Д. М. Васильков

Дискретная математика и математическая логика

Курс лекций

для студентов, обучающихся по специальности
«Прикладная информатика»

Минск 2017

Оглавление

Глава 1

Введение в математическую логику

Логика – наука древняя. Интерес к ней среди математиков и философов возник еще в Греции IV-VI веков до н.э. Первые серьезные сочинения принадлежат Аристотелю, который максимально приблизил логику к уровню математики.

Математическая логика занимается формальными законами построения рассуждений и доказательств. Одной из причин, подтолкнувших развитие этого направления, стало появление ряда проблем, выявленных в теории множеств на рубеже XIX-XX веков. Открытие так называемых парадоксов Рассела и Кантора свидетельствовало о наличии противоречий внутри целой теории (в ее наивном изложении) и поэтому потребовало пересмотра ее основ. Дальнейшие работы в этом направлении привели к еще более неожиданным результатам: немецкий математик Курт Гёдель в 1930-х годах строго доказал, что всякая достаточно «богатая» теория обязательно содержит утверждения, которые нельзя ни доказать ни опровергнуть в рамках этой теории.

1.1. Высказывания

Понятие высказывания является первичным, то есть неопределяемым. Под *высказыванием* понимается повествовательное предложение, относительно которого можно сказать, что оно либо истинно, либо ложно, но ни то и другое одновременно.

Примеры высказываний: $2 + 3 = 5$, $\sqrt{4} = 3.89$, «Сегодня 17 января 1087 года по Юлианскому календарю». Запись $2 + 3$ или фраза «17 января 1087 года по Юлианскому календарю» высказываниями не являются.

Среди всех высказываний выделяют *простые* высказывания, или

атомы: это высказывания, истинность или ложность которых не вызывает сомнения. Будем обозначать такие высказывания строчными буквами (a, α, а и т.д.).

Простые высказывания могут принимать одно из двух значений – истина или ложь, поэтому пишут: **И** или **Л**, **T** (True) или **F** (False), **1** или **0**.

Кроме простых высказываний, структура которых не анализируется, рассматриваются их комбинации, связанные логическими знаками, заменяющими слова обычного языка. В отличие от первичного понятия высказывания результаты применения логических связок строго определяются. Ниже перечислены основные логические связки, их обозначение и трактовка:

- $\neg$ НЕ (*отрицание*): высказывание a истинно тогда и только тогда, когда высказывание $\neg a$ ложно (и наоборот).

- $\wedge$ И (*конъюнкция*): высказывание $a \wedge b$ истинно тогда и только тогда, когда истинны одновременно высказывания a и b.

- $\vee$ ИЛИ (*дизъюнкция*): высказывание $a \vee b$ истинно тогда и только тогда, когда истинно высказывание a или высказывание b или оба одновременно.

- $\Rightarrow$ СЛЕДУЕТ (*импликация*). По определению, высказывание $a \Rightarrow b$ ложно только в том случае, когда a истинно, а b ложно (во всех остальных случаях высказывание $a \Rightarrow b$ истинно). Другими словами, из истинного утверждения не может следовать ложное. Такая трактовка импликации также означает, что из ложного высказывания может следовать все что угодно. Запись $a \Rightarrow b$ читается следующим образом:

  - **Если** истинно утверждение a, **то** истинно утверждение b.
  - Утверждение b истинно **тогда, когда** истинно утверждение a.
  - Утверждение a истинно **только тогда, когда** истинно утверждение b.
  - Истинность b есть **необходимое** условие истинности a.
  - Истинность a есть **достаточное** условие истинности b.

- $\Leftrightarrow$ РАВНОСИЛЬНО (*эквивалентность*): запись $a \Leftrightarrow b$ означает, что a и b истинны или ложны одновременно. Читается она следующим образом:

– Утверждение a истинно **тогда и только тогда, когда** истинно утверждение b.

– Выполнение a есть **необходимое и достаточное** условие выполнения b.

Результат логической связки двух высказываний удобно записывать в виде *таблицы истинности*:

a	$\neg b$
0	1
1	0

a	b	$a \wedge b$	$a \vee b$	$a \Rightarrow b$	$a \Leftrightarrow b$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Немецкий логик Э. Шрёдер приводит следующий пример построения и анализа сложного утверждения, представленного в виде логической связки простых высказываний: «Один химик высказал предположение, что соли, которые не окрашены, есть соли, не являющиеся органическими соединениями, или органические соединения, которые не окрашены. Верно это или нет?»

Обозначим высказывания буквами:

- с: «Это вещество – соль».
- к: «Это вещество окрашено».
- о: «Это вещество – органическое соединение».

Тогда записанное на языке логики высказываний, утверждение химика выглядит следующим образом:

$$(\text{с} \wedge \neg\text{к}) \Rightarrow (\text{с} \wedge \neg\text{о}) \vee (\text{о} \wedge \neg\text{к}).$$

Построим таблицу истинности для каждого высказывания в отдельности и для всего утверждения в целом:

с	к	о	с $\wedge \neg$к	с $\wedge \neg$о	о $\wedge \neg$к	$(\text{с} \wedge \neg\text{о}) \vee (\text{о} \wedge \neg\text{к})$	$\Rightarrow$
0	0	0	0	0	0	0	1
0	0	1	0	0	1	1	1
0	1	0	0	0	0	0	1
0	1	1	0	0	0	0	1
1	0	0	1	1	0	1	1
1	0	1	1	0	1	1	1
1	1	0	0	1	0	1	1
1	1	1	0	0	0	0	1

Таким образом, с помощью таблицы истинности мы установили, что с точки зрения формальной логики при любой комбинации истинности или ложности высказываний относительно данного вещества, конечное утверждение истинно.

1.2. Формулы

Будем обозначать простые высказывания *переменными* – символами из произвольного алфавита[1]. Для записи истинности высказываний будем использовать символы 0 (ложь) и 1 (истина), которые называются *логическими константами*. С помощью простых высказываний, логических констант и логических связок могут быть получены более сложные, составные высказывания, подобные высказыванию из предыдущего примера.

Правильно построенные высказывания называются *формулами* (пример неправильно построенного высказывания: $x \Rightarrow \vee y$). Дадим индуктивное определение формулы.

Определение 1.1.

1. Логические константы 0 и 1 есть формулы.
2. Переменная есть формула.
3. Если A и B – формулы, то $\neg A$, $A \wedge B$, $A \vee B$, $A \Rightarrow B$ и $A \Leftrightarrow B$ – также формулы.
4. Определение закончено.

Заметим, что при записи высказываний могут возникнуть неоднозначности в их трактовке, например: $a \Rightarrow b \vee c \Rightarrow d$. Для определения порядка применения логических связок используются скобки: $((a \Rightarrow b) \vee c) \Rightarrow d$.

Пример. Пусть a, b, s_1, s_2 – переменные. Тогда $(\neg 0)$, $((\neg a) \wedge (\neg b))$, $((a \vee b) \Rightarrow ((\neg s_1) \wedge s_2)$ – формулы.

[1]Ранее для простых высказываний мы договорились использовать только строчные символы, хотя эта договоренность чисто условная.

Чтобы уменьшить количество скобок, затрудняющих чтение формулы, вводится следующий порядок применения операций:

$$\neg,\ \wedge,\ \vee,\ \Rightarrow,\ \Leftrightarrow .$$

С учетом этого порядка формулу из предыдущего примера можно переписать без использования скобок:

$$a \vee b \Rightarrow \neg s_1 \wedge s_2.$$

Существует более краткое (и интуитивно более понятное) определение формулы:

Определение 1.2.

1. Любая переменная есть формула.
2. Если A – формула, то $(\neg A)$ также формула.
3. Если A и B – формулы, то $(A \Rightarrow B)$ также формулы.
4. Никаких других формул в математической логике нет.

Докажем эквивалентность этих двух определений. Покажем для этого, что с помощью одних только логических связок $\neg$ и $\Rightarrow$ можно построить любую формулу, содержащую логические связки $\wedge$, $\vee$ и $\Leftrightarrow$, а также логические константы 0 и 1. Действительно:

1. Для записи константы 1 можно использовать формулу $(x \Rightarrow x)$, а для записи константы 0 – формулу $(\neg 1)$. То есть, согласно определению 1.2, логические константы – это формулы.
2. Для записи высказывания $(A \vee B)$ можно использовать формулу $(\neg A \Rightarrow B)$.
3. Для записи высказывания $(A \wedge B)$ можно использовать формулу $\neg(\neg A \vee \neg B)$.
4. Для записи высказывания $(A \Leftrightarrow B)$ можно использовать формулу $(A \Rightarrow B) \wedge (B \Rightarrow A)$.

Задание. С помощью таблиц истинности проверить использованные выше эквивалентности.

1.3. Интерпретации, тавтологии и противоречия

Заменив в формуле простые высказывания буквами, мы абстрагируемся от конкретного смысла утверждения, заложенного в этой формуле. Другими словами, в виде формулы мы имеем логическую связку неких фактов, суть которых нас не интересует. По словам Аристотеля, основной задачей логики является установление истинности или ложности утверждения при условии истинности приведенных фактов, исходя лишь из формальной структуры рассуждения, а не из смысла этого утверждения и самих фактов.

Определение 1.3. Пусть $A(x_1, \ldots, x_n)$ – формула, где $x_1, \ldots, x_n$ – входящие в нее переменные. Конкретный набор I значений переменных называется *интерпретацией* формулы A.

Формула может быть истинной при одной интерпретации и ложной при другой.

Определение 1.4. Если формула истинна при некоторой интерпретации, то она называется *выполнимой*. Если она истинна при всех возможных интерпретациях, то она называется *тавтологией* (или *общезначимой*). Формула, ложная при всех интерпретациях, называется *противоречием* (или *невыполнимой*).

Пример. $A \vee \neg A$ – тавтология (закон исключенного третьего), «одно из двух: он присутствовал или отсутствовал».

Из определения тавтологии и противоречия вытекают следующие простые утверждения:

Предложение 1.1. *Если A – тавтология, то $\neg A$ – противоречие и наоборот.*

Предложение 1.2. *Если формулы A и $A \Rightarrow B$ – тавтологии, то формула B – тавтология.*

Доказательство. Действительно, пусть существует интерпретация I, при которой $B(I) = 0$. Но $A(I) = 1$. Значит для интерпретации I формула $A \Rightarrow B$ является ложной, то есть не является тавтологией, что противоречит условию. ∎

1.4. Логическое следствие и логическая эквивалентность

Определение 1.5. Формула B называется *логическим следствием* формулы A, обозначается $A \vdash B$, если для любой интерпретации, при которой формула A истинна, формула B тоже истинна. В этом случае говорят, что формула (высказывание) B *логически следует* из A.

Определение 1.6. Говорят, что формулы A и B логически *эквивалентны*, пишут $A = B$, если они являются логическим следствием друг друга. Логически эквивалентные формулы имеют одинаковые значения при любой интерпретации.

Ниже перечислены основные эквивалентности логики высказываний.

Идемпотентность:	$A \wedge A = A$ $A \vee A = A$
Коммутативность:	$A \wedge B = B \wedge A$ $A \vee B = B \vee A$
Ассоциативность:	$A \wedge (B \wedge C) = (A \wedge B) \wedge C$ $A \vee (B \vee C) = (A \vee B) \vee C$
Дистрибутивность:	$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$
Законы *де Моргана*:	$\neg(A \vee B) = \neg A \wedge \neg B$ $\neg(A \wedge B) = \neg A \vee \neg B$
Инволютивность:	$\neg(\neg A) = A$
Свойства *нуля* и *единицы*:	$A \vee \neg A = 1,\ A \wedge \neg A = 0$ $A \vee 0 = A,\ A \wedge 0 = 0$ $A \vee 1 = 1,\ A \wedge 1 = A$
Правила *поглощения*:	$(A \vee B) \wedge A = A,\ (A \wedge B) \vee A = A$
Контрапозиция:	$A \Rightarrow B = \neg B \Rightarrow \neg A$
Другие равенства:	$A \Rightarrow B = \neg A \vee B$ $A \Leftrightarrow B = (A \wedge B) \vee (\neg A \wedge \neg B)$

Определение 1.7. Формула B называется *логическим следствием* формул $A_1, \ldots, A_n$, обозначается $A_1, \ldots, A_n \vdash B$, если для любой интерпретации, для которой каждая из формул $A_1, \ldots, A_n$ истинна, формула B тоже истинна.

Заметим, что для для данной интерпретации I каждая из формул $A_1, \ldots, A_n$ истинна тогда и только тогда, когда для этой интерпретации истинна формула $A_1 \wedge \ldots \wedge A_n$.

Пример.

$$x,\ (x \Rightarrow y) \vdash y, \quad (x \Rightarrow y),\ (y \Rightarrow z) \vdash (x \Rightarrow z)$$

Иногда для обозначения логического следствия используют запись:

$$\begin{array}{l} x \\ x \Rightarrow y \\ \hline \therefore y \end{array} \qquad \begin{array}{l} x \Rightarrow y \\ y \Rightarrow z \\ \hline \therefore x \Rightarrow z \end{array}$$

Рассмотрим следующие суждения:

«Если бы он не сказал ей, она бы не узнала.»
«А не спроси она его, он бы ей и не сказал.»
«Но она узнала.»
«Следовательно, она спросила.»

Обозначим простые высказывания символами: Ск – «Он сказал.», Сп – «Она спросила.», У – «Она узнала.». Получаем последовательность из трех формул и их логическое следствие:

$$\neg\,\text{Ск} \Rightarrow \neg\,\text{У},\ \neg\,\text{Сп} \Rightarrow \neg\,\text{Ск},\ \text{У} \vdash\ \text{Сп}$$

или в альтернативной записи

$$\begin{array}{l} \neg\,\text{Ск} \Rightarrow \neg\,\text{У} \\ \neg\,\text{Сп} \Rightarrow \neg\,\text{Ск} \\ \quad \text{У} \\ \hline \therefore \text{Сп} \end{array}$$

Важно отметить, что логическое следствие является *новым знанием*, новой информацией, полученной из известных фактов.

Теорема 1.1 (Основная теорема логического вывода).
Формула B является логическим следствием формул $A_1, \ldots, A_n$ тогда и только тогда, когда формула $A_1 \wedge \ldots \wedge A_n \Rightarrow B$ – является тавтологией.

Доказательство. $\Rightarrow$ («только тогда...», необходимость). Пусть B – логическое следствие, докажем, что $A_1 \wedge \ldots \wedge A_n \Rightarrow B$ – тавтология. Пусть I – произвольная интерпретация. Если все $A_i(I)$ истинны, то,

по условию, B также истинна, и стало быть, истинна импликация $A_1 \wedge \ldots \wedge A_n \Rightarrow B$. Если же среди $A_i(I)$ имеется хотя бы одно ложное утверждение, то эта импликация также истинна (по определению). Следовательно, эта формула – тавтология.

$\Leftarrow$ («тогда...», достаточность). (Формула – тавтология, докажем, что B – логическое следствие.) Пусть I – интерпретация, для которой все A_i истинны. Значит левая часть формулы $A_1 \wedge \ldots \wedge A_n \Rightarrow B$ истинна и, стало быть, B тоже истинна, поскольку истинна импликация. ∎

Следствие 1.1. *Формула B является логическим следствием формул $A_1, \ldots, A_n$ тогда и только тогда, когда $A_1 \wedge \ldots \wedge A_n \wedge \neg B$ – противоречие.*

Доказательство. Формула $A_1 \wedge \ldots \wedge A_n \Rightarrow B$ является тавтологией тогда и только тогда, когда $\neg (A_1 \wedge \ldots \wedge A_n \Rightarrow B)$ – противоречие. Но

$$\begin{aligned} \neg (A_1 \wedge \ldots \wedge A_n \Rightarrow B) &= \neg (\neg (A_1 \wedge \ldots \wedge A_n) \vee B) = \\ \text{(по закону де Моргана)} &= \neg \neg (A_1 \wedge \ldots \wedge A_n) \wedge \neg B = \\ &= A_1 \wedge \ldots \wedge A_n \wedge \neg B. \end{aligned}$$ ∎

1.5. Предикаты

В естественном языке встречаются высказывания, истинность которых может меняться в зависимости от объектов, о которых идет речь. Например, высказывание «синий кит есть млекопитающее» является истинным, а «белая акула есть млекопитающее» – ложным. В логике высказывание, зависящее от параметров, называется *предикатом*. Как и обычное высказывание, предикат принимает только два значения – 0 (ложь) или 1 (истина). Например, чтобы записать высказывания о принадлежности животных классу млекопитающих, можно ввести предикат МЛЕКОПИТАЮЩЕЕ(x), который обозначает фразу «x относится к классу млекопитающих». Тогда МЛЕКОПИТАЮЩЕЕ(синий кит) = 1, а МЛЕКОПИТАЮЩЕЕ(белая акула) = 0. Другими примерами предикатов являются любые математические равенства и неравенства, такие как $x + y < 3$ и $x^n + y^n = z^n$, которые могут быть верными или неверными в зависимости от конкретных значений переменных.

Предикаты могут иметь сложную внутреннюю структуру. Рассмотрим ее более подробно.

Пусть M – элементы некоторой предметной области. В нашем примере такими элементами являются виды животных «синий кит» и «белая акула».

Определение 1.8. Переменная или константа предметной области M называется *термом*. Высказывание $P(t_1, \ldots, t_n)$ относительно термов $t_1, \ldots, t_n$ называется *n-местным атомным предикатом*.

Чтобы построить высказывания, охватывающие все элементы предметной области, в логике предикатов вводится дополнительная логическая связка $\forall$, которая читается «для всех». Запись $(\forall x)P(x)$ читается так: «для всех x таких, что $P(x) = 1$» или «для всех x таких, что выполняется свойство $P(x)$».

Определим рекурсивно понятие *формулы логики предикатов*.

Определение 1.9.

1. Атомный предикат есть формула.
2. Если P – формула, то $\neg P$ – тоже формула.
3. Если P и Q – формулы, то $P \Rightarrow Q$ – тоже формула.
4. Если P – формула, а x – переменная предметной области, то $(\forall x)\, P$ – также формула.
5. Никаких других формул в логике предикатов нет.

Как и в логике высказываний, относительно предикатов допустимы логические связки $\wedge$, $\vee$ и $\Leftrightarrow$, которые также могут участвовать в построении формул. Кроме того, вводится еще одна дополнительная связка $\exists$, которая читается как «существует». По определению формула $(\exists x)P(x)$ эквивалентна формуле $\neg(\forall x)\neg P(x)$. Действительно, фраза «существует x такой, что выполняется свойство $P(x)$» эквивалентна другому выражению: «не верно, что для всех x не выполняется свойство $P(x)$».

Новые логические связки $\forall$ и $\exists$ называются *кванторами*: $\forall$ называется *квантором всеобщности*, а $\exists$ – *квантором существования*.

Пусть задан одноместный предикат $P(x)$, где переменная x может принимать значения $x_1, \ldots, x_n$. Тогда, по определению

$$\begin{array}{rcl} (\forall x)P(x) & = & P(x_1) \wedge \ldots \wedge P(x_n), \\ (\exists x)P(x) & = & P(x_1) \vee \ldots \vee P(x_n). \end{array} \qquad (1.1)$$

Для формул логики предикатов справедливы следующие равносильности, которые непосредственно следуют из соотношений (1.1):

(а) Комбинации кванторов:

$$\begin{aligned}(\forall\, x)(\forall\, y)P(x,y) &= (\forall\, y)(\forall\, x)P(x,y)\\ (\exists\, x)(\exists\, y)P(x,y) &= (\exists\, y)(\exists\, x)P(x,y)\\ (\forall\, x)(\exists\, y)P(x,y) &\neq (\exists\, y)(\forall\, x)P(x,y)\end{aligned}$$

(б) Комбинации кванторов и отрицаний:

$$\begin{aligned}\neg(\forall\, x)P(x) &= (\exists\, x)\neg P(x)\\ \neg(\exists\, x)P(x) &= (\forall\, x)\neg P(x)\end{aligned}$$

(в) Расширение области действия кванторов:

$$\begin{aligned}(\forall\, x)P(x) \wedge (\forall\, x)Q(x) &= (\forall\, x)(P(x) \wedge Q(x))\\ (\exists\, x)P(x) \vee (\exists\, x)Q(x) &= (\exists\, x)(P(x) \vee Q(x))\\ (\forall\, x)P(x) \vee (\forall\, x)Q(x) &\Rightarrow (\forall\, x)(P(x) \vee Q(x))\\ (\exists\, x)(P(x) \wedge Q(x)) &\Rightarrow (\exists\, x)P(x) \wedge (\exists\, x)Q(x)\end{aligned}$$

1.6. Метод математической индукции

Этот метод является эффективным инструментом построения доказательств в различных областях математики.

