая им пары вершин, не связанные дугой графа. Вычисление минимума — коммутативная идемпотентная операция, относительно которой сложение дистрибутивно.

Анализ процедуры непосредственного вычисления Ω^* показывает, что алгоритм сводится к методу динамического программирования.

Из теории полуколец вытекает, что итерация A^* матрицы A может быть получена как решение матричного уравнения $X = AX + E$. Вычисление отдельного j-го столбца x_j матрицы X сводится к матричному уравнению $x_j = Ax_j + e_j$, где e_j — j-й столбец единичной матрицы. А это матричное уравнение может интерпретироваться как матричная форма записи системы линейных уравнений.

Таким образом, вычисление матрицы стоимости сводится к решению нескольких систем линейных уравнений в некотором полукольце. В данном случае можно ограничиться вычислением одного столбца матрицы стоимостей, что равносильно вычислению массива $D[v]$ стоимостей для пар вершин v и v_0. Это можно реализовать как решение одной системы линейных уравнений над полукольцом.

Вычисление отдельных элементов матрицы Ω^* можно свести к решению системы линейных уравнений над полукольцом. Решение такой системы осуществляется методом Гаусса (методом исключения неизвестных). Поскольку в конкретном рассматриваемом приложении требуется лишь один элемент матрицы Ω^* (путь наименьшего веса, связывающий две данные вершины), то процедуру вычисления можно провести так, что в методе Гаусса будет использоваться только прямой ход (в системе нужно будет исключать последовательно все неизвестные, кроме того, которое определяется).

Прямое вычисление итерации матрицы весов (например, в случае, когда сложение — это минимум, а умножение — это обычное сложение) можно представить как последовательный пересчет матрицы весов по формуле

$$c'_{ij} = \min\left\{c_{ij},\ \min_{k=\overline{1,n}}(c_{ik} + w_{kj})\right\}, \tag{6.1}$$

где c_{ij} — найденные стоимости, а w_{ij} — веса ребер. С одной стороны, видно, что это позволяет сравнить уже найденный минимальный маршрут длины m с маршрутом, проходящим через вершину k и тем самым получить минимальный маршрут длины $m + 1$. С другой стороны, в терминах полукольца указанный пересчет записывается так:

$$c'_{ij} = c_{ij} + \sum_{k=1}^{n} c_{ik}w_{kj},$$

что в матричной форме равносильно равенству $C' = C + CW$. В итерационной процедуре $C_0 = E$, $C_{k+1} = C_k + C_kW$ мы через n шагов получим итерацию W^* матрицы весов.

Описанный процесс вычисления итерации можно улучшать в разных направлениях. Нетрудно заметить, что итерацию можно получить по итерационной формуле $C_{k+1} = E + C_kW$. Правда, с точки зрения объема вычислений это не дает преимуществ. Преимущество дает такой процесс: $C_1 = E + W$, $C_2 = C_1^2$, $C_4 = C_2^2$, $C_8 = C_4^2$ и т.д. Для вычисления итерации W^* вместо n произведений достаточно вычислить $\log_2 n$ квадратов.

Еще один прием связан с нарушением структуры матричных произведений (все матричные многочлены вычисляются параллельно). Оказывается, что для построения матрицы стоимостей в (6.1) можно не искать минимум по всем вершинам (минимум по k). Достаточно на каждом шаге использовать фиксированную вершину, меняя фиксированную вершину от шага к шагу. Точнее, на k-м шаге мы вычисляем

$$c_{ij}^{(k)} = \min\left\{c_{ij}^{(k-1)},\ c_{ik}^{(k-1)} + c_{kj}^{(k-1)}\right\}, \tag{6.2}$$

где $C^{(0)} = E$. Через n шагов мы получим матрицу $C^{(n)}$, которая и будет являться матрицей стоимостей. Изложенный алгоритм был предложен Р. Флойдом.