Пусть $A(n)$ – высказывание, зависящее от натурального числа n, истинность которого требуется доказать для всех натуральных n. Принцип математической индукции можно записать в виде следующей формулы логики предикатов:

$$[A(1) \wedge (\forall k)(A(k) \Rightarrow A(k+1))] \Rightarrow (\forall n)A(n).$$

Утверждение $A(1)$ называется *базой индукции*, а импликация $A(k) \Rightarrow A(k+1)$ – *индуктивным переходом.*

Если доказаны база индукции и индуктивный переход, то утверждение $A(n)$ является верным для всех натуральных n. При этом верность импликации достаточно доказать лишь для случая, когда истинно утверждение $A(k)$, которое называют *индуктивным предположением.*

Таким образом, доказательство с помощью математической индукции состоит из трех шагов:

1. Формулировка и доказательство базы индукции.
2. Формулировка индуктивного предположения.
3. Доказательство индуктивного перехода.

Рассмотрим пример. Пусть требуется доказать, что для любого положительного числа n число $n^3 - n$ делится на 3.

Сначала сформулируем и докажем базу индукции. Действительно, для $n = 1$ имеем: $1^3 - 1 = 0$ делится на 3.

Далее формулируем индуктивное предположение: пусть утверждение истинно для $n = k,\ k > 1$, то есть $k^3 - k = 3m$ для некоторого положительного m. Покажем, что истинность этого предположения влечет истинность утверждения для $n = k+1$, то есть $(k+1)^3 - (k+1)$ делится на 3.

Действительно, имеем

$$\begin{aligned}(k+1)^3 - (k+1) &= (k^3 + 3\,k^2 + 3\,k + 1) - (k+1) \\ &= (k^3 - k) + 3\,(k^2 + k) \\ &= 3\,m + 3\,(k^2 + k).\end{aligned}$$

Получили число, которое очевидно делится на 3. ∎

Задание. Доказать методом математической индукции, что плоскость, разделенная на области любым количеством прямых, может быть раскрашена в черный и белый цвет таким образом, что смежные области (имеющие общую границу) будут раскрашены в разные цвета.

Глава 2

Множества

2.1. Основные понятия и обозначения

Множество – фундаментальное неопределяемое математическое понятие. Под множеством понимают произвольную совокупность отличных друг от друга объектов, рассматриваемых как единое целое.

Объекты, из которых состоит множество называются его *элементами*. Множества обозначаются, как правило, заглавными буквами: A, B, $\mathbb{N}$, а элементы – строчными: a, x, n. Для записи множества используют фигурные скобки:

$$A = \{a_1, a_2, a_3\}, \quad \mathbb{N} = \{1, 2, 3, \ldots\}.$$

Если A – множество, а a – его элемент, то пишут $a \in A$, в противном случае – $a \notin A$. Множество, не содержащее ни одного элемента, называется *пустым* и обозначается символом $\varnothing$. Два множества A и B называются *равными*, если они состоят из одних и тех же элементов. В этом случае пишут $A = B$. Например, $\{1, 3, 7\} = \{7, 1, 3\}$.

Существует два основных способа задания множеств:

1. *Перечисление* всех его элементов.
2. *Описание свойства*, по которому можно определить, какие элементы принадлежат данному множеству.

Понятно, что первый способ годится только для множеств, число элементов которых невелико. Свойство задается в виде некоего условия (предиката), которое выполняется для всех элементов данного множества. Например, $x \in \mathbb{N}$ и $1 \leq x \leq 100$.

Для множества, заданного условием, используется запись вида

$$M = \{x \in \mathbb{N} \mid 1 \leq x \leq 100\},$$

которая читается следующим образом: «Множество всех натуральных x, таких что $1 \leq x \leq 100$».

2.2. Подмножества, операции над множествами

Определение 2.10. Множество A называется *подмножеством* множества B, пишут $A \subseteq B$, если любой элемент A является также элементом B:

$$(A \subseteq B) \Leftrightarrow (\forall x)((x \in A) \Rightarrow (x \in B)).$$

Например:

$$\{a, e\} \subseteq \{a, b, c, d, e\},$$
$$\{x \in \mathbb{N} \mid 10 \leq x \leq 20\} \subseteq \{x \in \mathbb{N} \mid 1 \leq x \leq 100\}.$$

Если A – подмножество B, то говорят, что A *содержится* в B или что B *включает* A. Если $A \subseteq B$ и при этом $A \neq B$, то говорят, что A есть *собственное подмножество* B, и пишут $A \subset B$.

По определению, $M \subseteq M$ и $\varnothing \subseteq M$ для любого множества M.

Рассмотрим следующие операции над множествами:

- *Объединение* множеств A и B есть множество
$$A \cup B = \{x \mid (x \in A) \vee (x \in B)\}.$$
- *Пересечение* множеств A и B есть множество
$$A \cap B = \{x \mid (x \in A) \wedge (x \in B)\}.$$
- *Разность* множеств A и B есть множество
$$A \setminus B = \{x \mid (x \in A) \wedge (x \notin B)\}.$$
- *Симметрическая разность* множеств A и B есть множество
$$A \oplus B = (A \setminus B) \cup (B \setminus A) = (A \cup B) \setminus (A \cap B)).$$
- *Дополнение* множества $A \subseteq U$ есть множество
$$\bar{A} = \{x \in U \mid x \notin A\}.$$
Эта операция определена при условии, что существует некое универсальное множество U (*универсум*), содержащее A. Тогда, по определению, $\bar{A} = U \setminus A$.

Пример. Пусть $A = \{1, 2, 3, 4, 5, 6\}$ и $B = \{2, 4, 6, 8\}$. Тогда

$$A \cup B = \{1, 2, 3, 4, 5, 6, 8\},$$
$$A \cap B = \{2, 4, 6\},$$
$$A \setminus B = \{1, 3, 5\},$$
$$A \oplus B = \{1, 3, 5, 8\},$$
$$\bar{A} = \{x \in \mathbb{N} \mid x > 6\}.$$

Операции над множествами удобно иллюстрировать с помощью *диаграмм Эйлера*, изображенных на рис. 2.1:

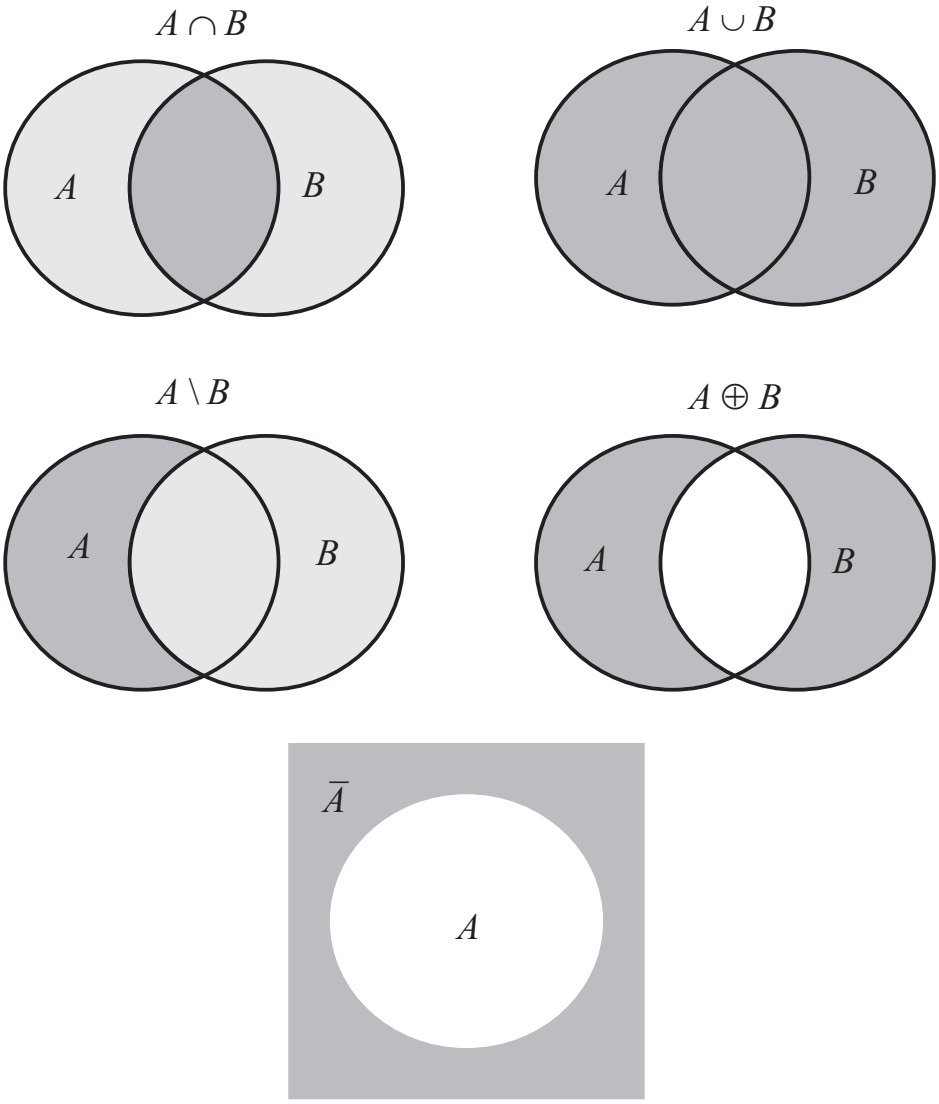


Рис. 2.1. Диаграммы Эйлера

Пусть U – универсум, $A, B, C \subset U$. Тогда выполняются следующие соотношения:

Идемпотентность:	$A \cap A = A$
	$A \cup A = A$
Коммутативность:	$A \cap B = B \cap A$
	$A \cup B = B \cup A$
Ассоциативность:	$A \cap (B \cap C) = (A \cap B) \cap C$
	$A \cup (B \cup C) = (A \cup B) \cup C$
Дистрибутивность:	$A \cap (B \cup C) = A \cap B \cup A \cap C$
	$A \cup B \cap C = (A \cup B) \cap (A \cup C)$
Законы *де Моргана*:	$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$
	$\overline{(A \cap B)} = \overline{A} \cup \overline{B}$
Инволютивность:	$\overline{\overline{A}} = A$
Свойства *нуля* и *единицы*:	$A \cup \overline{A} = U,\ A \cap \overline{A} = \varnothing$
	$A \cup \varnothing = A,\ A \cap \varnothing = \varnothing$
	$A \cup U = U,\ A \cap U = A$
Правила *поглощения*:	$(A \cup B) \cap A = A,\ (A \cap B) \cup A = A$

Все указанные свойства можно доказать формально на основе определений и формул логики высказываний. Например:

$$\begin{aligned}
\overline{A \cup B} &= \{x \mid x \notin A \cup B\} \\
&= \{x \mid \neg(x \in A \cup B)\} \\
&= \{x \mid \neg((x \in A) \vee (x \in B))\} \\
&= \{x \mid \neg(x \in A) \wedge \neg(x \in B)\} \\
&= \{x \mid (x \notin A) \wedge (x \notin B)\} \\
&= \{x \mid (x \in \overline{A}) \wedge (x \in \overline{B})\} \\
&= \overline{A} \cap \overline{B}.
\end{aligned}$$

Согласно свойствам коммутативности и ассоциативности, результат объединения или пересечения более чем двух множеств не зависит от порядка выполнения операций, поэтому часто используется следующая сокращенная форма записи:

$$A_1 \cup \ldots \cup A_n = \bigcup_{i=1}^{n} A_i \quad \text{и} \quad A_1 \cap \ldots \cap A_n = \bigcap_{i=1}^{n} A_i.$$

2.3. Декартово произведение множеств

Часто при рассмотрении множества из двух элементов $\{a, b\}$ важно учитывать порядок взаимного расположения этих элементов друг относительно друга. Таким образом, возникает потребность в новом термине – упорядоченная пара. Хотя понятие упорядоченности интуитивно понятное, тем не менее, хотелось бы определить его через уже известные нам понятия множества и подмножества.

Определение 2.11. *Упорядоченной парой* элементов a и b называется множество $(a, b) = \{\{a\}, \{a, b\}\}$.

Из данного определения следует простое утверждение:

Утверждение 2.1. $(a, b) = (c, d) \Leftrightarrow (a = c) \wedge (b = d)$.

Аналогичным образом определяется *упорядоченный набор* длины n элементов $a_1, \ldots, a_n$: $(a_1, \ldots, a_n) = \{\{a\}, \{a, b\}, \ldots, \{a_1, \ldots, a_n\}\}$. В наборе $(a_1, \ldots, a_n)$ элемент a_i называется его *i-й компонентой* или *координатой*. Упорядоченный набор часто называют *вектором* или *кортежем*, а вместо круглых скобок используют угловые: $\langle a_1, \ldots, a_n \rangle$.

Определение 2.12. *Декартовым произведением* множеств A и B, обозначается $A \times B$, называется множество всех упорядоченных пар таких, что первый элемент каждой пары принадлежит множеству A, а второй – B: $A \times B = \{(a, b) \mid (a \in A) \wedge (b \in B)\}$.

Например, для множеств $A = \{1, 2, 3\}$ и $B = \{a, b\}$ их декартово произведение есть множество

$$A \times B = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}.$$

Декартово произведение n множеств определяется аналогично:

$$A_1 \times \ldots \times A_n = \left\{ (a_1, \ldots, a_n) \mid \bigwedge_{i=1}^{n} (a_i \in A_i) \right\}.$$

Степенью множества A называется декартово произведение этого множества самого на себя:

$$A^n = \underbrace{A \times \ldots \times A}_{n \text{ раз}}$$

2.4. Отношения

Определение 2.13. *Бинарным отношением* R из множества A в множество B называется подмножество множества $A \times B$: $R \subseteq A \times B$.

Множество D_R первых элементов пар, входящих в R, называется *областью определения* бинарного отношения:

$$D_R = \{a \in A \mid (a, b) \in R\}.$$

Аналогично, множество V_R вторых элементов этих пар называется *множеством значений* бинарного отношения:

$$V_R = \{b \in B \mid (a, b) \in R\}.$$

Часто для краткости вместо $(a, b) \in R$ пишут aRb. Если $A = B$, то говорят, что R есть отношение *на множестве* A.

Рассмотрим два примера.

- Из арифметики нам хорошо известны отношения на множестве чисел, обозначаемые привычными символами: $=$, $\neq$, $<$, $>$, $\leq$, $\geq$. В частности, отношение $<$ на множестве $\mathbb{N}$ представляет собой множество $\{(n, m) \in \mathbb{N}^2 \mid (\exists k \in \mathbb{N})\,(m = n + k)\}$.
- Отношения $\subset$, $\subseteq$, $=$ определяются на множествах. Например, по определению, $\subseteq$ есть множество $\{(A, B) \mid (\forall x)(x \in A) \Rightarrow (x \in B)\}$.

Бинарные отношения называют также 2-местными. По аналогии, *n-местным* или *n-арным отношением* R на множествах $A_1, \ldots, A_n$ называется подмножество декартова произведения этих множеств:

$$R \in A_1 \times \ldots \times A_n.$$

Далее мы будем рассматривать только бинарные отношения.

Определение 2.14. Пусть $R_1 \in A \times C$ и $R_2 \in C \times B$ – два отношения. Композицией отношений R_1 и R_2 называется отношение $R \subseteq A \times B$, которое определяется следующим образом:

$$R = \{(a, b) \mid (a \in A) \wedge (b \in B) \wedge (\exists c \in C)((a, c) \in R_1 \wedge (c, b) \in R_2)\}.$$

Если отношение R является композицией отношений R_1 и R_2, то пишут $R = R_2 \circ R_1$. *Степенью* отношения R называется композиция вида:

$$R^n = \underbrace{R \circ \ldots \circ R}_{n \text{ раз}}.$$

Рассмотрим пример композиции родственных отношений. На рис. 2.2 изображена схема, где пунктиром обозначены пары, составляющие отношение «зять-теща», которое является композицией двух отношений – брачного («жених-невеста») и родительского («невеста-мама невесты»).

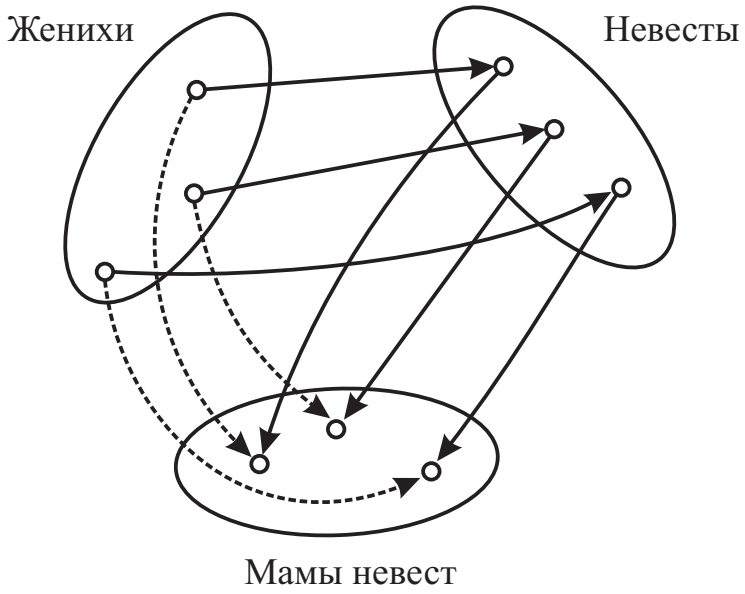


Рис. 2.2. Композиция родственных отношений:
«зять-теща» = «дочька-мама» ∘ «жених-невеста»

2.5. Типы и свойства отношений

Определение 2.15. Пусть R – отношение на множестве A. Тогда отношение R называется

пустым	если $R = \varnothing$,
рефлексивным	если $(\forall\, a)\ aRa$,
антирефлексивным	если $(\forall\, a)(\forall\, b)\ aRb \Rightarrow a \neq b$,
симметричным	если $(\forall\, a)(\forall\, b)\ aRb \Rightarrow bRa$,
антисимметричным	если $(\forall\, a)(\forall\, b)\ aRb \wedge bRa \Rightarrow a = b$,
транзитивным	если $(\forall\, a)(\forall\, b)(\forall\, c)\ aRc \wedge cRb \Rightarrow aRb$,
линейным	если $(\forall\, a)(\forall\, b)\ a \neq b \Rightarrow aRb \vee bRa$,
универсальным	если $R = A^2$.

Например, отношение $\leq$ на множестве $\mathbb{N}$ является рефлексивным, антисимметричным, транзитивным и линейным. Отношение $\subset$ на

множестве 2^M является антирефлексивным, антисимметричным и транзитивным, но не является линейным.

Отношение $R^{-1} = \{(b, a) \mid (a, b) \in R\}$ называется *обратным* к отношению R, а отношение $\overline{R} = \{(a, b) \mid (a, b) \notin R\}$ называется *дополнением* к отношению R.

Определение 2.16. Рефлексивное и транзитивное отношение на множестве A называется *предпорядком* на A.

Определение 2.17. Отношение предпорядка, обладающее свойством симметричности, называется отношением *эквивалентности*. Если R – отношение эквивалентности и $(a, b) \in R$, то пишут $a \sim b$ или $a \equiv b$ и говорят, что a и b *эквивалентны по отношению* R.

Примерами отношения эквивалентности являются отношение конгруэнтности фигур на плоскости, отношение параллельности прямых, отношение равенства чисел и множеств.

Еще один пример связан с понятиями покрытия и разбиения множеств. *Покрытием* множества X называется семейство подмножеств $\mathcal{X} = \{X_1, \ldots, X_k\}$ таких, что $X \subseteq \bigcup_{i=1}^{k} X_i$. Если при этом $X_i \cap X_j = \varnothing$ и $\bigcup_{i=1}^{k} X_i = X$, то семейство $\mathcal{X}$ называется *разбиением* множества X.

Верно следующее утверждение: всякое отношение эквивалентности на множестве X определяет разбиение множества X. И наоборот, всякое разбиение множества X, не содержащее пустых элементов, определяет отношение эквивалентности. Другими словами, на любом множестве X можно задать отношение эквивалентности, построив произвольное разбиение этого множества. Элементы X, относящиеся к одному подмножеству, входящему в разбиение, будут считаться эквивалентными.

Определение 2.18. Отношение предпорядка, обладающее свойством антисимметричности, называется отношением *частичного* порядка. Если R – отношение частичного порядка на множестве X, то пара (X, R) называется *частично-упорядоченным множеством*.

Если при этом отношение R является линейным, то множество (X, R) называют *линейно упорядоченным*.

Отношение $\subseteq$ на множестве 2^X всех подмножеств множества X является отношением частичного порядка, а отношение $\leq$ на множестве $\mathbb{N}$ натуральных чисел является отношением линейного порядка. Читателю предлагается доказать оба эти утверждения самостоятельно.