Покажем, что алгоритм Флойда действительно приводит к минимальным весам маршрутов между любой парой вершин. Отметим, что на k-м шаге матрица C^k будет содержать для каждой пары вершин минимальные веса, получаемые по маршрутам, проходящим по первым k вершинам графа. Доказательство можно строить методом индукции. На старте в качестве $C^{(0)}$ выбираем матрицу E, имея в виду нуль и единицу соответствующего полукольца, т.е. в данном случае $c_{ij}^{(0()} = 0$ при $i = j$ и $c_{ij}^{(0()} = \infty$ при $i \neq j$. Очевидно, что эта матрица дает веса оптимальных маршрутов длины 0. Предположим, что матрица $C^{(k-1)}$ содержит веса оптимальных маршрутов, пролегающих по вершинам с номерами до $k-1$ включительно. Тогда формула (6.2) для вершин с номерами i и j выбирает наилучший маршрут среди „старых“ веса $c_{ij}^{(k-1)}$ и новых с дополнительной вершиной с номером k. В самом деле, наилучший маршрут, проходящий через k-ю вершину может состоять из двух оптимальных маршрутов, первый из i-й вершины в k-ю, второй из k-й вершины в j-ю. Через n шагов получим для каждой пары вершин вес оптимального маршрута без ограничений на включаемые вершины.

6.5. Циклы, разрезы и задача Эйлера

Эйлеровы цепи и циклы. Циклы в графе. Фундаментальные циклы, числа Бетти. Разрезы. Критерий существования. Приложения.

Эйлеровы графы. Обнаружение замкнутых цепей, и в частности циклов — важная задача в исследовании графов. Во-первых, циклы в графе — существенная характеристика структуры этого графа. Во-вторых, наличие циклов влияет на многие алгоритмы на графах. Например, циклы отрицательного веса могут привести к тому, что задача о кратчайших путях не будет иметь решения.

Одной из задач теории графов, фактически положившей начало этой теории, является задача обнаружения ***эйлеровых циклов*** и ***эйлеровых цепей***. Эйлеров цикл — это закнутая цепь, которая проходит через каждое ребро графа и притом один раз. Эйлеров цикл может не быть циклом в нашем понимании, так как некоторые вершины могут проходится несколько раз. В этом случае мы введем в обиход термин ***квазицикл***, т.е. любая замкнутая цепь, в которой ребра не повторяются. Отметим, что вершины, не попадающие в эйлеров цикл, являются изолированными. Мы этот случай рассматривать не будем и дополним определение эйлерова цикла требованием, что каждая вершина попадает в эйлеров цикл.

Понятие эйлерова цикла имеет прямое отношение к задачам прорисовки фигур без отрыва карандаша от бумаги. Другой исторической задачей является ***задача о Кенигсбергских мостах***, изучавшаяся Л. Эйлером и приведшая к понятию эйлерова цикла. Задача состояла в обходе Кенигсберга с условием, что каждый мост проходится ровно один раз (рис. 6.2). Рассматривая разделенные участки суши как вершины графа, а соединяющие эти участки мосты — как ребра (рис. 6.3), приходим к задаче поиска такой цепи в графе, которая содержит каждое ребро, и притом один раз, т.е. к задаче поиска эйлеровой цепи.

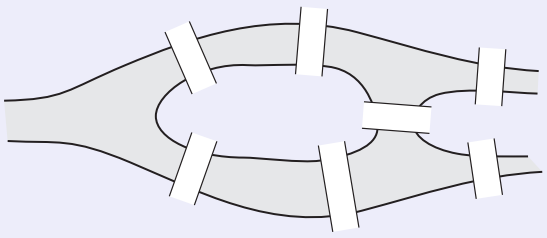


Рис. 6.2

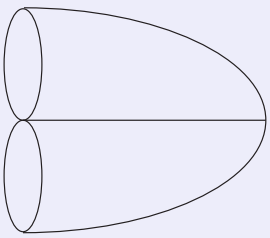


Рис. 6.3

Любой квазицикл можно разделить на несколько циклов, возможно, имеющих общие вершины, но не имеющих общих ребер. Действительно, начав перемещение по квазициклу мы получим последовательность вершин $a_1, a_2, \ldots, a_k, a_i$, где $i < k$. Поскольку в цепи ребра не

повторяются, последовательность вершин a_i, a_{i+1}, ..., a_k есть цикл. Выделим этот цикл, а маршрут движения сократим: a_1, a_2, ..., a_{i-1}, a_i, a_k. Повторяя эту процедуру и учитывая, что количество вершин в цикле конечно, мы придем к набору циклов с общими вершинами, но без общих ребер.