Частично-упорядоченное множество (X, R) удобно изображать в виде *диаграммы Хассе* (рис. 2.3), где точки, представляющие элементы $a, b \in X$, соединены отрезком тогда и только тогда, когда $(a, b) \in R$ и не существует $c \in X$ такого, что $(a, c) \in R \wedge (c, b) \in R$.

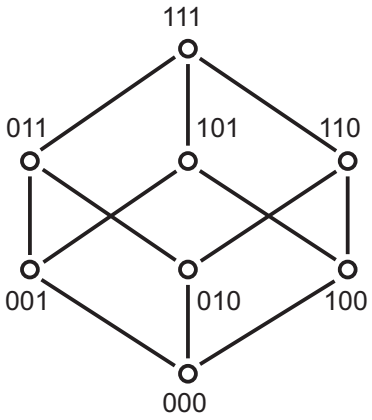


Рис. 2.3. Диаграмма Хассе частично-упорядоченного множества $(\mathbb{B}^3, \leq)$

2.6. Функции

Функцией или *отображением* называется отношение f из множества A в множество B такое, что $D_f = A$, и для любого элемента $a \in A$ существует единственный элемент $b \in B$ такой, что $(a, b) \in f$:

$$(\forall\, a \in A)\,(\forall\, b, c \in B)((a, b) \in f) \wedge ((a, c) \in f) \Rightarrow (b = c).$$

Такое свойство отношения f называется *однозначностью* или *функциональностью*.

Говорят, что функция f *отображает* множество A *в* множество B и используют запись $f : A \to B$. Если $(a, b) \in f$, то пишут $b = f(a)$. При этом элемент $a \in A$ называется *аргументом*, а $b \in B$ – *значением*[1] функции f.

Функция вида $f : A_1 \times \ldots \times A_n \to B$ называется *функцией n аргументов*. Если $((a_1, \ldots, a_n), b) \in f$, то пишут $b = f(a_1, \ldots, a_n)$.

Определение 2.19. Пусть $f : A \to B$. Тогда f называется:

- *инъективной* (*инъекцией*), если $f(a) = f(b) \Rightarrow a = b$ (каждому значению функции соответствует в точности один ее аргумент);

[1] Другие названия: a – *прообраз* b, а b – *образ* a. Или: a – *независимая переменная*, а b – *зависимая переменная*.

- *сюръективной* (*сюръекцией*), если $V_f = B$, то есть

$$(\forall\, b \in B)\ (\exists a \in A)\ ((a, b) \in f)$$

(говорят, что сюръективная функция отображает множество A *на* множество B);

- *биективной* (*биекцией* или *взаимно-однозначной*), если она является одновременно инъективной и сюръективной.

Рис. 2.4 иллюстрирует отличия между отношениями и функциями различных типов.

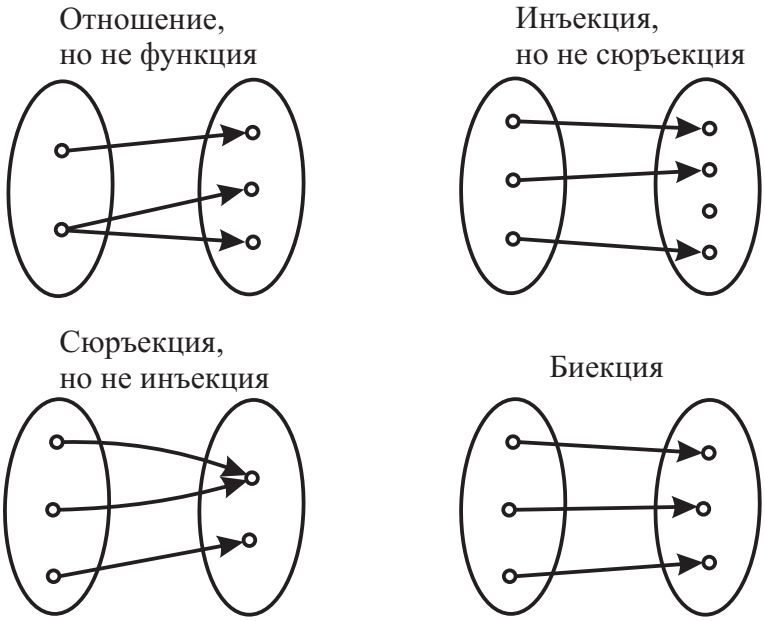


Рис. 2.4. Типы функций

Теорема 2.2. *Если f – инъекция, то $a \neq b \Rightarrow f(a) \neq f(b)$.*

Доказательство. Утверждение теоремы есть контрапозиция формулы, используемой в определении инъекции. ∎

Поскольку функции суть отношения, к ним применимы все понятия, относящиеся к отношениям в целом. Пусть $f : A \to B$ – функция. Тогда отношение f^{-1}, обратное к отношению f, называется функцией, *обратной к* f. По определению, $y = f(x) \Leftrightarrow x = f^{-1}(y)$.

Теорема 2.3. *Пусть $f : A \to B$ – биекция. Тогда f^{-1} – также биекция и $D_{f^{-1}} = B$.*

Доказательство. Покажем, что

1. $D_{f^{-1}} = B$,
2. f^{-1} есть функция,
3. f^{-1} есть инъекция,
4. f^{-1} есть сюръекция.

(1) По условию теоремы f – сюръекция. Значит,

$$(\forall b \in B)\,(\exists a \in A)\,(a, b) \in f,$$

откуда

$$(\forall b \in B)\,(\exists a \in A)\,(b, a) \in f^{-1},$$

то есть $D_{f^{-1}} = B$.

(2) Пусть $(c, a) \in f^{-1}$ и $(c, b) \in f^{-1}$. Тогда $(a, c) \in f$ и $(b, c) \in f$, то есть $f(a) = c$ и $f(b) = c$. Поскольку f – инъекция, значит $a = b$, откуда заключаем, что f^{-1} есть функция по определению.

(3) Пусть $f^{-1}(b) = a$ и $f^{-1}(c) = a$, то есть $(b, a) \in f^{-1}$ и $(c, a) \in f^{-1}$. Тогда $(a, b) \in f$ и $(a, c) \in f$. Поскольку f – функция, имеем $b = c$. Значит f^{-1} – инъекция.

(4) Имеем: $D_f = A \Rightarrow (\forall a \in A)\,(\exists b \in B)\,(a, b) \in f \Rightarrow (b, a) \in f^{-1}$, значит f^{-1} – сюръекция. ∎

Пусть $f : A \to B$ и $g : B \to C$ – две функции. Композиция отношений f и g называется *композицией функций*: $g \circ f : A \to C$. Из определения композиции отношений следует, что $(g \circ f)(a) = g(f(a))$.

Пример. Пусть $f(x) = x^2 + 3,\ g(y) = \sqrt{y}$. Тогда

$$(g \circ f)(x) = g(f(x)) = \sqrt{x^2 + 3}.$$

Теорема 2.4. *Пусть $f : A \to C$ и $g : C \to B$ – биекции. Тогда*

1. $(\forall a \in A)\ (f^{-1} \circ f)(a) = a$.
2. $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$.

Доказательство. Первое утверждение следует непосредственно из определения обратной функции. Докажем второе утверждение. Действительно, пусть $f : A \to C$ и $g : C \to B$ – функции и для некоторых

$a \in A$, $b \in B$ и $c \in C$ выполняется $f(a) = c$ и $g(c) = b$. Тогда $(g \circ f)(a) = b$ и, по определению обратной функции, $(g \circ f)^{-1}(b) = a$. С другой стороны,

$$(f^{-1} \circ g^{-1})(b) = f^{-1}(g^{-1}(b)) = f^{-1}(c) = a.$$

■

2.7. Мощность множества

В предыдущих разделах мы специально не рассматривали такое важное свойство множества как его размер, и для этого были серьезные основания. Если размер конечного множества можно определить как число его элементов, то как поступить с бесконечным множеством? Можно ли сравнивать между собой бесконечные множества? Можно ли ответить на вопрос, каких чисел больше – натуральных или действительных?

Для того, чтобы сравнивать бесконечные множества нам потребуется ввести понятие *мощности*, основанное на уже определенных ранее понятиях отношения эквивалентности и взаимно-однозначной функции. Начнем с определения *конечного* множества и его мощности.

Определение 2.20. Пустое множество есть *множество мощности* 0. Если существует взаимно-однозначное соответствие между множеством A и множеством $\{1, \ldots, n\}$, то говорят, что A есть *конечное множество мощности* n и пишут $|A| = n$.

Например, множество $A = \{a, b, c\}$ есть множество мощности 3 ($|A| = 3$), так как существует биекция из множества $\{1, 2, 3\}$ в A.

Непустое множество, не являющееся конечным, называется *бесконечным*. Обобщим понятие мощности на бесконечные множества.

Определение 2.21. Множества A и B называются *равномощными*, если существует биекция из A в B.

Введем для множеств отношение эквивалентности M таким образом, что эквивалентными буду считаться равномощные множества. (Докажите, что отношение равномощности есть отношение эквивалентности.) Тогда мощность множества A можно определить как *класс* всех множеств, эквивалентных A относительно M.

Определение 2.22. Множество, равномощное множеству $\mathbb{N}$ натуральных чисел, называются *счетным*.

Другими словами, все элементы счетного множества A можно занумеровать натуральными числами и определить таким образом бесконечную последовательность вида

$$(a_n)_{n=1}^{\infty} = a_1, a_2, \ldots$$

Теорема 2.5. *Любое подмножество счетного множества либо конечно, либо счетно.*

Доказательство. Пусть A – счетное множество и $B \subseteq A$. По условию теоремы существует бесконечная последовательность $(a_n)_{n=1}^{\infty}$, включающая все элементы множества A. Определим отображение $\varphi : \mathbb{N} \to B$ следующим образом. Пусть $\varphi(1) = a_{m_1}$, где m_1 – минимальный номер элемента последовательности (a_n), такой что $a_{m_1} \in B$. Пусть далее $\varphi(2) = a_{m_2}$, где m_2 – следующий после m_1 номер, такой что $a_{m_2} \in B$ и т. д. Очевидно, φ – биекция. Если существует такое число $N > 0$, что $B = \{a_{m_i} \in A \,|\, 1 \leq i \leq N\}$, то, по определению, B – конечное множество мощности N, если нет, то B – счетное. ∎

Теорема 2.6. *Если A и B – счетные множества, то их объединение множество $A \cup B$ также также счетное.*

Доказательство. Сначала докажем данное утверждение для случая, когда A и B – непересекающиеся множества. По условию теоремы существуют биективные отображения $\alpha : \mathbb{N} \to A$ и $\beta : \mathbb{N} \to B$. Определим новое биективное отображение $\gamma : \mathbb{N} \to A \cup B$ следующим образом:

$$\gamma(n) = \begin{cases} \alpha\left(\frac{n+1}{2}\right), & \text{если } n \text{ нечетное,} \\ \beta\left(\frac{n}{2}\right), & \text{если } n \text{ четное.} \end{cases}$$

(Например, $\gamma(1) = \alpha(1)$, $\gamma(2) = \beta(1)$, $\gamma(3) = \alpha(2)$, $\gamma(4) = \beta(2)$ и т. д.)

Пусть теперь A и B – произвольные счетные множества. Имеет место соотношение $A \cup B = (A \backslash B) \cup B$, где $A \backslash B \subset A$, а множества $A \backslash B$ и B не пересекаются. Из теоремы 2.5 следует, что множество $A \backslash B$ счетно. Значит $A \cup B$ также счетно. ∎

Теорема 2.7. *Если S – счетное множество, то множество $S \times S$ также счетное.*

Доказательство. Покажем сначала, что множество $\mathbb{N} \times \mathbb{N}$ счетно. Построим взаимно-однозначное соответствие $\varphi : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ следующим образом. Положим $\varphi(1) = (1, 1)$. Пусть $\varphi(i) = (m, n)$ для

некоторого $i \geq 1$. Тогда

$$\varphi(i+1) = \begin{cases} (m+1, n-1), & \text{если } n \geq 2, \\ (1, m+n) & \text{иначе.} \end{cases}$$

Определенная таким образом функция перечисляет по порядку пары с одинаковой суммой элементов, равной сначала $2 = (1+1)$, затем $3 = (1+2) = (2+1)$, и т. д., и очевидно является биективной. Следовательно, множество $\mathbb{N} \times \mathbb{N}$ счетно. Установим теперь взаимно-однозначное соответствие между этим множеством и множеством $S \times S$. Поскольку множество S – счетное, существует биекция $\theta : \mathbb{N} \to S$. Определим искомое отображение $\Theta : \mathbb{N} \times \mathbb{N} \to S \times S$ следующим образом: $\Theta((m,n)) = (\theta(m), \theta(n))$. ∎

Теорема 2.8. *Множество Q_+ положительных рациональных чисел счетное.*

Доказательство. Из алгебры звестно, что любое положительное рациональное число представимо единственным образом в виде дроби $\frac{p}{q}$, где p и q – взаимно простые натуральные числа. Другими словами, существует биективное отображение множества Q_+ на подмножество M множества $\mathbb{N} \times \mathbb{N}$, состоящее из пар вида (p,q), образованных взаимно простыми числами. Из теорем 2.5 и 2.7 следует, что множество M – счетное. Значит множество Q_+ также счетное. ∎

Возникает вопрос: а существуют ли вообще множества, не являющиеся счетными? Ответ дает следующая теорема.

Теорема 2.9. *Множество $I = \{x \in \mathbb{R} \mid 0 < x < 1\}$ не является счетным.*

Доказательство. Применим так называемый *диагональный метод Кантора*. Известно, что каждому действительному числу из интервала $(0,1)$ соответствует одна и только одна бесконечная десятичная дробь вида $0, a_1 a_2, \ldots$ с бесконечным числом отличных от нуля цифр (конечные десятичные дроби имеют бесконечное представление, например, $0,123 = 0,122999\ldots$).

Предположим, что теорема не верна и существует взаимно-однозначное соответствие между множествами $\mathbb{N}$ и I. Тогда числа из I можно выписать по порядку:

$$
\begin{aligned}
a_1 &= 0, a_{11}\, a_{12}\, a_{13} \ldots a_{1k} \ldots, \\
a_2 &= 0, a_{21}\, a_{22}\, a_{23} \ldots a_{2k} \ldots, \\
\ldots & \quad \ldots\ldots\ldots\ldots\ldots\ldots \\
a_k &= 0, a_{k1}\, a_{k2}\, a_{k3} \ldots a_{kk} \ldots, \\
\ldots & \quad \ldots\ldots\ldots\ldots\ldots\ldots
\end{aligned}
$$

Построим теперь действительное число $b = 0, b_1\, b_2\, b_3 \ldots$, которое, с одной стороны, принадлежит интервалу $(0, 1)$, а с другой – не совпадает ни с одним из чисел a_k из этого интервала. Для этого выберем цифры b_i такими, что $b_i \neq a_{ii}$. Получаем, что число b не равно никакому из чисел a_k, так как не совпадают их k-е разряды b_k и a_{kk}. Полученное противоречие доказывает теорему. ∎

Определение 2.23. Бесконечные множества, не являющиеся счетными, называются *несчетными*. Множества, равномощные множеству I, называются *множествами мощности континуум*.

Из теоремы 2.9 следуют важные утверждения.

Следствие 2.2. *Множество $\mathbb{R}$ действительных чисел несчетно.*

Действительно, если бы $\mathbb{R}$ было счетным, то по теореме 2.5 любое его подмножество, включая I, должно быть счетным.

Предлагаем читателю следующие утверждения доказать самостоятельно.

Следствие 2.3. *Множество иррациональных чисел несчетно.*

Следствие 2.4. *Множество 2^A всех подмножеств любого счетного множества A несчетно.*

(Подсказка: не теряя общности, возьмите $\mathbb{N}$ в качестве счетного множества и примените диагональный метод Кантора).

Глава 3

Элементы комбинаторики

Изучая теорию множеств мы умышленно избегали давать числовые характеристики множеств и отношений. Исследованием этих характеристик занимается специальная область дискретной математики – комбинаторика.

3.1. Основной принцип комбинаторики

Рассмотрим следующую задачу: сколько существует различных вариантов перелета из Гомеля в Нью-Йорк, если существуют 2 авиарейса из Гомеля в Минск, 5 рейсов из Минска в Лондон и 6 рейсов из Лондона в Нью-Йорк? Обозначим через a_1 и a_2 авиарейсы из Гомеля в Минск, через $b_1, \ldots, b_5$ – авиарейсы из Минска в Лондон и через $c_1, \ldots, c_6$ – авиарейсы из Лондона в Нью-Йорк. Очевидно, решение задачи сводится к подсчету числа всевозможных троек вида (a_i, b_j, c_k) или, другими словами, элементов декартова произведения

$$A \times B \times C,$$

где через A, B и C обозначены множества соответствующих авиарейсов. В виду особой важности сформулируем это решение в виде следующего утверждения.

Утверждение 3.2 (Основной принцип комбинаторики). *Пусть требуется выполнить k действий, причем i-е действие можно выполнить n_i способами. Тогда все k действий можно выполнить $n_1 \times \ldots \times n_k = \prod_{i=1}^{k} n_i$ способами.*

Из этого простого принципа следует ряд полезных утверждений, которые нам понадобятся в дальнейшем.

Следствие 3.1.

1. $|\mathbb{B}^n| = 2^n$,
2. $|2^M| = 2^{|M|}$,
3. $|A_1 \times \ldots \times A_n| = \prod_{i=1}^{n} |A_i|$ *(правило произведения)*,
4. $|A^n| = |A|^n$.

Пример. Число векторов, компоненты которых могут принимать два значения (например, 0 и 1), равно 2^n. Если компоненты могут принимать три значения – то 3^n и т.д.

Утверждение 3.3 (Принцип Дирихле). *Если семейство подмножеств $\mathcal{X} = \{X_1, \ldots, X_k\}$ множества X является покрытием множества X и $|X| = n$, то существует подмножество $X_i \in \mathcal{X}$ такое, что $|X_i| \geq \dfrac{n}{k}$.*

(Доказательство от противного: помечаем элементы подмножеств.)

Утверждение 3.4 (Правило суммы).

Для разбиения $\mathcal{X} = \{X_1, \ldots, X_k\}$ множества X справедливо

$$|X| = \sum_{i=1}^{n} |X_i|.$$

Теорема 3.1 (Мощность объединения множеств).

$$|X_1 \cup X_2| = |X_i| + |X_2| - |X_1 \cap X_2|.$$

Доказательство. Нетрудно показать, что

$$X_1 \cup X_2 = (X_1 \setminus X_2) \cup (X_2 \setminus X_1) \cup (X_1 \cap X_2).$$

Поскольку $X_1 \setminus X_2$, $X_2 \setminus X_1$ и $X_1 \cap X_2$ – непересекающиеся множества, из правила сложения имеем:

$$|X_1 \cup X_2| = |X_1 \setminus X_2| + |X_2 \setminus X_1| + |X_1 \cap X_2|. \tag{3.1}$$

С другой стороны

$$X_1 = (X_1 \setminus X_2) \cup (X_1 \cap X_2) \;\Rightarrow\; |X_1 \setminus X_2| = |X_1| - |X_1 \cap X_2|,$$
$$X_2 = (X_2 \setminus X_1) \cup (X_1 \cap X_2) \;\Rightarrow\; |X_2 \setminus X_1| = |X_2| - |X_1 \cap X_2|.$$

Подставляем в равенство (3.1) и получаем утверждение теоремы. ∎

Это утверждение является частным случаем общего принципа «включения и исключения», дающего формулу для подсчета элементов объединения любого количества множеств.

3.2. Размещения, перестановки и сочетания

В задачах комбинаторного анализа интерес представляют не сами элементы множества, а их количество. Поэтому любое множество, состоящее из n элементов, называют *n-множеством*, а вектор длины n – *n-вектором*. Введем также обозначение $[n] = \{i \in \mathbb{N} \,|\, 1 \leq i \leq n\}$.

Определение 3.1. Произвольный m-вектор с координатами из n-множества называется *размещением с повторениями из n элементов по m*.

Например, 3-вектора $(8, 8, 10)$, $(1, 10, 7)$ и $(2, 2, 2)$, состоящие из элементов множества $[10]$, являются размещениями с повторениями из 10 по 3. Возникает вопрос: а сколько всего существует различных размещений с повторениями из 10 по 3? Ответ следует непосредственно из основного принципа комбинаторики.

Теорема 3.2. *Число различных размещений с повторениями из n по m равно n^m.*

Определение 3.2. m-вектор с попарно различными координатами из n-множества называется *размещением из n элементов по m*.

Оценим число P_n^m всех различных размещений из n по m. Но сначала введем важное обозначение, широко применяемое в комбинаторных формулах.

Произведение первых n натуральных чисел, обозначается $n!$, называется *n-факториалом*: $n! = 1 \times \ldots \times n$. По определению $0! = 1$.