Теорема 6.4. Граф G имеет эйлеров цикл тогда и только тогда, когда он связен и любая его вершина имеет четную степень.

◄ Отметим, что оба отмеченных условия (связность и четность степеней) необходимы для существования эйлерова цикла, поскольку при движении по циклу мы попадаем в каждую вершину и выходим. В результате, во-первых, цикл соединяет все вершины, что обеспечивает связность графа. А во-вторых, для каждой вершины инцидентные ребра разделяются на пары, в каждой из которых одно ребро входящее, а другое выходящее. Поэтому степени всех вершин четные.

Предположим, что граф связен, а все вершины имеют четную степень. Выберем произвольную вершину v_0 и начнем движение по графу по любому инцидентному ребру. Это движение не может закончиться в вершине, отличной от стартовой, поскольку четная степень вершины гарантирует, что если мы пришли в нее, то есть хотя бы одно не использовавшееся ребро. Поэтому в такой вершине цепь можно продолжить, обеспечивая неповторяемость ребер. В то же время в силу конечности множества всех ребер цепь в какой-то момент должна закончиться. Это может произойти только в тот момент, когда мы вернемся в стартовую вершину и не окажется ни одного инцидентного ребра, которое еще не использовалось.

Обозначим полученный квазицикл γ_1. Предположим, что в этот квазицикл входят не все ребра графа. Тогда должно быть хотя бы одно ребро, не вошедшее в квазицикл γ_1, инцидентное одной из вершин v_1 этого квазицикла (в противном случае мы имеет несвязный граф). Отметим, что после удаления квазицикла γ_1 все вершины графа будут иметь четную степень. Начинаем движение из вершины v_1 по ребрам, не вошедшим в γ_1. Рассуждая как и выше, заключаем, что движение закончится в вершине v_1, и мы получаем квазицикл γ_2, не имеющий с γ_1 общих ребер, но имеющий с ним общую вершину v_1. Эти два квазицикла можно объединить в один квазицикл γ_2' (рис. 6.4). Если объединенный квазицикл охватывает не все ребра графа, мы опять можем выбрать вершину v_3 этого квазицикла, у которой есть незадействованные инцидентные ребра, и, начиная движение из вершины v_3, построить следующий квазицикл γ_3. Присоединяем квазицикл γ_3 к γ_2'. Этот процесс продолжаем далее. Он завершится тогда, когда не останется незадействованных ребер. В результате будет получен квазицикл, содержащий все ребра графа, т.е. эйлеров цикл. ►

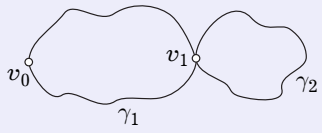


Рис. 6.4

Следствие 6.2. Для того чтобы граф G имел эйлерову цепь, необходимо и достаточно, чтобы он был связен, две его вершины имели нечетную степень, а остальные — четную.

◄ Пусть вершины v_1 и v_2 имеют нечетную степень. Если эти вершины несмежные, добавим к графу ребро γ, соединяющее v_1 и v_2. Получим связный граф, у которого все вершины имеют четную степень. Такой граф имеет эйлеров цикл. Удалив из эйлерова цикла ребро γ, получим эйлерову цепь исходного графа.

Если вершины v_1 и v_2 смежные, мы можем добавить к графу еще одну вершину v', соединив ее с вершинами v_1 и v_2. Опять в расширенном графе существует эйлеров цикл, который после удаления дополнительной вершины и двух ребер становится эйлеровой цепью исходного графа. ►

Фундаментальные циклы. Отметим две характеристики графа, условно противоположные друг другу: несвязность и существование циклов. Для связного графа количество ребер, при которых могут отсутствовать циклы, ограничено ($n-1$ ребро). При увеличении количества ребер появляются циклы, при уменьшении — теряется связность.

Пусть граф G связный. Выберем в нем остовное дерево T. Добавление к T одного ребра графа G приводит к появлению одного цикла. В самом деле, если добавляемое ребро γ входит в два цикла, то объединяя циклы, но уже без γ, получим цикл в дереве T (рис. 6.5). Поскольку дерево не имеет циклов, то и предположение о появлении в связи с γ двух циклов оказывается ложным.

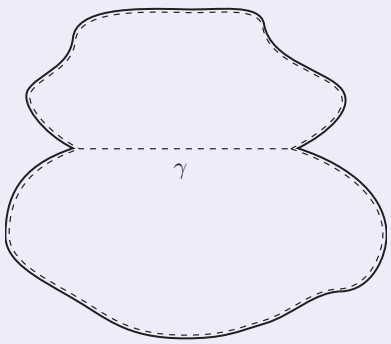


Рис. 6.5

Итак, выбор в графе G остовного дерева разделяет все ребра графа на две группы: древесные ребра, вошедшие в остов, и обратные ребра, не вошедшие в остов. С каждым обратным ребром графа связан вполне определенный цикл графа G. Совокупность этих циклов образует ***фундаментальную систему циклов***. Основным ее свойством является то, что с помощью фундаментальной системы циклов можно построить любой цикл графа, в то время как ни один из циклов фундаментальной системы не выражается через другие. Это похоже на ситуацию в линейном пространстве при рассмотрении понятия базиса. Отметим, что ассоциация с линейной алгеброй оказывается весьма содержательной и не является поверхностной.

Любую цепь в графе без повторений ребер (полупростую цепь) можно интерпретировать как некоторой множество ребер этого графа. Если из данной совокупности ребер графа можно сложить полупростую цепь, то такая цепь единственна (если, конечно, не различать цепи из одних и тех же ребер, но с противоположной ориентацией). Другими словами, множество всех цепей графа можно рассматривать как подмножество булеана множества всех ребер графа G. Булеан множестува ребер с операцией симметрической разности представляет собой абелеву группу, причем в этой группе каждый элемент является противоположным самому себе. Такая абелева группа может рассматриваться как линейное пространство над полем $\mathbb{Z}_2$. В результате мы приходим к понятию ***пространства цепей графа***. В этом линейном пространстве рассмотрим линейную оболочку всех циклов. Получим ***пространство циклов графа***. Наша цель — показать, что фундаментальная система циклов есть базис в пространстве циклов графа.

Отметим, что пространство циклов содержит все квазициклы, так как каждый квазицикл есть сумма нескольких циклов.

Теорема 6.5. Сумма двух квазициклов есть либо пустое множество, либо квазицикл, либо объединение нескольких попарно непересекающихся квазициклов.

◀ Рассмотрим общие вершины двух циклов. В сумме степеней каждой вершины, относящихся к каждому из квазициклов, общие ребра считаются дважды. Поэтому после удаления общих ребер, т.е в результате суммирования, мы из двух квазициклов получим граф, у которого все вершины имеют четные степени. Это означает, что каждая компонента связности такого графа есть эйлеров граф, или квазицикл. Значит, сумма двух квазициклов есть объединение не пересекающихся квазициклов. ▶

Теорема 6.6. Фундаментальная система циклов, порожденная выбранным остовом, является базисом в пространстве циклов связного графа.

◀ Фундаментальная система циклов линейно независима в пространстве циклов. В самом деле, выберем и приравняем нулю произвольную линейную комбинацию $\alpha_1 c_1 + \alpha_2 c_2 + \ldots + \alpha_k c_k$ циклов c_1, c_2, …, c_k с коэффициентами $\alpha_i \in \mathbb{Z}_2$. Цикл c_1 имеет обратное ребро r_1, которое не входит ни в какой другой цикл. Значит, равенство нулю линейной комбинации возможно только при $\alpha_1 = 0$. Повторяя рассуждения для всех остальных циклов линейной комбинации, заключаем, что все коэффициенты линейной комбинации равны нулю.