Заметим, что $n!$ – очень быстро растущая функция:

$$\begin{aligned}
0! &= 1, \\
1! &= 1, \\
2! &= 2, \\
3! &= 6, \\
4! &= 24, \\
5! &= 120, \\
6! &= 720, \\
7! &= 5040, \\
8! &= 40320, \\
9! &= 362880, \\
10! &= 3628800, \\
&\ldots
\end{aligned}$$

Теорема 3.3. *Число различных размещений из n по m равно*

$$P_n^m = n\,(n-1)\ldots(n-m+1) = \frac{n!}{(n-m)!}.$$

Доказательство. Действительно, первый элемент m-вектора можно выбрать n способами. Для каждого из вариантов выбора первого элемента существует ровно $n-1$ вариант выбора второго элемента, не совпадающего с первым. Для каждого из вариантов выбора первых двух элементов третий можно выбрать уже $(n-2)$-мя способами и т. д. Таким образом, из основного принципа комбинаторики получаем утверждение теоремы. ∎

Пример. По теореме $P_3^2 = \dfrac{3!}{(3-2)!} = 3! = 6$. Чтобы убедиться в этом, выпишем всевозможные размещения из 3 по 2 для элементов множества [3]: $(1,2)$, $(1,3)$, $(2,1)$, $(2,3)$, $(3,1)$, $(3,2)$.

Определение 3.3. Размещение из n по n называется перестановкой n-множества.

Все перестановки данного множества можно получить, выписывая их в виде векторов, отличающихся только порядком следования элементов. Обозначим через P_n число всех перестановок n-множества.

Теорема 3.4. $P_n = n!$.

Доказательство. Следует непосредственно из теоремы 3.3. ∎

Пример. Для множества [3] имеем: $P_3 = 3! = 6$. Соответствующие перестановки: $(1,2,3)$, $(1,3,2)$, $(2,1,3)$, $(2,3,1)$, $(3,1,2)$, $(3,2,1)$.

Размещения с повторениями и без являются векторами, то есть *упорядоченными* подмножествами некоторого множества. Оценим теперь число обычных (неупорядоченных) подмножеств. Обозначим через $\binom{X}{m}$ множество всех m-подмножеств множества X.

Определение 3.4. m-подмножество n-множества называется *сочетанием из n элементов по m*. Число различных сочетаний из n по m обозначается C_n^m или $\binom{n}{m}$, $0 \le m \le n$.

Теорема 3.5.

$$C_n^m = \frac{n!}{m!(n-m)!}. \tag{3.2}$$

Доказательство. Рассмотрим все m-размещения данного n-множества и сгруппируем их таким образом, чтобы в каждую группу входили размещения, отличающиеся друг от друга только порядком своих элементов. Каждой группе можно поставить в соответствие в точности одно m-подмножество (сочетание) данного n-множества, состоящее из элементов m-размещений, входящих в эту группу. Таким образом, искомое число сочетаний равно числу групп. Имеем: всего размещений – P_n^m, число размещений в группе равно числу различных перестановок элементов m-размещения, то есть $m!$, откуда $C_n^m = P_n^m/m!$. ∎

Пример. Подсчитаем число костей домино с попарно различными значениями. Указанное число равно числу способов выбора двух разных значений из семи возможных (от нуля («пусто») до шести), то есть числу сочетаний из 7 по 2. Получаем

$$C_7^2 = \frac{7!}{2!\,5!} = \frac{5! \times 6 \times 7}{2 \times 5!} = 42/2 = 21.$$

Непосредственно из формулы (3.2) вытекают следующие полезные соотношения:

$$1. \quad C_n^m = C_n^{n-m}, \tag{3.3}$$

$$2. \quad C_n^m = C_{n-1}^m + C_{n-1}^{m-1} \quad \text{(треугольник Паскаля)}, \tag{3.4}$$

$$3. \quad \sum_{m=0}^{n} C_n^m = 2^n. \tag{3.5}$$

Равенство (3.3) очевидно. Докажем формулу (3.4). Действительно,

$$C_{n-1}^m + C_{n-1}^{m-1} = \frac{(n-1)!}{m!(n-m-1)!} + \frac{(n-1)!}{(m-1)!(n-m)!} =$$

$$= \frac{(n-1)!(n-m) + (n-1)!m}{m!(n-m)!} = \frac{n!}{m!(n-m)!} = C_n^m.$$

Формула (3.5) следует из определения C_n^m и оценки числа всех подмножеств n-множества.

3.3. Бином Ньютона

Рассмотрим формулу разложения квадрата суммы, хорошо известную из школьного курса математики:

$$(x+y)^2 = x^2 + 2\,xy + y^2,$$

Нетрудно заметить, что ее можно переписать, используя числа C_n^m:

$$(x+y)^2 = C_2^0\, x^n y^0 + C_2^1\, x^1 y^1 + C_2^2\, x^0 y^2.$$

Оказывается, существует более общая закономерность, благодаря которой числа C_n^m получили специальное название – *биномиальные коэффициенты*.

Теорема 3.6 (Формула бинома Ньютона).

$$(x+y)^n = C_n^0\, x^n y^0 + C_n^1\, x^{n-1} y^1 + \ldots + C_n^n\, x^0 y^n = \sum_{m=0}^{n} C_n^m\, x^{n-m} y^m.$$

Доказательство. Используем индукцию по n. Для $n = 1$ имеем:

$$(x+y)^1 = C_1^0\, x^1 y^0 + C_1^1\, x^0 y^1 = x + y.$$

Равенство верно, следовательно база индукции доказана. Предположим, что формула бинома верна для степени $n-1$. Покажем, что она верна для степени n. Имеем:

$$\begin{aligned}
(x+y)^n &= (x+y) \sum_{m=0}^{n-1} C_{n-1}^m\, x^{n-m-1} y^m \\
&= \sum_{m=0}^{n-1} C_{n-1}^m\, x^{n-m} y^m + \sum_{m=0}^{n-1} C_{n-1}^{(m+1)-1}\, x^{n-(m+1)} y^{m+1} \\
&= \sum_{m=1}^{n-1} C_{n-1}^m\, x^{n-m} y^m + C_{n-1}^0\, x^n y^0 \\
&+ \sum_{m=1}^{n-1} C_{n-1}^{m-1}\, x^{n-m} y^m + C_{n-1}^{n-1}\, x^0 y^n.
\end{aligned}$$

Поскольку $C_{n-1}^0 = C_n^0 = 1$ и $C_{n-1}^{n-1} = C_n^n = 1$, а также в силу равенства (3.4), последнее выражение преобразуется к виду:

$$C_n^0\, x^n y^0 + \sum_{m=1}^{n-1} \left(C_{n-1}^m + C_{n-1}^{m-1} \right) x^{n-m} y^m + C_n^n\, x^0 y^n = \sum_{m=0}^{n} C_n^m\, x^{n-m} y^m.$$

■

Из формулы бинома можно получить множество других полезных соотношений, в частности, равенство (3.5) – для этого достаточно в формуле (3.2) положить $x = 1$ и $y = 1$.

3.4. Мультимножества и сочетания с повторениями

Рассмотрим понятие мультимножества, то есть множества с повторяющимися элементами.

Определение 3.5. Пусть задано конечное множество X и функция $f : X \to \mathbb{N}$. Пара (X, f) называется *мультимножеством* на множестве X с *функцией кратности* f.

Значение $f(x)$, $x \in X$, есть число повторений элемента x в данном мультимножестве. Два мультимножества на множестве X называются *равными*, если равны их функции кратности. Если число элементов X невелико, то для описания мультимножества используют сокращенную запись, указывая кратности всех элементов множества X. Например, мультимножество $\{1, 1, 1, 1, 1\}$ на множестве $X = [3]$ можно записать в виде $\{1^5, 2^0, 3^0\}$. Если зафиксировать порядок записи элементов множества X, то для задания мультимножества достаточно просто указать значения функции кратности: $\{1, 1, 1, 1, 1\} = (5, 0, 0)$.

Число $\sum_{x \in X} f(x)$ называется *мощностью* мультимножества. Для мультимножеств мощности m введем более короткий термин *m-мультимножество*. Множество всех m-мультимножеств на множестве X обозначается $\left(\!\binom{X}{m}\!\right)$. Например:

$$\left(\!\binom{[3]}{2}\!\right) = \{\{1, 1\}, \{2, 2\}, \{3, 3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}\}$$

или через функции кратности:

$$\left(\!\binom{[3]}{2}\!\right) = \{(2, 0, 0), (0, 2, 0), (0, 0, 2), (1, 1, 0), (1, 0, 1), (0, 1, 1)\}.$$

Определение 3.6. m-мультимножество на n-множестве называется *сочетанием с повторениями* из n элементов по m. Число всех сочетаний с повторениями из n по m обозначается H_n^m или $\left(\!\binom{n}{m}\!\right)$.

Теорема 3.7. $H_n^m = C_{n+m-1}^m$.

Доказательство. Построим взаимно-однозначное соответствие между сочетаниями с повторениями из n по m и сочетаниями без повторений из $n + m - 1$ по m, то есть между элементами множеств $\left(\!\binom{[n]}{m}\!\right)$ и $\binom{[n+m-1]}{m}$. Пусть некоторе m-сочетание с повторениями из элементов

множества $[n]$ состоит из элементов $a_1, \ldots, a_m,\ 1 \le a_i \le n$. Поскольку порядок элементов в мультимножестве не имеет значения, предположим, не теряя общности, что эти элементы расположены в порядке неубывания значений: $a_1 \le \ldots \le a_m$. Поставим в соответствие этому m-сочетанию с повторениями m-сочетание без повторений $\{b_1, \ldots, b_m\},\ 1 \le b_i \le n + m - 1$ по следующему правилу: $b_i = a_i + i - 1,\ 1 \le i \le m$. Очевидно, построенное отображение является биективным, а элементы построенного m-сочетания попарно различны, поскольку $b_1 < \ldots < b_m$. ∎

Пример. Сколько существует костей домино? Каждая кость содержит два из семи возможных значений, которые могут повторяться. Поэтому ответ такой:

$$H_7^2 = C_8^2 = \frac{8!}{2!\,6!} = 56/2 = 28.$$

3.5. Числовые разложения и разбиения

Определение 3.7. *m-разложением* натурального числа n называется m-вектор $(x_1, \ldots, x_m)$ натуральных чисел, такой что

$$x_1 + \ldots + x_m = n.$$

Например, для $n = 4$ существуют ровно три 3-разложения:

$$1 + 1 + 2,\ 1 + 2 + 1,\ 2 + 1 + 1.$$

Возникает вопрос: сколько существует различных m-разложений числа n? Представим число n схематично в виде n точек:

$$\cdot\ \cdot\ \cdot\ \cdot \quad (n = 4)$$

Тогда разложение числа n в виде m слагаемых можно отобразить с помощью $m - 1$-го разделителя, которые можно расположить в $n - 1$-й позиции, например:

$$\cdot\,|\,\cdot\ \cdot\,|\,\cdot \quad (1 + 2 + 1)$$

Значит, число различных m-разложений числа n равно C_{n-1}^{m-1}.

Попутно мы доказали следующую теорему.

Теорема 3.8. *Число различных решений в натуральных числах уравнения*

$$x_1 + \ldots + x_m = n$$

равно C_{n-1}^{m-1}.

Определение 3.8. *Слабым m-разложением* натурального числа n называется m-вектор целых неотрицательных чисел $(x_1, \ldots, x_m)$, такой что

$$x_1 + \ldots + x_m = n.$$

Если в этом уравнении положить $y_i = x_i + 1$, то получим (обычное) m-разложение числа $n + m$:

$$y_1 + \ldots + y_m = n + m.$$

Примененное здесь отображение векторов $(y_1, \ldots, y_m)$ на вектора $(x_1, \ldots, x_m)$ биективно, откуда следует, что число различных слабых m-разложений числа n равно C^{m-1}_{n+m-1}.

Например, число слабых 2-разложений числа 3 равно $C^1_4 = 4$:

$$0 + 3,\ 3 + 0,\ 1 + 2,\ 2 + 1,$$

что соответствует числу (обычных) 2-разложений числа 5:

$$1 + 4,\ 4 + 1,\ 2 + 3,\ 3 + 2.$$

Определение 3.9. *m-разбиением* натурального числа n называется m-мультимножество $\{x_1, \ldots, x_m\}$ на множестве натуральных чисел, такое что

$$x_1 + \ldots + x_m = n.$$

Разбиение рассматривается как представление числа в виде суммы натуральных слагаемых, порядок следования которых игнорируется. Таким образом, выражения $1 + 2 + 1$ и $2 + 1 + 1$ задают одно и тоже разбиение.

Например, существует ровно пять различных разбиений числа 4:

$$4,\ 3 + 1,\ 2 + 2,\ 2 + 1 + 1,\ 1 + 1 + 1 + 1.$$

Одна любопытная формула

Ответ на вопрос, сколько существует различных разбиений натурального числа n, был получен лишь в начале XX века. Асимптотическое представление для этого числа было впервые опубликовано английским математиком Г. Харди (1887-1947) и индийским математиком С. Рамануйана (1887-1920), а затем уточнено немецким математиком Г.Радемахером (1892-1969).

Теорема 3.9. *Пусть $p(n)$ – число различных разбиений числа $n \in \mathbb{N}$. Тогда*

$$p(n) = \frac{1}{\pi\sqrt{2}} \sum_{k=1}^{\infty} A_k(n)\sqrt{k}\frac{d}{dn}\left(\frac{\sinh\left(\frac{\pi}{k}\sqrt{\frac{2}{3}(n-\frac{1}{24})}\right)}{\sqrt{n-\frac{1}{24}}}\right),$$

где

$$A_k(n) = \sum_{\substack{0 \le m < k; \\ m \text{ и } k \text{ – взаимно} \\ \text{простые}}} \exp(\pi\, i\, \sigma(m,k) - 2\,\pi\, i\frac{m}{k}),$$

$\sigma(m,k)$ *– сумма Дедекинда:*

$$\sigma(m,k) = \frac{1}{4k}\sum_{n=1}^{k-1} \cot\frac{\pi\, n}{k}\cot\frac{\pi\, n\, m}{k}.$$

Как заметил Г. Эндрюс [...], «это необычайное тождество, в котором левой частью служит простая арифметическая функция $p(n)$, а правой – бесконечный ряд, включающий в себя π, квадратные корни, комплексные корни из единицы и производные гиперболических функций...».

3.6. Подстановки

Определение 3.10. Взаимно однозначная функция $f : X \to X$ называется *подстановкой* на множестве X.

Не ограничивая общности, будем считать, что $X = [n]$. Тогда подстановку удобно записывать в виде таблицы из двух строк: первая строка содержит аргументы функции f, а вторая – ее значения. Например:

$$f = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 4 & 1 & 2 \end{vmatrix}, \quad g = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 2 & 5 & 4 \end{vmatrix}.$$

К подстановкам применима операция композиции:

$$g \circ f = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 5 & 1 & 3 \end{vmatrix}.$$

Функция, обратная к подстановке, называется *обратной подстановкой*. Заметим, что по определению обратная подстановка всегда существует. Для подстановки f имеем:

$$f^{-1} = \left|\begin{array}{ccccc} 3 & 5 & 4 & 1 & 2 \\ 1 & 2 & 3 & 4 & 5 \end{array}\right| = \left|\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 1 & 3 & 2 \end{array}\right|.$$

Подстановки имеют удобное графическое представление в виде ориентированных графов (рис. 3.1). *Циклом* подстановки f называется

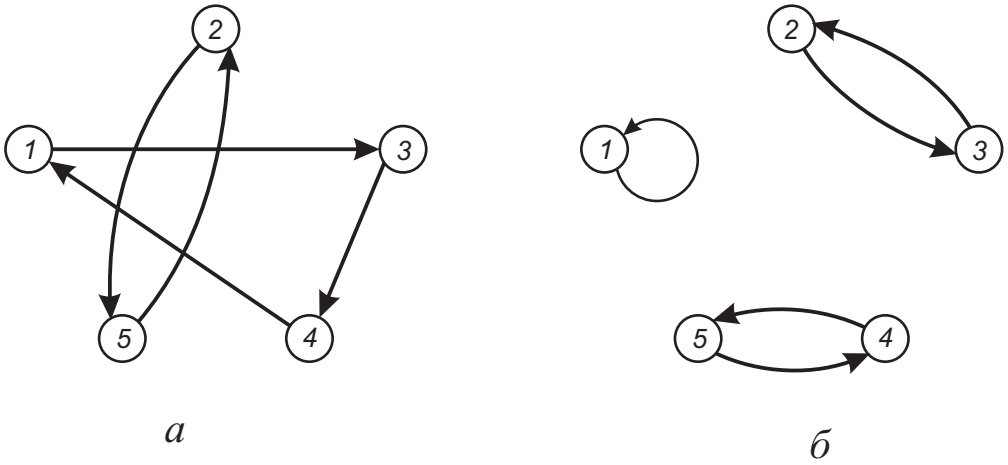


Рис. 3.1. Подстановки f и g из приведенных выше примеров и их циклы: подстановка ($а$) содержит цикл длины 3 и цикл длины 2, подстановка ($б$) содержит цикл длины 1 и два цикла длины 2

последовательность элементов $x_0, \ldots, x_k$ множества X такая, что

$$f(x_0) = x_1,\ f(x_1) = x_2,\ \ldots,\ f(x_{k-1}) = x_k,\ f(x_k) = x_0.$$

Подстановка на n-множестве называется *транспозицией*, если она соднржит один цикл длины 2 и $n-2$ цикла длины 1. Композиция какой-либо подстановки и транспозиции меняет местами два элемента этой подстановки, например:

$$f = \left|\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 5 & 3 & 1 & 2 & 4 \end{array}\right|, \quad t = \left|\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 1 & \underline{3} & \underline{2} & 4 & 5 \end{array}\right|, \quad f \circ t = \left|\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 5 & \underline{1} & \underline{3} & 2 & 4 \end{array}\right|.$$

Транспозиция, которая меняет местами соседние элементы i и $i+1$ подстановки, называется *стандартной*.

Не теряя общности, будем считать, что элементы в верхней строке таблицы всегда упорядочены по возрастанию. Тогда элементы в нижней строке представляют собой перестановку множества X.

Другими словами, существует взаимно-однозначное соответствие между подстановками на множестве X и перестановками его элементов.

Определение 3.11. Пара (x_i, x_j) элементов перестановки называется *инверсией*, если $i < j$ и $x_i > x_j$.

Например, перестановка $\langle 5, 3, 1, 2, 4\rangle$ имеет 6 инверсий.

Обозначим через $I(f)$ число инверсий в перестановке f.

Теорема 3.10. *Произвольную перестановку f можно представить в виде композиции $I(f)$ стандартных транспозиций.*

Доказательство. Обозначим через e тождественную подстановку:

$$e = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{vmatrix}.$$

Если $f = e$, то $I(f) = 0$. Пусть $f \neq e$ и пара (x_i, x_{i+1}) является инверсией. Тогда соответствующая стандартная транспозиция t_1 удаляет эту инверсию (и только ее): $I(f \circ t_1) = I(f) - 1$. Применив $I(f)$ стандартных транспозиций, получим подстановку, в которой отсутствуют инверсии, то есть тождественную подстановку

$$f \circ t_{I(f)} \circ t_{I(f)-1} \circ \ldots \circ t_1 = e. \tag{3.6}$$

Рассмотрим подстановку, обратную к композиции подстановок в левой части равенства (3.6)

$$T^{-1} = (t_{I(f)} \circ \ldots \circ t_1)^{-1} = t_1^{-1} \circ \ldots \circ t_{I(f)}^{-1}$$

и применим ее к обеим частям этого равенства

$$(f \circ t_{I(f)} \circ \ldots \circ t_1) \circ T^{-1} = e \circ T^{-1},$$

что равносильно

$$f = t_1^{-1} \circ \ldots \circ t_{I(f)}^{-1}.$$

■

Идея упорядочения элементов заданной перестановки путем последовательного исключения инверсий соседних элементов заложен в основу простейшего алгоритма сортировки, который называется «методом пузырька». Пары соседних элементов просматриваются и сравниваются между собой, и в случае обнаружения инверсии элементы пары меняются местами. Процесс повторяется до тех пор, пока в данной перестановке не будут исключены все инверсии.

Глава 4

Введение в теорию графов

4.1. Основные определения

Термин *граф* впервые ввел венгерский математик Денеш Кёниг в 1936 году, хотя первые работы по теории графов проводились еще Леонардом Эйлером в XVIII веке.

Определение 4.12. Пусть V – конечное непустое множество, E – подмножество множества всех неупорядоченных пар элементов множества V. Пара $G = (V, E)$ называется *неориентированным графом* или просто *графом*. Элементы множества V называются *вершинами* графа G, элементы множества E – его *ребрами*. Граф G с множеством вершин V и множеством ребер E обозначается $G(V, E)$.