Пусть c_1, c_2, …, c_k — фундаментальная система циклов, причем каждый цикл c_i связан с обратным ребром γ_i. Выберем произвольный цикл c. Пусть этот цикл содержит обратные ребра r_{i_1}, r_{i_2}, …, r_{i_l}. Тогда множество $c \triangle c_{i_1} \triangle c_{i_2} \triangle \ldots \triangle c_{i_l}$ не содержит обратных ребер, а следовательно, состоит из ребер, принадлежащих остову. Это множество как сумма нескольких циклов, либо пусто, либо является объединением некоторого набора не пересекающихся квазициклов, а значит, набора циклов, не имеющих общих ребер. Однако в остовном дереве вообще нет циклов. Поэтому указанная сумма есть пустое множество, т.е.

$$c \triangle c_{i_1} \triangle c_{i_2} \triangle \ldots \triangle c_{i_l} = \varnothing.$$

Это равенство эквивалентно равенству

$$c = c_{i_1} \triangle c_{i_2} \triangle \ldots \triangle c_{i_l},$$

поскольку в пространстве циклов каждый элемент является противоположным самому себе.

Таким образом, любой цикл графа является линейной комбинацией фундаментальной системы циклов. Следовательно, фундаментальная система циклов — базис в пространстве циклов графа. ▶

Доказанная теорема позволяет определить размерность пространства циклов по числу обратных ребер. Если n — число вершин связного графа, m — число его ребер, то размерность пространства циклов равна $m - n + 1$.

Понятие пространства циклов можно ввести и для несвязного графа. В этом случае любой цикл (квазицикл) есть цикл (квазицикл) одной из компонент. Остов графа будет лесом, но при этом сохраняется свойство, что любое обратное ребро порождает ровно один цикл. Таким образом, пространство циклов имеет размерность $m-n+p$, где p — число компонент связности.

Размерность пространства циклов, равная $c(G) = m - n + p$, называется ***цикломатическим числом графа***. Число $\nu(G) = n - p$, равное числу ребер в остове графа, называется ***коцикломатическим числом***. Обе эти характеристики — инварианты, которые могут использоваться при анализе графов на изоморфность.

Отметим, что не любой базис пространства циклов можно получить исходя из некоторого остова этого графа. Фундаментальная система циклов обладает следующим свойством: каждый цикл фундаментальной системы имеет ребро, при удалении которого разрывается только один цикл. Для графа на рис. 6.6 цикломатическое число равно 4. Нетрудно увидеть четыре независимых цикла этого графа: $1-2-3$, $2-4-5$, $2-3-5$, $3-5-6$. Сумма двух, трех или всех четырех циклов не может быть равна нулю, поскольку у трех циклов из четырех есть „персональные“ ребра. В то же время у внутреннего цикла ребра таковы, что удаление любого из них разрывает сразу два цикла из четырех. Это означает, что рассмотренная система циклов не порождается каким-либо остовом.

Для построения фундаментальной системы циклов следует выполнить следующие шаги.

1. построить остов графа (дерево или лес);
2. разделить все ребра графа на древесные и обратные;
3. для каждого обратного ребра построить соответствующий ему цикл. Для этого в остове необходимо найти единственную цепь, соединяющую концы ребра, и к этой цепи добавить само ребро.

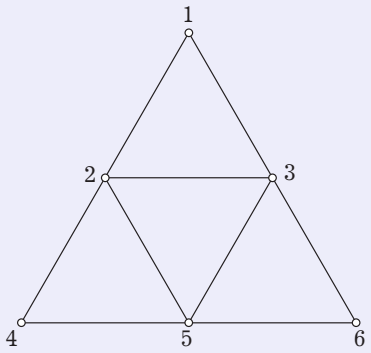


Рис. 6.6

Фундаментальные разрезы. Разделим вершины графа произвольным образом на две группы и выберем в графе множество тех ребер, которые соединяют вершины разных групп. Это множество может быть пустым (это произойдет в том случае, если каждая компонента связности целиком попадает в одну из групп). Если же оно не пустое, то удаление ребер множества из графа увеличивает его связность. Описанное множество называется ***разрезом графа***.

Изучение разрезов позволяет оценить важные свойства графа. Например, анализ разрезов графа городских дорог позволяет найти узкие места в организации транспортных потоков и найти выверенные решения по развитию сети дорог.