Если $\{a, b\} \in E$, то вершины a и b называются *концами* ребра $\{a, b\}$. При этом ребро $\{a, b\}$ называют *инцидентным* вершинам a и b, а вершины a и b называются *смежными*. Два ребра называются *смежными*, если они инцидентны одной вершине. Множество всех вершин графа G, смежных вершине v, обозначается $\Gamma(v)$.

Вершины графа принято отображать точками на плоскости, а ребра – непрерывными линиями, соединяющими пары смежных вершин (рис. 4.1). Несмотря на то, что графы имеют удобное графическое

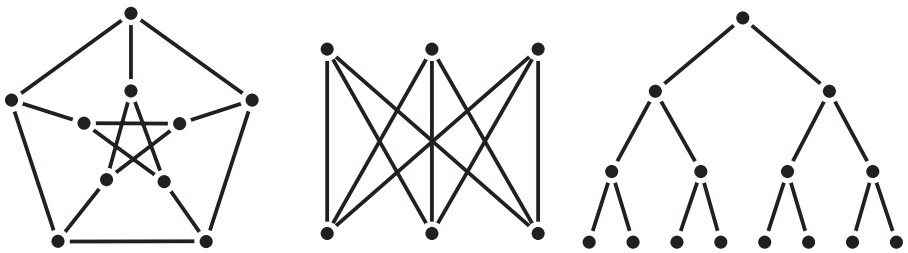


Рис. 4.1. Примеры графов: граф Петерсена, двудольный граф и дерево

представление, следует помнить, что граф это *комбинаторный* объект, определяемый в терминах абстрактных множеств[1].

Если в паре $G = (V, E)$ E является мультимножеством, то G называется *мультиграфом*. В мультиграфе допускается наличие более одного ребра между парой вершин (рис. 4.2).

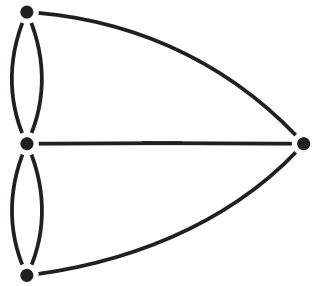


Рис. 4.2. Мультиграф

Степенью вершины v, обозначается $\deg(v)$, называется количество ребер, инцидентных вершине v. Если $\deg(v) = 0$, то вершина v называется *изолированной*, если $\deg(v) = 1$, то *висячей*.

Лемма 4.1 («о рукопожатиях»). *Сумма степеней вершин графа всегда четная и равна удвоенному числу его ребер.*

Действительно, поскольку каждое ребро имеет два конца, степень каждой вершины увеличивается на 1 за счет одного ребра. Таким образом, перечисляя все ребра, получим, что сумма степеней всех вершин равна удвоенному числу ребер графа:

$$\sum_{v \in G} \deg(v) = 2|E|.$$

Следствие 4.2. *В любом графе количество вершин нечетной степени четно.*

Доказательство. От противного: пусть в некотором графе число вершин с нечетной степенью нечетно. Тогда сумма степеней таких вершин также нечетна (сумма нечетного числа нечетных чисел всегда нечетна). С другой стороны сумма степеней вершин с четной степенью четна (как и сумма любого количество четных чисел). Если теперь просуммировать число степеней вершин с четной и нечетной степенью, то получится нечетное число, что противоречит лемме 4.1. ∎

Граф $G'(V', E')$ называется *подграфом* графа $G(V, E)$ (обозначается $G' \subseteq G$), если $V' \subseteq V$ и $E \subseteq E$, и *собственным подграфом*, если $V' \subset V$ и $E \subset E$ (обозначается $G' \subset G$). Пример подграфа изображен на рис. 4.3.

[1]Плоский граф, рассматриваемый в последующих разделах, сторого говоря, является не графом, а геометрической фигурой, которой соответствует некоторый планарный граф.

Путем (*маршрутом*) в графе называется чередующаяся последовательность вершин и ребер $v_0\, e_1\, v_1\, e_2 \ldots e_k\, v_k$, в которой любые соседние элементы инцидентны. Число k ребер в пути называется *длиной пути*. Если $v_0 = v_k$, то путь *замкнут*, иначе *открыт*. Если все ребра различны, то путь называется *цепью*, обозначается $\langle u, v\rangle$. Если все вершины различны (а значит, и ребра), то путь называется *простой цепью*.

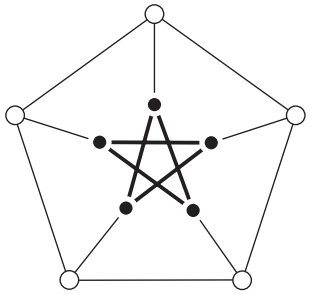


Рис. 4.3. Подграф графа Петерсена

Очевидно, что если между некоторыми вершинами существует цепь, то между ними существует также простая цепь. Замкнутая цепь называется *циклом*, замкнутая простая цепь – *простым циклом*. Граф без циклов называется *ациклическим*.

На рис. 4.4 последовательность вершин $\langle 1, 4, 1, 2, 3\rangle$ задает путь, но не цепь, последовательность $\langle 2, 4, 1, 2, 3\rangle$ задает цепь, которая не является простой, $\langle 4, 1, 2, 3\rangle$ есть простая цепь, а $\langle 2, 4, 1, 2\rangle$ образует цикл.

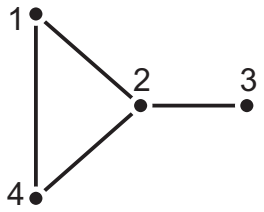


Рис. 4.4. Пути в графе

Расстоянием между вершинами u и v, обозначается $d(u, v)$, называется длина кратчайшей $\langle u, v\rangle$ – цепи. Максимальное расстояние между вершинами называется *диаметром* графа и обозначается $D(G)$. Диаметр графа на рис. 4.4 равен двум.

Граф, состоящий из одной вершины, называется *тривиальным*. Граф называется *полным*, если все его вершины смежны. Полный граф с n вершинами обозначается K_n.

4.2. Оценки числа графов

Ответим на вопрос: сколько существует различных графов с n вершинами? Заметим, что в самой формулировке вопроса присутствует неоднозначность. Действительно, какие графы следует считать различными, а какие одинаковыми?

Начнем с рассмотрения так называемых помеченных графов. Граф называется *помеченным*, если его множество вершин есть множество $[n] = \{1, \ldots, n\}$. Легко убедиться, что три помеченных графа, изображенные на рис. 4.5, различны, поскольку их множества ребер отличаются.

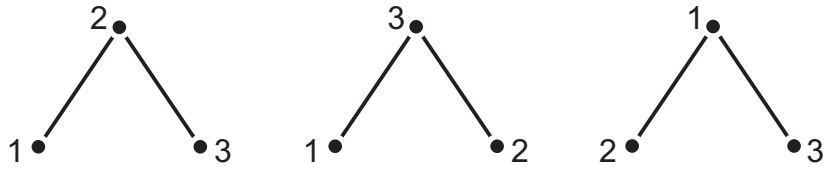


Рис. 4.5. Три различных помеченных графа с множествами ребер, соответственно, $\{\{1,2\},\{2,3\}\}$, $\{\{1,3\},\{2,3\}\}$ и $\{\{1,2\},\{1,3\}\}$

Теорема 4.11. *Число помеченных графов с n вершинами равно* $2^{\binom{n}{2}}$.

Действительно, по определению, число ребер в полном графе с n вершинами равно числу всевозможных пар вершин:

$$\binom{n}{2} = \frac{n(n-1)}{2}.$$

Тогда число всех графов с фиксированным множеством вершин равно числу всех подмножеств множества всевозможных пар этих вершин, то есть

$$2^{\binom{n}{2}}.$$

Однако не всегда графы следует различать: в ряде задач важны не вершины, а структура взаимосвязей между ними. Поэтому если один граф можно получить из другого лишь переименованием вершин, то такие графы считают одинаковыми, или изоморфными.

Определение 4.13. Графы $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$ называются *изоморфными*, если существует биекция $\varphi : V_1 \to V_2$, сохраняющая отношение смежности, то есть удовлетворяющая условию

$$\{u, v\} \in E_1 \Leftrightarrow \{\varphi(u), \varphi(v)\} \in E_2.$$

Биекция φ называется *изоморфизмом* графа G_1 на граф G_2. Если графы G_1 и G_2 изоморфны, то пишут $G_1 \cong G_2$. Два изоморфных графа изображены на рис. 4.6.

Для доказательства изоморфизма графов необходимо указать соответствующее биективное отображение одного графа на другой, например, присвоив их вершинам номера таким образом, чтобы множества ребер обоих графов состояли из одинаковых пар номеров. Для доказательства обратного достаточно найти в данных графах хотя бы одно несовпадающее свойство или числовую характеристику, например,

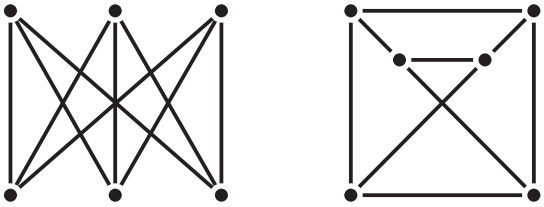


Рис. 4.6. Изоморфные графы

количество циклов определенной длины, количество вершин заданной степени, значение диаметра и т. д.

При необходимости подчеркнуть, что рассматриваемые графы различаются лишь с точностью до изоморфизма, используют термин *непомеченный* (или *абстрактный*) граф, под которым понимается не один граф с заданным множеством вершин и ребер, а все множество изоморфных ему графов. Отсюда следует, что на рис. 4.5 трижды изображен один и тот же непомеченный граф с тремя вершинами и двумя ребрами.

Для числа непомеченных (то есть попарно неизоморфных) графов справедлива следующая теорема.

Теорема 4.12 (Пойа). *Пусть $g(n)$ – число всех неизоморфных графов с n вершинами. Тогда*

$$g(n) \sim 2^{\binom{n}{2}}/n!$$

Символ $\sim$ означает *асимптотическое равенство*, т. е.

$$\lim_{n\to\infty} \frac{g(n)}{2^{\binom{n}{2}}/n!} = 1.$$

4.3. Способы задания графов

Определение 4.14. Рассмотрим граф $G = (V, E)$, где $V = \{v_1, \ldots, v_n\}$ и $E = \{e_1, \ldots, e_m\}$, и матрицу $B_{n,m}$, элементы которой равны

$$b_{i,j} = \begin{cases} 1, & \text{если вершина } v_i \text{ инцидентна ребру} e_j, \\ 0 & \text{в противном случае.} \end{cases}$$

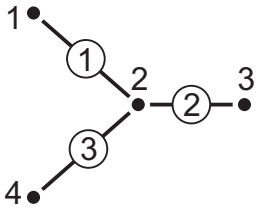
Матрица B называется *матрицей инциденции* графа G.

Определение 4.15. Рассмотрим граф $G = (V, E)$, где $V = \{v_1, \ldots, v_n\}$ и матрицу $A_{n,n}$, элементы которой равны

$$a_{i,j} = \begin{cases} 1, & \text{если } \{v_i, v_j\} \in E, \\ 0 & \text{в противном случае.} \end{cases}$$

Матрица B называется *матрицей смежности* графа G.

На рис. 4.7 изображен граф вместе с его матрицами смежности и инциденции. Легко заметить, что степень i-й вершины равна сумме элементов i-й строки каждой из матриц, а матрица смежности симметрична относительно главной диагонали (то есть $a_{ij} = a_{ji}$).



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Рис. 4.7. Матрицы смежности и инциденции

Матрицы смежности и инциденции позволяют вычислять количественные характеристики графа, выполнять обходы вершин и решать различные задачи.

4.4. Операции над графами

Рассмотрим следующие операции над графами:

1. *Дополнением* $\overline{G}$ графа $G = (V, E)$ называется граф, в котором две вершины смежны тогда и только тогда, когда они несмежны в G: $\overline{G} = (V, \overline{E})$.

2. *Объединением* графов $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$, обозначается $G_1 \cup G_2$, называется граф $G_3 = (V_1 \cup V_2, E_1 \cup E_2)$.

3. *Удаление* вершины u из графа G дает граф $G - u$, в котором присутствуют все вершины и ребра графа G, кроме вершины u и инцидентных ей ребер.

4. *Разбиение* ребра $\{u, v\}$ графа $G_1 = (V_1, E_1)$ дает граф $G_2 = (V_2, E_2)$, где $V_2 = V_1 \cup w$ и $E_2 = (E_1 \backslash \{u, v\}) \cup \{u, w\} \cup \{w, v\}$.

5. *Стягивание* ребра $\{u, v\}$ графа $G_1 = (V_1, E_1)$ дает граф $G_2 = (V_2, E_2)$, где $V_2 = V_1 \backslash v$ и $E_2 = E_1 \backslash \{u, v\} \cup \{\{u, w\} \ : \ w \in \Gamma(v)\}$.

6. *Стягивание подграфа* $G' = (V', E')$ графа $G_1 = (V_1, E_1)$ дает граф $G_2 = (V_2, E_2)$, где $V_2 = V_1 \backslash V' \cup v$, и $E_2 = E_1 \backslash E' \cup \{\{v, w\} \,|\, w \in \Gamma(V')\}$, где через $\Gamma(V') \subseteq V_1$ обозначено множество вершин, смежных вершинам из V' (рис. 4.8).

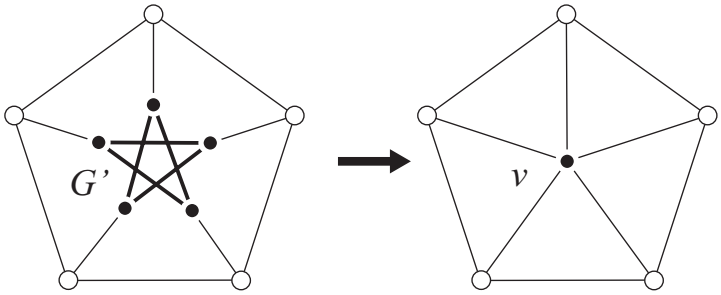


Рис. 4.8. Стягивание подграфа

4.5. Связность

Рассмотрим компьютерную сеть, в которой обмен информацией между двумя компьютерами осуществляется либо через соединяющую их линию связи (если она существует), либо через другие компьютеры и линии связи. Надежность, или *живучесть*, сети выражается в ее способности функционировать при выходе из строя части компьютеров и линий связи. Сеть считается тем более надежной, чем больше компьютеров или линий связи нужно вывести из строя, чтобы нарушить соединение между какой-либо парой компьютеров. Математической моделью такой сети является граф, в котором вершинам сопоставляются компьютеры, а ребрам – линии связи между ними. Свойством графа, характеризующим степень живучести сети, является связность.

Говорят, что две вершины в графе *связны*, если между ними существует соединяющий их путь. Граф, в котором все вершины связны, называется *связным*, в противном случае *несвязным*.

Непустой связный подграф G' графа G называется *компонентой связности* графа G, если для любого связного подграфа G'' графа G из $G' \subseteq G''$ следует $G' = G''$ (другими словами, G' – максимальный

связный подграф графа G)[2].

Число компонент связности графа G обозначается $k(G)$. Таким образом, граф является *связным*, если $k(G) = 1$.

Вершина v графа G называется *разделяющей*, если если граф $G - v$ имеет больше компонент связности, чем G, то есть $k(G) < k(G - v)$. Аналогично, ребро e называется *мостом*, если $k(G) < k(G-e)$. *Блоком* называется связный граф, не имеющий разделяющих вершин.

На рис. 4.9 изображен связный граф, в котором вершины u, v и w являются разделяющими. Ребро $\{v, w\}$ есть мост, удаление которого даст две компоненты связности, причем правая компонента является блоком.

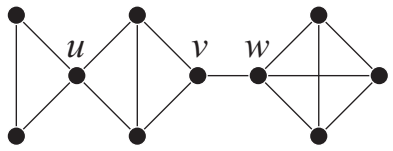


Рис. 4.9. Элементы связности графа

Теорема 4.13. *Пусть n, m и k – соответственно, число вершин, ребер и компонент связности графа. Тогда*

$$n - k \leq m < \frac{(n-k)(n-k+1)}{2}.$$

Доказательство. Докажем сначала, что $n - k \leq m$. Воспользуемся индукцией по n. Для $n = 1$ имеем: $k = 1$ и $m = 0$ – неравенство верно. Пусть неравенство верно для для всех графов с числом вершин меньшим n. Заметим, что в любом нетривиальном графе существуют по крайней мере две вершины, которые не являются разделяющими (например концы диаметра графа). Удалим из графа одну из таких вершин и рассмотрим полученный граф G', в котором $n' = n - 1$ вершин, k' компонент связности и m' ребер. Из индуктивного предположения следует, что $n' - k' \leq m'$. Возможны два варианта. Если удаленная вершина была изолированной в графе G, то $k' = k - 1$ и $m' = m$. Тогда $(n-1)-(k-1) \leq m$, то есть $n-k \leq m$. Если удаленная вершина не была изолированной, тогда $k' = k$ и $m' \leq m-1$ (удаленная вершина могла быть инцидентной одному или более ребрам). Значит $(n-1) - k \leq m' \leq m - 1$, то есть $n - k \leq m$. Таким образом, мы доказали, что для n вершин неравенство верно.

[2]В общем случае максимальный элемент частично упорядоченного множества $(X, \leq)$ определяется как такой элемент x, что $(\forall y \in X)(x \leq y) \Rightarrow (x = y)$.

Докажем теперь, что $m < (n-k)(n-k+1)/2$. Ответим на вопрос: какой граф с n вершинами и k компонентами связности имеет максимальное число ребер? Очевидно, что это граф, каждая компонента связности которого есть полный граф. Заметим также, что среди всех таких графов наибольшим числом ребер обладает граф, включающий $k-1$ тривиальную компоненту и одну нетривиальную – полный граф K_{n-k+1}. Действительно, пусть K_{n_1} и K_{n_2} – две компоненты связности графа G такие, что $1 < n_1 \leq n_2$. Тогда перенос одной вершины из K_{n_1} в K_{n_2} уменьшит число ребер в первой компоненте на $n_1 - 1$ и увеличит число ребер в другой компоненте на n_2, то есть в целом увеличит число ребер в графе G на $n_2 - n_1 + 1 > 0$.

Таким образом, максимальное число ребер графа с n вершинами и k компонентами связности равно числу ребер графа K_{n-k+1}, то есть $(n-k+1)(n-k)/2$. ∎

Следствие 4.3. *Если* $(n-1)(n-2)/2 < m$, *то граф связен.*

Действительно, неравенства

$$\frac{(n-1)(n-2)}{2} < m \leq \frac{(n-k+1)(n-k)}{2},$$

верны одновременно только при $k = 1$.

Меры связности графа

Для решения многих практических задач желательно уметь вычислять количественные характеристики графа, позволяющие оценить степень его связности.

Вершинной связностью графа G, обозначается $\varkappa(G)$, называется наименьшее число вершин, удаление которых дает несвязный или тривиальный граф. По определению, число $\varkappa(G) = 0$ соответствует несвязному графу, а $\varkappa(G) = n-1$ – полному графу K_n. Если $\varkappa(G) = 1$, то граф имеет разделяющую вершину. Граф G называется *k-связным*, если $\varkappa(G) = k$. Максимальный k-связный подграф графа называется его *k-компонентой*.

Аналогично, *реберной связностью* графа G, обозначается $\lambda(G)$, называется наименьшее число ребер, удаление которых дает несвязный или тривиальный граф. Для несвязного графа имеем $\lambda(G) = 0$, для полного $\lambda(K_n) = n-1$. Если $\lambda(G) = 1$, то граф G содержит мост.

Обозначим через $\delta(G)$ наименьшую степень вершин графа G.

Теорема 4.14. *Для любого графа G верны неравенства*

$$\varkappa(G) \leq \lambda(G) \leq \delta(G).$$

Доказательство. Докажем, что $\varkappa(G) \leq \lambda(G)$. Если $\lambda(G) = 0$, то граф несвязен и $\varkappa(G) = 0$. Если $\lambda(G) = 1$, то граф имеет мост. Значит либо граф G содержит разделяющую вершину, инцидентную этому мосту, либо $G = K_2$. В любом случае $\varkappa(G) = 1$. Наконец пусть $\lambda(G) \geq 2$. Тогда в графе G существует множество E', состоящее из $\lambda - 1$ ребер, удаление которых даст граф, содержащий мост $\{u, v\}$. Заметим, что удалить ребра E' можно путем удаления не более чем $\lambda - 1$ вершин, инцидентных ребрам E' и отличных от u и v. Если удаление этих вершин даст несвязный граф, то $\varkappa \leq \lambda - 1 < \lambda$. Иначе несвязный граф получается удалением одного из концов ребра $\{u, v\}$, то есть $\varkappa \leq \lambda$. ■

Теорема Менгера

Интуитивно понятно, что граф тем более связен, чем больше существует различных цепей, соединяющих одну вершину с другой, и тем менее связен, чем меньше нужно удалить промежуточных вершин, чтобы отделить одну вершину от другой. Теорема Менгера утверждает соотношение между числом таких цепей и числом таких разделяющих вершин.