Вообще, под разрезом можно было бы понимать любое множество ребер графа, удаление которых повышает связность этого графа. Однако такое определение не эквивалентно предыдущему, оно является расширением понятия разреза. В самом деле, в графе на рис. 6.6 выберем множество из трех ребер $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$. Удаление этих ребер делает граф несвязным (вершина 1 становится изолированной). Однако не существует такого разделения вершин на две группы, при котором концы все трех ребер будут принадлежать разным группам. Множество ребер, удаление которых повышает связность графа, будем называть ***разделяющим множеством***. Разрез — частный случай разделяющего множества.

Множество всех разрезов графа можно рассматривать как полугруппу с операцией объединения (точнее, совокупность всех разрезов в реберном булеане графа с операцией объединения порождает некоторую подполугруппу). Поскольку объединение — идемпотентная операция, ***алгебра разрезов*** есть полурешетка, отношение порядка которой совпадает с отношением включения множеств. Интересна база этой алгебры, т.е. минимальное множество разрезов, порождающее всю алгебру разрезов. Естественно к базе разрезов относить только минимальные разрезы, т.е. такие разрезы, которые не содержат подмножеств, также являющихся разрезами (такие разрезы также называют ***простыми разрезами***).

Условие минимальности позволяет стереть границу между разрезом и разделяющим множеством: минимальное разделяющее множество оказывается разрезом. Отметим, что минимальное разделяющее множество состоит из ребер лишь одной компоненты связности графа. Увеличение связности означает, что хотя бы одна из компонент связности графа разделяется на большее число компонент. Очевидно, что можно ограничиться множеством ребер, входящих только в эту разделяемую компоненту. В свете этого можно ограничиться рассмотрением частного случая связного графа.

Теорема 6.7. Любое минимальное разделяющее множество связного графа разделяет граф точно на две компоненты и тем самым определяется некоторым разделением вершин графа на два множества (т.е. является разрезом).

◀ Предположим, что граф разделился на три или большее число компонент. Поскольку изначально граф был связным, в разрез входит ребро, соединяющее первую компоненту (при любой изначально выбранной нумерации компонент) с какой-то другой. Удаление этого ребра из разделяющего множества приводит к слиянию двух компонент, но оставляет граф несвязным.

Таким образом, мы приходим к другому разделяющему множеству, которое является подмножеством рассматриваемого разделяющего множества. Это противоречит условию минимальности разделяющего множества.

Минимальное разделяющее множество делит граф в точности на две компоненты G_1 и G_2, разделяя тем самым множество вершин графа на две группы V_1 и V_2. Очевидно, что рассматриваемое разделяющее множество содержит все ребра, соединяющие вершины из V_1 с вершинами из V_2, а в силу минимальности не содержит никаких других ребер. Значит, рассматриваемое разделяющее множество порождается группами вершин V_1 и V_2 и является разрезом. ►

Теорема 6.8. Любой разрез графа можно представить как объединение минимальных разрезов.

◄ Чтобы доказать сформулированное утверждение, нужно показать, что любое ребро рассматриваемого разреза входит в некоторый минимальный разрез, являющийся частью рассматриваемого разреза. Из этого факта сразу вытекает, что любой разрез есть объединение минимальных разрезов, подчиненных данному.

Как и выше, можем ограничиться случаем связного графа $G = (V,\ E)$. Рассмотрим некоторое ребро γ разреза R, соединяющее вершины v_1 и v_2. Рассмотрим вспомогательный граф, вершинами которого являются компоненты связности графа $(V,\ E \setminus R)$ (графа G без разреза R), полагая, что две такие компоненты соединены ребром, если в разрезе есть ребро, соединяющее вершины двух компонент связности. Этот вспомогательный граф называют ***конденсацией графа*** G. Так как любые две вершины одной компоненты связности достижимы друг из друга, то связь между двумя компонентами связности можно провести по любому ребру, соединяющему вершины этих компонент. Возникает отношение эквивалентности на ребрах графа G, причем каждому классу эквивалентности будет соответствовать одно ребро конденсированного графа.

В силу построения разрез состоит из некоторого множества упомянутых выше классов эквивалентных ребер. Удаление разреза приводит к тому, что конденсированный граф становится графом без ребер. Отметим, что любую цепь в конденсированном графе, соединяющую компоненты с вершинами v_1, v_2 можно детализировать так, что получится цепь, соединяющая в исходном графе вершины v_1, v_2.