Пусть $G(V, E)$ – связный граф, и u и v – две его несмежные вершины. Две $<u, v>$-цепи называются *вершинно непересекающимися*, если они не имеют общих вершин. Множество вершин S графа разделяет две вершины u и v, если u и v принадлежат разным компонентам связности графа $G - S$.

Теорема 4.15. *Наименьшее число вершин, разделяющее две несмежные вершины u и v, равно наибольшему числу вершинно непересекающихся $<u, v>$-цепей.*

4.6. Деревья

Деревья представляют собой простейший класс графов, который, однако, находит самое широкое применение при решении практических задач.

Определение 4.16. *Деревом* называется связный граф без циклов (рис. 4.10). Любой (не обязательно связный) граф без циклов называется *лесом* или *ациклическим*.

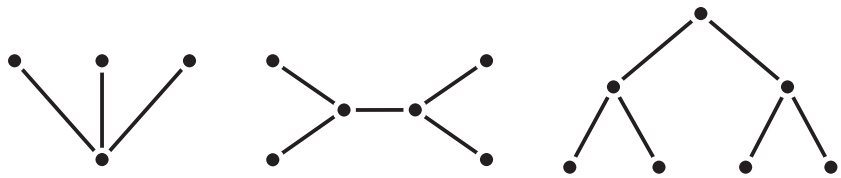


Рис. 4.10. Примеры деревьев

Существует другие эквивалентные варианты определения дерева, некоторые из них отражены в следующей теореме.

Теорема 4.16. *Пусть $G(V, E)$ – граф с n вершинами и m ребрами. Тогда следующие утверждения эквивалентны:*

1. *G – дерево.*
2. *G – связный граф, и $m = n - 1$.*
3. *G – ациклический граф, и $m = n - 1$.*
4. *Любые две несмежные вершины графа G соединяет единственная простая цепь.*
5. *G – такой ациклический граф, что если соединить ребром любую пару его несмежных вершин, то полученный граф будет иметь ровно один цикл.*

Прежде чем приступить к доказательству теоремы 4.16, докажем следующее простое утверждение.

Лемма 4.2. *Пусть $G(V, E)$ – связный граф и $e \in E$. Тогда*

1. *если ребро e принадлежит какому-либо циклу графа G, то граф $G - e$ связен;*
2. *если ребро e не принадлежит ни одному циклу, то граф $G - e$ имеет ровно две компоненты связности.*

Доказательство. (1) Действительно, пусть ребро $e = \{u, v\}$ принадлежит циклу C графа G, и две вершины x и y графа G соединены цепью $< x, y >$, содержащей участок $< u, e, v >$. Очевидно, в графе $G - e$ вершины x и y также являются связными, поскольку они соединены цепью $C - e$.

(2) Пусть ребро $e = \{u, v\}$ не входит ни в один цикл графа. Тогда, очевидно, вершины u и v принадлежат разным компонентам

графа $G - e$ (тривиально доказывается от противного). Обозначим эти компоненты, соответственно, G_u и G_v. Для любой вершины $x \neq u$ в G существует цепь из x в u. Если эта цепь не содержит ребро e, то $x \in G_u$. Если же она содержит ребро e, то она проходит через v, и при этом не существует другой цепи, соединяющей x и u, не содержащей e (иначе получим цикл, содержащий e). Значит $x \in G_v$. Таким образом, граф $G - e$ имеет ровно две компоненты связности. ∎

Теперь мы можем приступить к доказательству теоремы 4.16.

Доказательство. Чтобы установить эквивалентность всех утверждений, докажем, что каждое из них следует из предыдущего, а из последнего следует первое.

(1) $\Rightarrow$ (2). Пусть граф G связен и не содержит циклов. Докажем, что $m = n - 1$. Воспользуемся индукцией по n. При $n = 1$ доказательство тривиально. Пусть $n > 1$ и $e \in E$. Из леммы 4.2 следует, что граф $G - e$ имеет ровно две компоненты связности T_1 и T_2, каждая из которых есть дерево, то есть, по индуктивному предположению, для них выполняется равенство $m_i = n_i - 1,\ i = 1, 2$.

Тогда имеем

$$\begin{aligned} m &= m_1 + m_2 + 1 = (n_1 - 1) + (n_2 - 1) + 1 = \\ &= (n_1 + n_2) - 1 = n - 1. \end{aligned}$$

(2) $\Rightarrow$ (3). Граф G связен и $m = n - 1$. Докажем, что в нем отсутствуют циклы. От противного: предположим, что цикл существует, и e – ребро этого цикла. Но тогда, по лемме 4.2, граф $G - e$ также связен и имеет $n - 2$ ребра, что противоречит теореме 4.13, согласно которой $n - k \leq m$ (в нашем случае $k = 1$). Значит, граф G не имеет циклов.

(3) $\Rightarrow$ (4). Пусть k – число компонент связности графа G. Каждая компонента T_i есть дерево, в котором $m_i = n_i - 1$ ребер. Тогда

$$\begin{aligned} n - 1 = m &= m_1 + \ldots + m_k = (n_1 - 1) + \ldots + (n_k - 1) = \\ &= (n_1 + \ldots + n_k) - k = n - k, \end{aligned}$$

откуда следует, что $k = 1$, то есть граф G связен, и любые две несовпадающие вершины соединены в нем простой цепью. Если бы таких цепей было две или более, то они образовывали бы цикл, что противоречит условию (3). Значит любые две вершины в G соединены единственной простой цепью.

(4) $\Rightarrow$ (5). Пара несовпадающих вершин, принадлежащих одному циклу, соединена по крайней мере двумя простыми цепями. Значит,

граф G – ациклический. Пусть u и v – две его несмежные вершины. Добавим к графу G ребро $e = \{u, v\}$. По условию (4) в G существует *единственная* простая $< u, v >$-цепь, которая в $G + e$ дополняется до *единственного* простого цикла.

(5) $\Rightarrow$ (1). Достаточно показать, что граф G, удовлетворяющий условию (5), связен. Если бы некоторые вершины u и v принадлежали разным компонентам связности графа G, то граф $G+\{u, v\}$ не имел бы циклов, но из условия (5) следует, что такой цикл существует. Значит, граф G связен и является деревом. ∎

Следствие 4.4. *В любом нетривиальном дереве существует не менее двух висячих вершин.*

Доказательство. Из леммы «о рукопожатиях» имеем

$$\sum_{i=1}^{n} \deg v_i = 2(n-1) = 2n - 2,$$

причем все $\deg v_i \geq 1$. Значит, хотя бы две вершины должны иметь степень, равную 1. ∎

Определение 4.17. *Остовным подграфом* графа G называется подграф, содержащий все вершины графа G. Если каждая компонента связности остовного подграфа является деревом, то он называется *остовом* графа G.

Ясно, что в любом графе существует остов: удаление нужных ребер в конце концов приведет к разрушению всех циклов.

Следствие 4.5. *Число ребер произвольного графа G, которое необходимо удалить для получения остова, не зависит от последовательности их удаления и равно $m - n + k$, где m, n и k обозначают, соответственно, число ребер, вершин и компонент связности графа G.*

Доказательство. Если G_1 является одной из компонент связности графа G, то число ребер, которые необходимо удалить для превращения ее в остовное дерево, равно $m_1 - (n_1 - 1) = m_1 - n_1 + 1$. Суммируя по всем компонентам связности, получим $m - n + k$. ∎

Определение 4.18. Число $\nu(G) = m - n + k$ называется *цикломатическим* числом графа G.

Если граф G является лесом, то $\nu(G) = 0$, если он имеет единственный цикл, то $\nu(G) = 1$.

4.7. Эйлеровы графы

В 1736 году Леонарду Эйлеру было предложено рассмотреть следующую задачу: можно ли обойти все семь мостов в городе Кёнигсберге (современный Калининград) и вернуться в исходную точку, побывав на каждом мосту по одному разу (рис. 4.11)?

Рис. 4.11. Истрорический центр современного Калининграда (два из семи Кёнигсбергских мостов, рассмотренных в задаче, ныне не существуют)

Комбинаторной моделью этой задачи является мультиграф, изображенный на рис. 4.12. Эйлер сформулировал более общую задачу: при каких условиях связный граф содержит цикл, проходящий через каждое ребро? Напомним, что, по определению, цикл в графе содержит ребра только по одному разу (вершины могут повторяться).

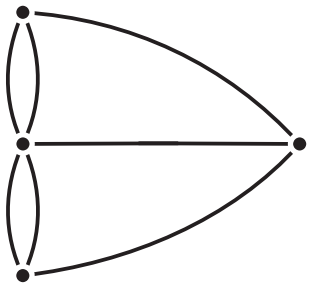


Рис. 4.12. Задача о Кёнигсбергских мостах

Цикл в графе называется *эйлеровым*, если он содержит все ребра графа. Связный граф, в котором содержится эйлеров цикл, называется *эйлеровым графом*.

Эйлер установил, что критерием существования эйлерова цикла в связном графе является отсутствие в нем вершин нечетной степени. Сформулируем и докажем более общее утверждение.

Теорема 4.17. *Пусть G – связный и нетривиальный граф. Тогда следующие утверждения эквивалентны.*

1. *G – эйлеров граф.*
2. *Каждая вершина графа G имеет четную степень.*
3. *Множество ребер графа G можно разбить на простые циклы.*

Доказательство. (1) $\Rightarrow$ (2). Пусть Z – эйлеров цикл в графе G. Двигаясь по Z, будем суммировать степени вершин, изначально полагая их нулевыми. Прохождение каждой вершины увеличивает ее степень на 2. Проход по всем ребрам означает также проход по всем вершинам. Значит степень каждой вершины – четная.

(2) $\Rightarrow$ (3). $G(V, E)$ – связный и нетривиальный граф с четными степенями вершин, стало быть, степень каждой его вершины не меньше двух. Это означает, что G не имеет висячих вершин и, согласно следствию 4.4, не является деревом. Значит, в G имеется хотя бы один цикл. Обозначим множество ребер этого цикла через Z_1. Тогда $G - Z_1$ есть остовный подграф, в котором степени всех вершин также четные. Если исключить из него изолированные вершины, то получится граф, который удовлетворяет условию (2), следовательно, он тоже имеет простой цикл Z_2. Действуя подобным образом, будем исключать из графа циклы до тех пор пока он не окажется пуст. Таким образом, мы показали, что множество ребер графа G есть объединение непересекающихся простых циклов.

(3) $\Rightarrow$ (1). Пусть $G = (V, E)$ где E можно представить в виде объединения непересекающихся простых циклов Z_i. Рассмотрим произвольный цикл Z_1. Если $Z_1 = E$, то утверждение доказано. Если нет, то существует цикл Z_2 и вершина v_1, принадлежащая обоим циклам, поскольку G – связный граф. Маршрут $Z_1 \cup Z_2$ является циклом и содержит все свои ребра по одному разу. Если $Z_1 \cup Z_2 = E$, то утверждение доказано. Если нет, то продолжим строить эйлеров цикл, пока он не будет содержать все ребра графа G. ∎

Эйлеровы графы достаточно редки. В виде формального утверждения этот факт звучит несколько забавно. Обозначим через $\mathcal{G}(n)$ множество помеченных графоа с n вершинами, а через $\mathcal{E}(n)$ – множество всех эйлеровых графов.

Теорема 4.18. *Эйлеровых графов почти нет, то есть*

$$\lim_{n\to\infty} \frac{|\mathcal{E}(n)|}{|\mathcal{G}(n)|} = 0.$$

Доказательство. Пусть $\mathcal{G}_2(n)$ – множество всех графов из $\mathcal{G}(n)$, степени вершин которых четные. Очевидно, что все графы из $\mathcal{G}_2(n)$ можно получить, добавив к некоторому графу из $\mathcal{G}(n-1)$ новую вершину и соединив ее со всеми вершинами нечетной степени (заменим, что число таких вершин четно).

Принимая во внимание оценку

$$|\mathcal{G}(n-1)| = 2^{\binom{n-1}{2}}$$

и тот факт, что не все графы в $\mathcal{G}(n-1)$ имеют вершины с нечетными степенями, получаем оценку

$$|\mathcal{G}_2(n)| < 2^{\binom{n-1}{2}}.$$

Согласно теореме 4.17, $\mathcal{E}(n) \subset \mathcal{G}_2(n)$ (классы не совпадают, так как $\mathcal{G}_2(n)$ содержит также несвязные графы), поэтому

$$\frac{|\mathcal{E}(n)|}{|\mathcal{G}(n)|} < \frac{|\mathcal{G}_2(n)|}{|\mathcal{G}(n)|} < \frac{2^{\binom{n-1}{2}}}{2^{\binom{n}{2}}} = 2^{\binom{n-1}{2}-\binom{n}{2}}.$$

Поскольку $\binom{n}{2} - \binom{n-1}{2} = n-1$, получаем

$$\frac{|\mathcal{E}(n)|}{|\mathcal{G}(n)|} < \frac{1}{2^{n-1}} \underset{n\to\infty}{\longrightarrow} 0.$$

■

4.8. Гамильтоновы графы

В 1859 году ирландский математик Уильям Гамильтон предложил игру «Кругосветное путешествие», в которой каждой из 20 вершин додекаэдра присваивается название одного из крупнейших городов мира. Требуется, передвигаясь из города в город по ребрам додекаэдра, посетить все города по одному разу и вернуться в исходный город (рис. 4.13).

Граф называется *гамильтоновым*, если в нем содержится простой цикл, содержащий каждую вершину этого графа. Такой цикл в графе называется *гамильтоновым*.

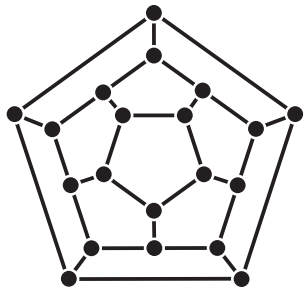


Рис. 4.13. Додекаэдр

Не все графы гамильтоновы – хотя бы потому, что такой граф должен быть двусвязным. Но даже не все двусвязные графы гамильтоновы.

Граф, содержащий две вершины степени 3, соединенные тремя попарно непересекающимися простыми цепями длины не менее 2, называется θ-*графом* (рис. 4.14). В следующей теореме сформулировано достаточное условие гамильтоновости двусвязных графов.

Теорема 4.19. *Каждый негамильтонов двусвязный граф содержит* θ-*подграф.*

Интуитивно понятно, что у графа больше шансов быть гамильтоновым, если он содержит много ребер и они равномерно распределены. Подтверждением этому является следующие достаточные условия гамильтоновости графов.

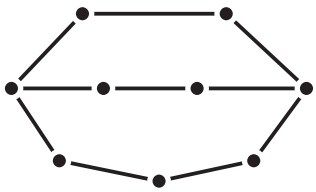


Рис. 4.14. θ-граф

Теорема 4.20 (Оре, 1960). *Пусть* G *– граф с числом вершин* $n \geq 3$*. Тогда если для любой пары* u *и* v *несмежных вершин графа* G *выполняется неравенство* $\deg(u) + \deg(v) \geq n$*, то граф* G *– гамильтонов.*

Заметим, что это лишь достаточное условие существования гамильтонова цикла в графе. В частности, для гамильтонова графа из игры в города (рис. 4.13) это условие не выполняется.

Следствие 4.6 (Теорема Дирака, 1952). *Если для любой вершины* v *графа* G *с числом вершин* $n \geq 3$ *выполняется* $\deg(v) \geq n/2$*, то граф* G *– гамильтонов.*

В отличие от эйлеровых графов, гамильтоновых графов «много». Обозначим через $\mathcal{H}(n)$ число всех гамильтоновых графов с n вершинами. Справедлива следующая теорема.

Теорема 4.21. *Почти все графы гамильтоновы, то есть*

$$\lim_{n \to \infty} \frac{\mathcal{H}(n)}{\mathcal{G}(n)} = 1.$$

4.9. Двудольные графы

Граф $G(V, E)$ называется *двудольным*, если множество его вершин V можно разбить на два непересекающихся подмножества V_1 и V_2, таких что концы любого ребра $e \in E$ принадлежат разным подмножествам. Подмножества V_1 и V_2 называются *долями* графа G, а сам двудольный граф обозначается $G(V_1, V_2, E)$.

Таким образом, в двудольном графе ребра могут существовать только между вершинами из разных долей.

Двудольный граф $G(V_1, V_2, E)$ называется *полным двудольным* графом, обозначается $K_{m,n}$, если $|V_1| = n$, $|V_2| = m$ и для любой пары вершин $v_1 \in V_1$ и $v_2 \in V_2$ в графе G существует ребро $\{v_1, v_2\}$.

Примеры двудольных графов изображены на рис. 4.15.

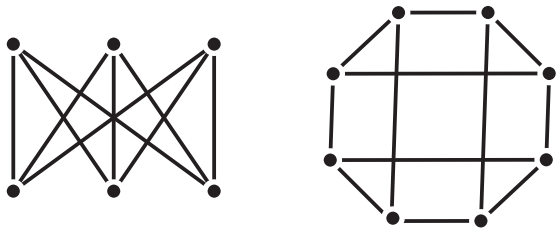


Рис. 4.15. Двудольные графы

Заметим, что в отличие от графа $K_{3,3}$, который изображен слева, изображение правого графа не позволяет сразу увидеть его доли и идентифицировать сам граф как двудольный. Тем не менее, существует простой критерий двудольности графа, сформулированный в следующей теореме.

Теорема 4.22 (Кёниг, 1936). *Нетривиальный граф является двудольным тогда и только тогда, когда он не содержит простых циклов нечётной длины.*

Доказательство. Необходимость очевидна. Действительно, если цикл в графе имеет вид $v_1, v_2, \ldots, v_{2k+1}, v_1$, то вершины v_{2k+1} и v_1 будут принадлежать одной доле, что противоречит определению двудольного графа.

Достаточность. Не ограничивая общности, будем считать, что граф связный, поскольку каждую компоненту связности можно рассматривать отдельно. Разбиение на доли произведем следующим образом. Выберем произвольную вершину $u \in V$ и отнесем ее к доле V_1. Любую другую вершину $v \in V$ отнесем к доле V_2, если расстояние $d(u, v)$

нечетно, и к V_1, если оно четно. Докажем теперь, что вершины одной доли несмежны.

От противного: пусть вершины одной доли v и w соединены ребром. Рассмотрим кратчайшие простые цепи $\langle u, v\rangle$ и $\langle u, w\rangle$. Пусть u' – наиболее удаленная от u вершина, принадлежащая обеим цепям (возможно, $u' = u$). Тогда имеем:

$$d(u, v) + d(u, w) = d(u, u') + d(u', v) + d(u, u') + d(u', w),$$

или

$$d(u', v) + d(u', w) = d(u, v) + d(u, w) - 2d(u, u').$$

Если обе вершины v и w принадлежат доле V_1, то оба расстояния $d(u, v)$ и $d(u, w)$ четны по построению. Если эти вершины принадлежат доле V_2, то оба расстояния нечетны. В любом случае их сумма четна. Но тогда сумма $d(u', v) + d(u', w)$ – четное число, и если между вершинами v и w существует ребро, то граф содержит цикл нечетной длины, что противоречит условию . ■

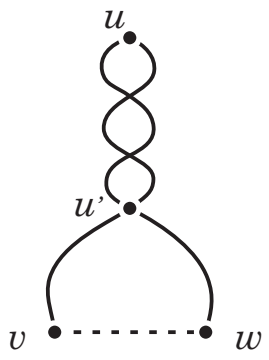


Рис. 4.16. К доказательству теоремы 4.22

4.10. Плоские и планарные графы

Рассмотрим следующую задачу о трёх колодцах. Есть три дома и три колодца. Можно ли так проложить дорожки между домами и колодцами, чтобы от каждого дома к каждому колодцу вела одна дорожка, и никакие две из них не пересекались?

Эту задачу можно сформулировать в терминах графов: можно ли нарисовать на плоскости полный двудольный граф $K_{3,3}$ (рис. 4.15) таким образом, чтобы его ребра пересекались только в вершинах?

Определение 4.19. *Плоским графом* называется граф, вершинам которого можно поставить в соответствие точки на плоскости, а ребрам – непересекающиеся непрерывные линии, соединяющие смежные вершины. Граф, изоморфный некоторому плоскому графу, называется *планарным*. О планарных графах говорят, что они *имеют плоскую укладку*.

На рис. 4.17 изображены плоский граф и изоморфный ему планарный граф. Ниже мы дадим ответ на вопрос, как определить, является

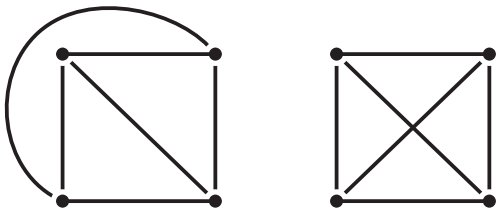


Рис. 4.17. Плоский граф и изоморфный ему планарный граф

граф планарным или нет. Если граф содержит мало вершин и ребер, то самый простой способ состоит в том, чтобы попытаться уложить его на плоскости, то есть найти изоморфный ему плоский граф. Но что делать, если уложить граф не получается?

Область плоскости, ограниченная ребрами плоского графа и не содержащая внутри себя вершин и ребер, называется *гранью*. Внешняя относительно всех граней часть плоскости также является гранью. Обозначим через n, m и r, соответственно, число вершин, ребер и граней плоского графа G.

Теорема 4.23 (Формула Эйлера). *Для связного планарного графа справедливо следующее соотношение*

$$n - m + r = 2.$$

Доказательство. По индукции: для $n = 1$ имеем: $m = 0$, $r = 1$ – равенство верно. Для $m = 1$ имеем: $n = 2$, $r = 1$ – тоже верно.

Пусть равенство выполняется для всех планарных графов с числом ребер от 1 до $m \geq 1$. Тогда возможно одно из двух: граф G содержит циклы или не содержит. Если не содержит, то G - дерево и имеет висячую вершину v. Удалим ее. Тогда для графа $G' = G - v$ имеем:

$$n' = n - 1,\ m' = m - 1,\ r' = r,$$

то есть

$$n - m + r = (n' + 1) - (m' + 1) + r' = n' - m' + r' = 2.$$

Пусть граф G содержит цикл. Удалим из этого цикла ребро e. Тогда для графа $G' = G - e$ имеем:

$$n' = n,\ m' = m - 1,\ r' = r - 1,$$

то есть

$$n - m + r = n' - (m' + 1) + (r' + 1) = 2.$$

■

Следствие 4.7. *Если G - связный планарный граф с числом вершин $n > 3$, то $m \leq 3n - 6$.*

Доказательство. Обозначим через S_r сумму ребер всех граней графа G. Поскольку каждая грань инцидентна не менее чем трем ребрам, а каждое ребро инцидентно не более чем двум граням, имеем

$$3r \leq S_r \leq 2m, \text{ то есть } r \leq \frac{2}{3}m.$$

Тогда

$$2 = n - m + r \leq n - m + \frac{2}{3}m, \text{ откуда } m \leq 3n - 6.$$

■

Формула Эйлера позволяет делать выводы о непланарности некоторых графов.

Следствие 4.8. *Графы K_5 и $K_{3,3}$ непланарны.*

Доказательство. Докажем, что граф K_5 непланарный. Имеем $n = 5$, $m = 10$. От противного: предположим, что он планарный. Тогда должно выполняться равенство из следствия 4.7, то есть

$$10 = m \leq 3n - 6 = 15 - 6 = 9.$$

Получили противоречие, значит граф K_5 непланарный.

Для графа $K_{3,3}$ имеем $n = 6$, $m = 9$. Предположим, что $K_{3,3}$ планарный. Тогда по формуле Эйлера получаем, что

$$2 = 6 - 9 + r, \text{ то есть } r = 5.$$

Граф двудольный, значит в нем отсутствуют грани, ограниченные тремя ребрами. Если просуммировать ребра всех его граней, получим неравенство $4r \leq 2m$ (см. доказательство следствия 4.7). То есть $2r \leq m$, откуда получаем $2 \times 5 \leq 9$ – противоречие. ■

Определение 4.20. Граф G' называется *производным* от графа G, если он получается из G в результате применения одной или нескольких операций разбиения ребер. Графы G и G' называются *гомеоморфными*, если существует граф G'' такой, что G и G' являются производными от G'' (рис. 4.18).

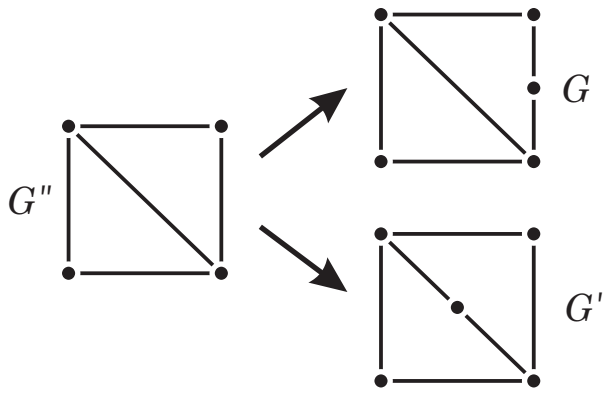


Рис. 4.18. Графы G и G' гомеоморфны, поскольку оба являются производными от графа G''

В следующей теореме, которую мы приводим здесь без доказательства[3], сформулировано необходимое и достаточное условие планарности графов.

Теорема 4.24 (Куратовский (1930), Понтрягин(1927)). *Граф планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных графам K_5 и $K_{3,3}$.*

4.11. Раскраска графов

Раскраской графа $G(V, E)$ называется функция $f : V \to \mathbb{N}$ такая, что $\{u, v\} \in E \Rightarrow f(u) \neq f(v)$. Другими словами, под раскраской понимается такое приписывание цветов (натуральных чисел) вершинам графа, что никакие две смежные вершины не получают одинаковый цвет. Если при этом использованы k цветов, то раскраска называется *k-раскраской*. Наименьшее возможное число в раскраске называется *хроматическим числом* графа и обозначается $\chi(G)$.

Ясно, что если существует k-раскраска графа $G(V, E)$, то

$$\chi(G) \leq k \leq |V|.$$

Обозначим через C_n граф, представляющий собой простой цикл длины n. Тогда относительно хроматического числа справедливы следующие соотношения:

$$\chi(K_n) = n,\ \chi(\overline{K}_n) = 1,\ \chi(K_{m,n}) = 2,\ \chi(C_{2n}) = 2,\ \chi(C_{2n+1}) = 3.$$

[3]Необходимость вытекает непосредственно из следствия 4.8.

В задачах о раскраске графов особое место занимают планарные графы. Рассмотрим так называемую *проблему четырёх красок*, сформулированную в 1852 году южноафриканским математиком Фрэнсисом Гутри: необходимо раскрасить политическую карту таким образом, чтобы при раскраске использовались не более четырех красок и любые две граничащие страны были раскрашены в разные цвета.

Моделью политической карты является плоский граф, в котором вершины соответствуют странам. Две вершины в этом графе соединены ребром тогда и только тогда, когда соответствующие им страны имеют общую границу. Вопрос ставится следующим образом: можно ли любой планарный граф раскрасить в четыре цвета?

Легко привести пример, когда трех красок может не хватить (например, в графе, изображенном на рис. 4.19). Покажем, что для раскраски любого планарного графа достаточно пяти красок. Сначала докажем простую лемму.

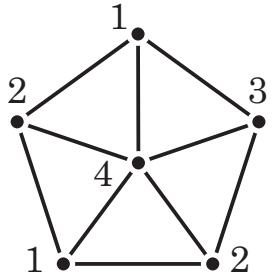


Рис. 4.19. Планарный граф с хроматическим числом 4

Лемма 4.3. *В любом планарном графе существует вершина, степень которой не больше пяти.*

Доказательство. Действительно, пусть степень каждой вершины больше или равна шести. Тогда из леммы о рукопожатиях и следствия 4.7 следуют соотношения

$$6n \leq \sum_{v \in V} \deg(v) \leq 2m \leq 2(3n-6) = 6n - 12.$$

Получили противоречие. ∎

Теорема 4.25. *Для раскраски любого планарного графа достаточно пяти красок.*

Доказательство. Будем рассматривать только связные графы, так как хроматическое число несвязного графа равно максимальному хроматическому его компонент связности.

По индукции: для $n \leq 5$ имеем $\chi(G) \leq n = 5$. Пусть теорема верна для всех графов с числом вершин $n \geq 5$. Покажем, что она верна для графов с $n+1$ вершиной. Из леммы 4.3 следует, что граф G содержит такую вершину v_0, что $\deg(v_0) \leq 5$. Рассмотрим два случая.

1. $|\Gamma(v_0)| \leq 4$. По индуктивному предположению, граф $G - v_0$ можно раскрасить в пять цветов. Зафиксируем одну из таких раскрасок. Тогда в графе G вершине v_0 присвоим цвет, который не использовался при раскраске вершин $\Gamma(v_0)$.

2. $|\Gamma(v_0)| = 5$. Заметим, что в множестве $\Gamma(v_0)$ существуют две несмежные вершины v_1 и v_2, иначе подграф, образованный вершинами из $\Gamma(v_0)$, представлял бы собой K_5, что противоречит планарности графа G. Рассмотрим граф G', образованный из графа $G - v_0$ слиянием вершин v_1 и v_2 в одну вершину v (рис. 4.20). Граф G' плоский и, по индуктивному предположению, может

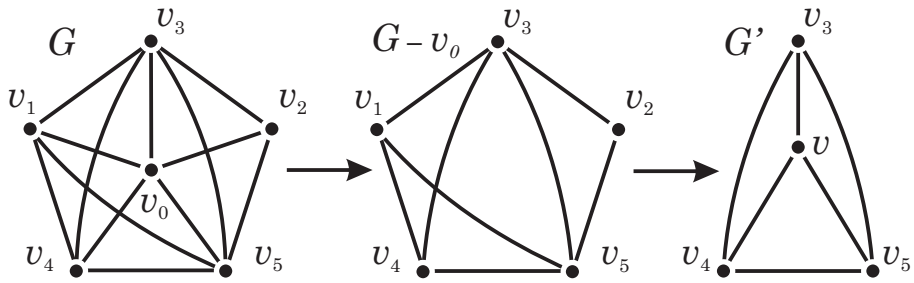


Рис. 4.20. В графе G удаляется вершина v_0, затем две несмежные вершины v_1 и v_2 стягиваются в одну вершину v

быть раскрашен пятью цветами. Зафиксируем одну из таких раскрасок. Затем в графе G окрасим вершины v_1 и v_2 в цвет вершины v, а остальным трем вершинам оставим цвет, который они имеют в графе G'. Затем окрасим вершину v_0 в цвет, не использованный при раскраске вершин из $\Gamma(v_0)$. ■

Задачу о четырех красках удалось решить неожиданным способом. В 1969 году немецкий математик Генрих Хееш свел задачу четырех красок к исследованию большого, но конечного множества графов. В 1976 году Кеннет Аппель и Вольфганг Хакен из Иллинойского университета раскрасили все графы этого множества, используя при этом не более четырёх цветов, с помощью мощного по тем временам компьютера, доказав тем самым, что любой планарный граф может быть раскрашен в четыре цвета. Это был первый случай в истории, когда нетривиальная математическая теорема была доказана с помощью компьютера.

4.12. Ориентированные графы

При решении многих задач приходится рассматривать графы, ребра которых ориентированы, то есть имеют начало и конец. С такими графами мы сталкивались в предыдущих разделах, в частности, когда рассматривали графы подстановок и диаграммы Хассе – графы частично-упорядоченных множеств.

Определение 4.21. Пусть V – конечное непустое множество и A – отношение на этом множестве. Пара (V, A) называется *ориентированным графом* (или *орграфом*) с множеством вершин V и множеством дуг A.

Таким образом, дуга орграфа это *упорядоченная* пара его вершин. Дуга из вершины u в вершину v обозначается (u, v). Вершина u называется *началом* дуги, а v – ее *концом*. Направления дуг орграфа принято обозначать стрелками (рис. 4.21).

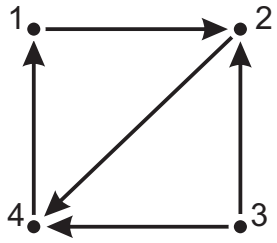


Рис. 4.21. Ориентированный граф

Понятия смежности и инцидентности определяются так же, как для неориентированных графов.

Множество концов всех дуг, исходящих из вершины u, обозначается $\mathbf{\Gamma}^+(u)$, а множество начал всех дуг, входящих в u, обозначается $\mathbf{\Gamma}^-(u)$. Число $\deg^+ u = |\mathbf{\Gamma}^+(u)|$ называется *полустепенью исхода* вершины u. Аналогично, число $\deg^- u = |\mathbf{\Gamma}^-(u)|$ называется *полустепенью захода*.

Следующая теорема является аналогом леммы о рукопожатиях для неориентированных графов (доказывается аналогичным образом).

Теорема 4.26. *Сумма полустепеней исхода всех вершин орграфа $G(V, A)$ равна сумме полустепеней захода и равна числу его дуг:*

$$\sum_{u \in G} \deg^+ u = \sum_{u \in G} \deg^- u = |A|.$$

Маршрутом в орграфе G называется чередующаяся последовательность вершин и дуг $v_0, a_1, \ldots, a_k, v_k$, такая, что $a_i = (v_{i-1}, v_i)$. Число дуг в маршруте называется его *длиной*. Маршрут называется *цепью*, если все входящие в него дуги различны. Если различны также все входящие в него вершины, то маршрут называется *путём*. Маршрут называется *циклическим*, если совпадают его первая и последняя вершины. Циклический путь называется *контуром*.

На рис. 4.21 последовательность вершин $\langle 3, 4, 1, 2, 4, 1\rangle$ определяет маршрут, последовательность $\langle 3, 4, 1, 2\rangle$ задает путь, а $\langle 1, 2, 4, 1\rangle$ – путь.

Орграф называется *турниром*, если он получен из полного графа введением ориентации его ребер (рис. 4.22). Такое название происходит из модели спортивного состязания, в котором соперники соревнуются по принципу «каждый с каждым». Дуга проводится от победителя к побежденному. В такой интерпретации соответствующий орграф определяет результаты встреч.

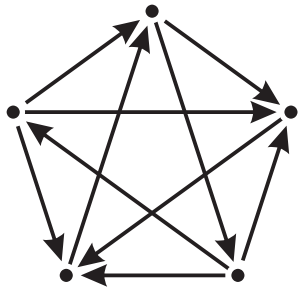


Рис. 4.22. Турнир

Связность в орграфах

Говорят, что две вершины u и v орграфа $G(V, A)$ *сильносвязны*, если в G существует путь из u в v и из v в u. Если в G существует хотя бы один путь – из u в v или из v в u, то вершины u и v называются *одностороннесвязными*. Если эти вершины связны в графе, полученном из G отменой ориентации ребер, то они называются *слабосвязными*.

Если все вершины в орграфе G сильно- (односторонне-, слабо-) связны, то G называется *сильно-* (*односторонне-*, *слабо-*) *связным*. Очевидно, что сильная связность орграфа означает одностороннюю связность, которая, в свою очередь, влечет слабую (обратное неверно). Примеры сильносвязного, одностороннесвязного и слабосвязного орграфа изображены на рис. 4.23.

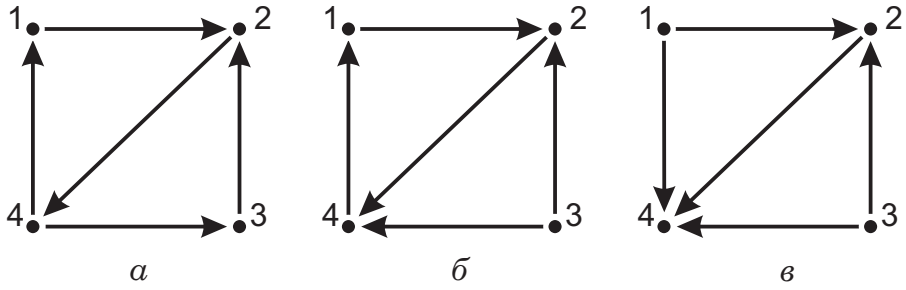


Рис. 4.23. Сильносвязный (*а*), одностороннесвязный (*б*) и слабосвязный (*в*) орграфы

Компонентой сильной связности (*сильной компонентой*) орграфа G называется его максимальный сильный подграф. Очевидно, любая вершина орграфа может принадлежать только одной сильной компо-

ненте. При этом изолированная вершина сама является сильной компонентой.

Конденсацией G^* орграфа G называется называется орграф, полученный из G стягиванием в одну вершину каждой его сильной компоненты. Пример конденсации орграфа изображен на рис. 4.24.

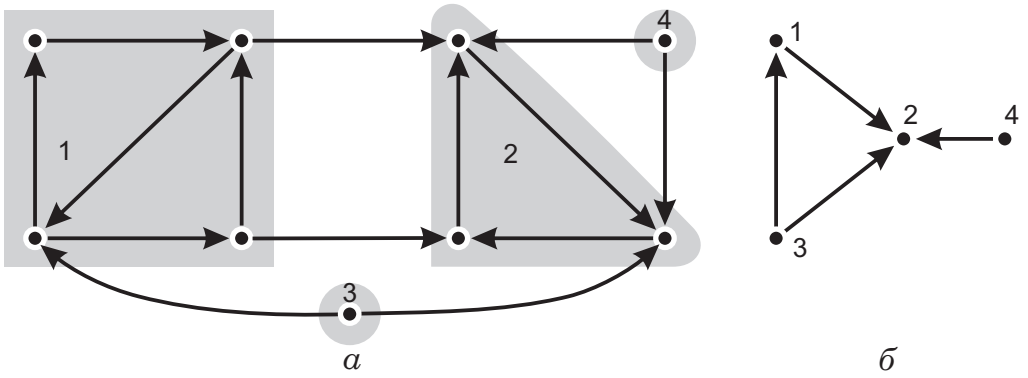


Рис. 4.24. Орграф (*а*) и его конденсация (*б*)

Нетрудно показать, что конденсация орграфа не содержит контуров. Действительно, любой контур представляет собой сильно связный подграф, входящий в состав некоторой сильной компоненты, которая, в свою очередь, соответствует некоторой вершине конденсации этого орграфа.

Глава 5

Булевы функции

В современных цифровых устройствах, таких как мобильный телефон или компьютер, информация хранится в двоичном представлении, то есть в виде векторов, элементами которых являются 0 или 1. В ходе вычислений функциональные элементы процессора дискретно, периодически с заданным интервалом времени выполняют преобразование одного двоичного вектора длины n в другой двоичный вектор длины m, то есть *преобразуют* информацию. Например, логическое дискретное цифровое устройство, называемое *сумматор*, преобразует вектор длины $2n$, хранящий два n-разрядных целых числа, в вектор длины n, хранящий сумму этих чисел.

Таким образом, возникает задача построения функции вида $F : \mathbb{B}^n \to \mathbb{B}^m$ по заданным значениям аргументов, подаваемым на вход, и результирующим значениям, которые должны получаться на выходе. Преобразователь информации, реализующий функцию F, можно изобразить в виде «черного ящика» (рис. 5.1). Решение общей задачи построения нужной функции можно свести к решению несколько более простых задач, заменив функцию F на m функций $f_i : \mathbb{B}^n \to \mathbb{B},\ i = 1, \ldots m$. Внутренняя схема преобразователя с учетом указанной декомпозиции представлена на рис. 5.2.

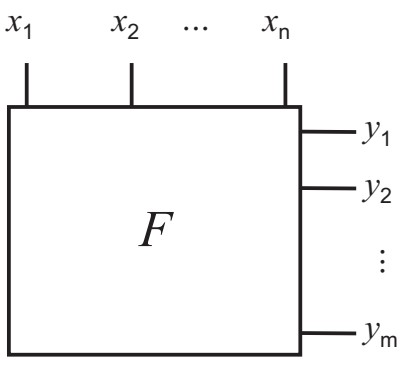


Рис. 5.1. Преобразователь информации в виде «черного ящика»

Определение 5.22. Функция вида $f : \mathbb{B}^n \to \mathbb{B}$ называется функцией *алгебры логики* или *булевой* функцией (по имени Дж. Буля (Bool), 1815-1864).