Из этих рассуждений вытекает, что минимальный разрез графа G, подчиненный разрезу R представляет собой объединение нескольких классов эквивалентности, а потому соответствует некоторому разрезу конденсированного графа.

Поскольку разрез порождается двумя множествами вершин V_1 и V_2, конденсированный граф будет состоять из двух групп вершин (компонент связности, попадающих в V_1 или V_2), причем ребра графа будут соединять вершины разных групп (такой граф называется ***двудольным***). Пусть Γ — класс эквивалентности, содержащий ребро γ. Покажем, что Γ является частью некоторого минимального разреза конденсированного графа. Этот разрез порождает минимальный разрез исходного графа, подчиненный разрезу R. Пусть $\tilde{v}_1$ и $\tilde{v}_2$ — вершины конденсированного графа, соответствующие вершинам v_1 и v_2 исходного.

Выберем совокупность тех ребер конденсированного графа, инцидентных $\tilde{v}_1$, которые являются началом цепи из вершины $\tilde{v}_1$ в вершину $\tilde{v}_2$. Эта совокупность является разделяющим множеством конденсированного графа, поскольку его вершины $\tilde{v}_1$ и $\tilde{v}_2$ после удаления множества ребер оказываются недостижимыми друг из друга. В то же время эта совокупность — минимальное разделяющее множество, поскольку для любых двух вершин есть цепь, соединяющая их. Удаляемое множество ребер либо не затрагивает указанную цепь, либо эта цепь проходит через $tilde v_1$. Но тогда либо восстанавливаемое ребро восстанавливает и всю цепь, либо восстановленное ребро позволяет по этому ребру попасть в вершину $\tilde{v}_2$, а затем вернувшись к вершине, соседней с $\tilde{v}_1$, затем построить альтернативную цепь. Значит, сокращение количества ребер в разделяющем множестве восстанавливает связность.

Поскольку минимальное разделяющее множество есть простой разрез, любое ребро разреза R будет входить в некоторый простой разрез, целиком входящий в R. ►

Систему минимальных разрезов, порождающих алгебру разрезов, можно строить, используя остов графа. Поскольку любые вершины связного графа можно соединить цепью в пределах остова графа, любой разрез графа должен быть и разрезом остова. Разрезы дерева устроены просто: минимальным разрезом дерева является любое ребро, поскольку удаление ребра в дереве автоматически делает это дерево несвязным.

Из этого рассуждения вытекает, что любой разрез графа должен содержать хотя бы одно древесное ребро. Можно построить систему разрезов, каждый из которых содержит ровно одно древесное ребро. Достаточно к этому древесному ребру добавить обратные ребра всех содержащих его фундаментальных циклов. Любая простая цепь, соединяющая концы древесного ребра есть часть цикла, содержащего это ребро. Но любой цикл есть сумма фундаментальных циклов, а потому содержит одно из выбранных обратных ребер. Следовательно, любая простая цепь, соединяющая концы древесного ребра, окажется разорванной, а граф станет несвязным. Ясно, что указанное разделяющее множество минимально, так как удаление из него любого ребра восстанавливает одну из цепей, связывающей две вершины. Граф возвращает свою связность.

Построенная система разрезов называется ***фундаментальной системой разрезов***. В фундаментальной системе разрезов каждый элемент — минимальный разрез. Однако отсюда не следует, что любой разрез графа может быть получен как объединение некоторых разрезов фундаментальной системы. Например, треугольник с вершинами 1, 2, 3 и ребрами $\gamma_1 = \{1, 2\}$, $\gamma_2 = \{1, 3\}$, $\gamma_3 = \{2, 3\}$ (например, подграф графа на рис. 6.6, порожденный вершинами 1, 2, 3) имеет остов с двумя ребрами γ_1 и γ_2. Фундаментальную систему разрезов образуют пары ребер $\{\gamma_1, \gamma_3\}$ и $\{\gamma_2, \gamma_3\}$. Разрез $\{\gamma_2, \gamma_3\}$ также минимальный, но не является объединением предыдущих двух.

ОГЛАВЛЕНИЕ