Множество булевых функций от n пере-

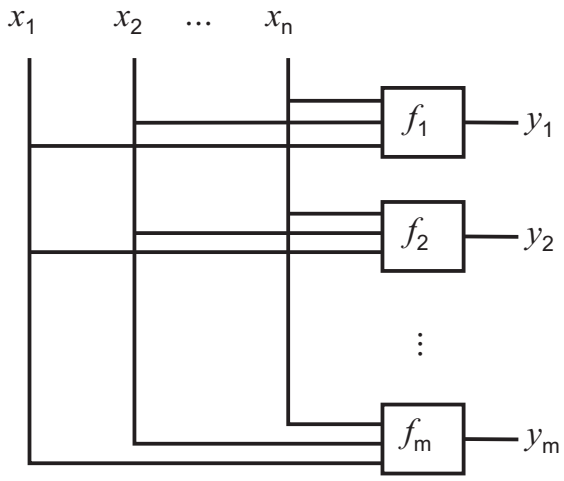


Рис. 5.2. Схема преобразователя, реализующего функцию $F : \mathbb{B}^n \to \mathbb{B}^m$

менных обозначается P_n:

$$P_n = \{f \mid f : \mathbb{B}^n \to \mathbb{B}\}.$$

Булеву функцию можно задать таблицей истинности:

x_1	$\ldots$	x_n	$f(x_1, \ldots, x_n)$
0	$\ldots$	0	$f(0, \ldots, 0)$
	$\ldots$		$\ldots$
1	$\ldots$	1	$f(1, \ldots, 1)$

Такая таблица содержит 2^n строк, поэтому число различных булевых функций от n переменных равно $|P_n| = 2^{2^n}$.

Определение 5.23. Говорят, что булева функция вида $f \in P_n$ *существенно зависит от переменной* x_i, если существует набор значений переменных $a_1, \ldots, a_n$ такой, что

$$f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \neq f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n).$$

Пример. Пусть булевы функции $f_1(x_1, x_2)$ и $f_2(x_1, x_2)$ заданы следующими таблицами истинности:

x_1	x_2	f_1	f_2
0	0	1	0
0	1	0	1
1	0	1	0
1	1	0	1

Для обеих функций переменная x_1 несущественная, а x_2 – существенная ($f_1(x_1, x_2) = \bar{x}_2$, $f_2(x_1, x_2) = x_2$).

Определение 5.24. Булевы функции *равны*, если одна из другой получается введением или удалением несущественной переменной.

В дальнейшем будем рассматривать и сравнивать булевы функции, зависящие от одних и тех же переменных (некоторые из которых могут оказаться несущественными).

Среди всего множества булевых функций выделяют множество F_0 *элементарных* булевых функций от одной и двух переменных:

x	0	1	$\bar{x}$
0	0	1	1
1	0	1	0

x	y	0	$\wedge$	$\vee$	$\oplus$	$\downarrow$	$\mid$	$\Rightarrow$	$\Leftrightarrow$	1
0	0	0	0	0	0	1	1	1	1	1
0	1	0	0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	1	0	0	1
1	1	0	1	1	0	0	0	1	1	1

Функции 0 и 1 называются, соответственно, *нуль* и *единица*. Через $\bar{x}$ обозначается *отрицание* x. Элементарные функции двух переменных имеют следующие названия:

- $\wedge$ - *конъюнкция*, *И*, *логическое произведение* (символ $\wedge$ часто опускают, вместо $x \wedge y$ пишут xy),
- $\vee$ - *дизъюнкция*, *ИЛИ*, *логическая сумма*,
- $\oplus$ - *сумма по модулю* 2, *исключающее ИЛИ*,
- $\downarrow$ - *стрелка Пирса*, *НЕ-ИЛИ*,
- $\mid$ - *штрих Шеффера*, *НЕ-И*,
- $\Rightarrow$ - *импликация*,
- $\Leftrightarrow$ - *эквивалентность*.

5.1. Формулы

Пусть задано некоторое непустое множество булевых функций $F \subseteq P^n$ (например, множество F_0). Дадим индуктивное определение формулы над множеством F:

Определение 5.25.

1. Каждая булева функция $f \in F$ называется *формулой над* F.
2. Пусть $f \in F$ и $A_1, \ldots, A_n$ – формулы над F. Тогда $f(A_1, \ldots, A_n)$ есть *формула над* F. Функция f называется *суперпозицией* формул A_i, а сами формулы A_i – *подформулами* формулы f.

3. Других определений формулы нет.

По определению, любую формулу над F можно представить как суперпозицию функций из F. Например, формула $(x \oplus y) \vee (\bar{x} \oplus \bar{y})$ является суперпозицией суммы по модулю 2, дизъюнкции и отрицания.

Каждой формуле $A(x_1, \ldots, x_n)$ над множеством F можно сопоставить булеву функцию $f(x_1, \ldots, x_n) = A(x_1, \ldots, x_n)$. Говорят, что формула A является *реализацией* функции f. Одна и та же булева функция может иметь множество реализаций. Формулы A и B, реализующие одну и ту же булеву функцию, называются *эквивалентными* (или *равносильными*). При этом пишут $A = B$.

В частности, имеют место следующие равносильности, справедливость которых легко установить с помощью таблиц истинности:

Коммутативность: $x \wedge y = y \wedge x$; $x \vee y = y \vee x$; $x \oplus y = y \oplus x$

Ассоциативность: $x \wedge (y \wedge z) = (x \wedge y) \wedge z$; $x \vee (y \vee z) = (x \vee y) \vee z$; $x \oplus (y \oplus z) = (x \oplus y) \oplus z$

Дистрибутивность: $x\,(y \vee z) = x\,y \vee x\,z$; $x \vee y\,z = (x \vee y)\,(x \vee z)$; $x\,(y \oplus z) = x\,y \oplus x\,z$

Идемпотентность: $x \wedge x = x$; $x \vee x = x$

Законы *де Моргана*: $\overline{x \vee y} = \bar{x} \wedge \bar{y}$; $\overline{x \wedge y} = \bar{x} \vee \bar{y}$

Инволютивность: $\bar{\bar{x}} = x$

Свойства *нуля* и *единицы*: $x \vee \bar{x} = 1,\ x \wedge \bar{x} = 0,\ x \oplus \bar{x} = 1$; $x \vee 0 = x,\ x \wedge 0 = 0,\ x \oplus 0 = x$; $x \vee 1 = 1,\ x \wedge 1 = x,\ x \oplus 1 = \bar{x}$; $x \oplus x = 0$

Правила *поглощения*: $(x \vee y)x = x,\ xy \vee x = x$

Другие равенства: $x \oplus y = x\bar{y} \vee \bar{x}y$; $x \Rightarrow y = \bar{x} \vee y$; $x \Leftrightarrow y = xy \vee \bar{x}\bar{y}$

Используются следующие сокращения:

$$\bigwedge_{i=1}^{n} = x_1 \wedge \ldots \wedge x_n, \quad \bigvee_{i=1}^{n} = x_1 \vee \ldots \vee x_n, \quad \bigoplus_{i=1}^{n} = x_1 \oplus \ldots \oplus x_n$$

5.2. Принцип двойственности

Определение 5.26. Пусть $f(x_1, \ldots, x_n) \in P_n$. Функция

$$f^*(x_1, \ldots, x_n) = \bar{f}(\bar{x_1}, \ldots, \bar{x_n})$$

называется *двойственной* к функции f.

Например:

f	1	0	x	$\bar{x}$	$x \vee y$	$x \wedge y$	$x \oplus y$
f^*	0	1	x	$\bar{x}$	$x \wedge y$	$x \vee y$	$x \Leftrightarrow y$

Из определения двойственной функции следует, что $(f^*)^* = f$.

Определение 5.27. Функция f называется *самодвойственной*, если

$$f(x_1, \ldots, x_n) = f^*(x_1, \ldots, x_n).$$

Теорема 5.27 (о двойственной функции). *Если булева функция* $\varphi(x_1, \ldots, x_n)$ *реализована формулой*

$$f(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)),$$

то булева функция $\varphi^*(x_1, \ldots, x_n)$ *реализуется формулой*

$$f^*(f_1^*(x_1, \ldots, x_n), \ldots, f_m^*(x_1, \ldots, x_n)).$$

Доказательство.

$$\begin{aligned}
\varphi^*(x_1, \ldots, x_n) &= \overline{\varphi}(\bar{x}_1, \ldots, \bar{x}_n) = \overline{f}(f_1(\bar{x}_1, \ldots, \bar{x}_n), \ldots, f_m(\bar{x}_1, \ldots, \bar{x}_n)) \\
&= \overline{f}(\overline{f}_1(\bar{x}_1, \ldots, \bar{x}_n), \ldots, \overline{f}_m(\bar{x}_1, \ldots, \bar{x}_n)) \\
&= \overline{f}(\overline{f_1^*}(x_1, \ldots, x_n), \ldots, \overline{f_m^*}(x_1, \ldots, x_n)) \\
&= f^*(f_1^*(x_1, \ldots, x_n), \ldots, f_m^*(x_1, \ldots, x_n)).
\end{aligned}$$

■

Следствие 5.9 (Принцип двойственности). *Пусть* $F = \{f_1, \ldots, f_m\}$ *и* $F^* = \{f_1^*, \ldots, f_m^*\}$. *Если формула* A *над* F *реализует функцию* f, *то формула* A^* *над* F^*, *полученная из* A *заменой* f_i *на* f_i^*, *реализует функцию* f^*.

Формула A^* называется *двойственной* к формуле A. Для формул над множеством $F = \{0, 1, \bar{x}, x \vee y, x \wedge y, x \oplus y\}$ принцип двойственности можно сформулировать так: для получения формулы, двойственной формуле A над F, нужно в формуле A заменить

$$\begin{array}{rcl} 0 & \text{на} & 1, \\ 1 & \text{на} & 0, \\ \vee & \text{на} & \wedge, \\ \wedge & \text{на} & \vee, \\ \oplus & \text{на} & \Leftrightarrow, \\ \Leftrightarrow & \text{на} & \oplus. \end{array}$$

Следствие 5.10. *Если* $A_1 = A_2$, *то* $A_1^* = A_2^*$.

В частности из закона де Моргана $\overline{x \vee y} = \bar{x}\,\bar{y}$ следует двойственный закон $\overline{x\,y} = \bar{x} \vee \bar{y}$.

5.3. Разложения булевых функций по переменным

Для функции $x \Leftrightarrow y$ введем обозначение

$$x^y = x \Leftrightarrow y = xy \vee \bar{x}\,\bar{y}.$$

По определению, $x^y = 1$ тогда и только тогда, когда $x = y$. Кроме того, переменная y может служить параметром, с помощью которого к переменной x можно применить отрицание: $x^1 = x$, $x^0 = \bar{x}$.

Теорема 5.28 (О разложении булевых функций по переменным).

$$f(x_1, \ldots, x_n) = \bigvee_{(\sigma_1, \ldots, \sigma_m)} x_1^{\sigma_1}\, x_2^{\sigma_2} \ldots x_m^{\sigma_m}\, f(\sigma_1, \ldots, \sigma_m, x_{m+1}, \ldots, x_n),$$

где дизъюнкция берется по всем наборам $(\sigma_1, \ldots, \sigma_m)$.

Доказательство. Рассмотрим произвольный набор значений переменных $a = a_1, \ldots, a_m, a_{m+1}, \ldots, a_n$ и оценим значение дизъюнкции на этом наборе:

$$\bigvee_{(\sigma_1, \ldots, \sigma_m)} a_1^{\sigma_1}\, a_2^{\sigma_2} \ldots a_m^{\sigma_m}\, f(\sigma_1, \ldots, \sigma_m, a_{m+1}, \ldots, a_n).$$

Для всех наборов $\sigma_1, \ldots, \sigma_m$, содержащих $\sigma_i \neq a_i,\ 1 \leq i \leq m$, конъюнкция $a_1^{\sigma_1} \ldots a_m^{\sigma_m}$ равна 0, поэтому дизъюнкция равна

$$a_1^{a_1} \ldots a_m^{a_m} f(a_1, \ldots, a_m, a_{m+1}, \ldots, a_n) = f(a_1, \ldots, a_n).$$

■

Следствие 5.11 (Случай для $m = 1$).

$$\begin{aligned} f(x_1, \ldots, x_n) &= x_i f(x_1, \ldots, x_{i-1}, 1, \ldots, x_n) \vee \\ &\quad \bar{x}_i f(x_1, \ldots, x_{i-1}, 0, \ldots, x_n). \end{aligned}$$

Следствие 5.12 (Случай для $m = n$).

$$\begin{aligned} f(x_1, \ldots, x_n) &= \bigvee_{(\sigma_1, \ldots, \sigma_n)} x_1^{\sigma_1} \ldots x_m^{\sigma_n} f(\sigma_1, \ldots, \sigma_n) \\ &= \bigvee_{f(\sigma_1, \ldots, \sigma_n)=1} x_1^{\sigma_1} \ldots x_n^{\sigma_n}. \end{aligned} \tag{5.1}$$

Правая часть равенства (5.1) называется *совершенной дизъюнктивной нормальной формой* (СДНФ).

Пример.

$$x \oplus y = \bigvee_{\substack{(0,1) \\ (1,0)}} x^{\sigma_x} y^{\sigma_y} = x^0 y^1 \vee x^1 y^0 = \bar{x} y \vee x \bar{y}.$$

Член разложения $K = x_1^{\sigma_1} \ldots x_n^{\sigma_n}$ называется *элементарной конъюнкцией.*

Определение 5.28. Представление булевой функции $f(x_1, \ldots, x_n)$ в форме дизъюнкции элементарных конъюнкций вида $\bigvee_{i=1}^{m} K_i$ называется *дизъюнктивной нормальной формой (ДНФ)* функции f. Если каждый конъюнкт K_i, $1 \leq i \leq m$, содержит в точности по одной переменной x_j, $1 \leq j \leq n$ (с отрицанием или без), то ДНФ называется *совершенной* (сокращенно *СДНФ*).

Из формулы (5.1) следует, что представление функции в виде СДНФ единственно.

Рассмотрим еще одну форму представления булевой функции $f(x_1, \ldots, x_n)$. Для этого выпишем разложение в виде СДНФ для функции f^*, двойственной к f:

$$f^*(x_1, \ldots, x_n) = \bigvee_{f^*(\sigma_1,\ldots,\sigma_n)=1} x^{\sigma_1} \ldots x^{\sigma_n}$$

Тогда

$$\begin{aligned} f(x_1, \ldots, x_n) = (f^*(x_1, \ldots, x_n))^* &= \bigwedge_{\overline{f}(\bar{\sigma_1},\ldots,\bar{\sigma}_n)=1} x^{\sigma_1} \vee \ldots \vee x^{\sigma_n} \\ &= \bigwedge_{f(\sigma_1,\ldots,\sigma_n)=0} x^{\bar{\sigma}_1} \vee \ldots \vee x^{\bar{\sigma}_n}. \end{aligned} \qquad (5.2)$$

Правая часть равенства (5.2) называется *совершенной конъюнктивной нормальной формой (СКНФ)*. Формула, двойственная к ДНФ, называется *конъюнктивной нормальной формой (КНФ)*.

Пример.

$$x \oplus y = \bigwedge_{\substack{(0,0)\\(1,1)}} x^{\bar{\sigma}_x} y^{\bar{\sigma}_y} = (x^1 \vee y^1)(x^0 \vee y^0) = (x \vee y)(\bar{x} \vee \bar{y}).$$

5.4. Минимизация булевых функций

В отличие от СДНФ, представление булевой функции в виде ДНФ не является единственным. Рассмотрим в качестве примера два разложения функции $f(x, y, z) = (\bar{x} \vee y) \oplus yz(x \vee z)$:

1. $\bar{x}\bar{y}\bar{z} \vee \bar{x}\bar{y}z \vee \bar{x}y\bar{z} \vee xy\bar{z}$ (СДНФ),
2. $\bar{x}\bar{y} \vee y\bar{z}$.

На основе этих разложений могут быть синтезированы две различные по сложности схемы преобразователей информации, реализующие одну и ту же булеву функцию (рис. 5.3). Эти схемы содержат разное число элементов и связей, поэтому созданные на их основе электронные устройства будут иметь разную скорость преобразования входного сигнала (x, y, z) в выходной сигнал $f(x, y, z)$. Таким образом, возникает задача минимизации сложности схемы, реализующей заданную булеву

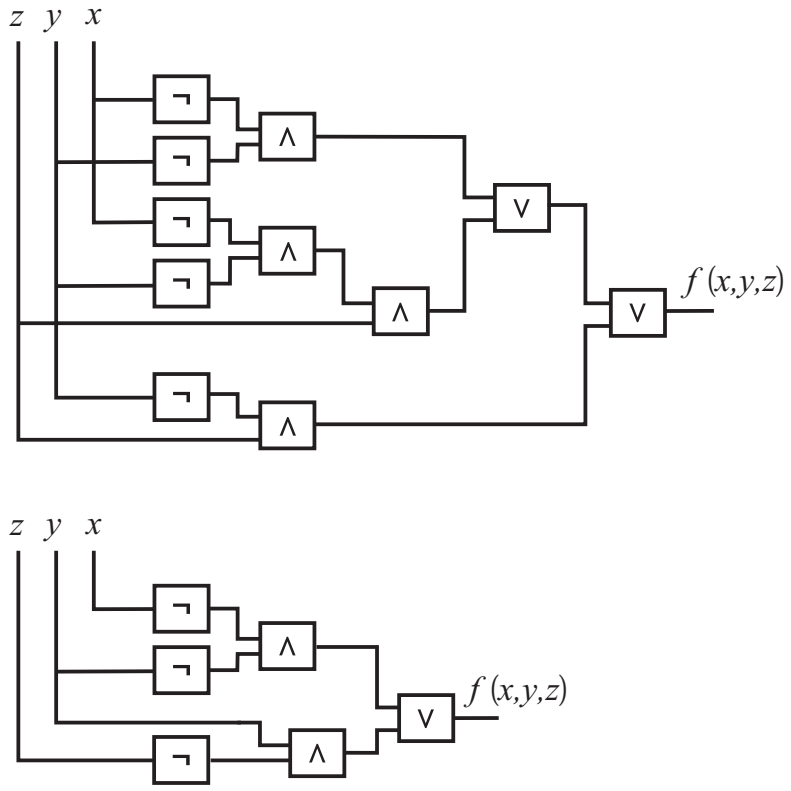


Рис. 5.3. Две схемы преобразователей информации, реализующих функцию $f(x, y, z) = (\bar{x} \vee y) \oplus yz(z \vee z)$

функцию формулой над заданным множеством элементарных булевых функций. Относительно множества $\{\wedge, \vee, \neg\}$ задача формулируется как поиск ДНФ минимальной сложности. Формализуем понятие сложности.

Определение 5.29. ДНФ называется

- *минимальной*, если она содержит наименьшее количество переменных и их отрицаний среди всех возможных ДНФ, реализующих данную функцию,
- *тупиковой*, если отбрасывание любого слагаемого или переменной приводит к неэквивалентной ДНФ.

В общем случае поиск минимальной ДНФ является гораздо более сложной задачей, чем поиск тупиковых ДНФ, причем для построения последних существуют сравнительно простые алгоритмы. Кроме того, среди множества всех тупиковых ДНФ, реализующих данную функцию, присутствует и минимальная ДНФ.

5.4.1. Метод Блейка

Для построения тупиковой ДНФ используется следующий метод:

1. Для данной булевой функции с помощью таблицы истинности строится СДНФ.

2. К построенной СДНФ применяется метод Блейка: для двух элементарных конъюнкций вида $xK_1 \vee \bar{x}K_2$ применяется правило обобщенного склеивания:

$$xK_1 \vee \bar{x}K_2 = xK_1 \vee \bar{x}K_2 \vee K_1K_2.$$

   Это правило применяется к ДНФ до тех пор, пока это возможно.

3. Для конъюнкций вида $K_1 \vee K_1K_2$ применяется правило поглощения: $K_1 \vee K_1K_2 = K_1$.

4. Удаляются лишние конъюнкции с помощью правила обобщенного склеивания, примененного «в обратном порядке»:

$$xK_1 \vee \bar{x}K_2 \vee K_1K_2 = xK_1 \vee \bar{x}K_2.$$

Пример. Пусть булева функция $f(x, y, z)$ задана таблицей:

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

1. Строим СДНФ:

$$f(x, y, z) = \bar{x}\bar{y}\bar{z} \vee \bar{x}\bar{y}z \vee \bar{x}yz \vee x\bar{y}\bar{z} \vee xy\bar{z} \vee xyz.$$

2. Применим правило обобщенного склеивания к первому и второму конъюнкту:

$$\bar{x}\bar{y}\bar{z} \vee \bar{x}\bar{y}z = \bar{x}\bar{y}\bar{z} \vee \bar{x}\bar{y}z \vee \bar{x}\bar{y}.$$

Аналогично поступим со вторым и третьим конъюнктом:

$$\bar{x}\bar{y}z \vee \bar{x}yz = \bar{x}\bar{y}z \vee \bar{x}yz \vee \bar{x}z.$$

Далее:

$$\bar{x}\bar{y}\bar{z} \vee x\bar{y}\bar{z} = \bar{x}\bar{y}\bar{z} \vee x\bar{y}\bar{z} \vee \bar{y}\bar{z},$$

$$xy\bar{z} \vee xyz = xy\bar{z} \vee xyz \vee xy.$$

В результате применения указанных операций получим эквивалентную ДНФ:

$$f(x,y,z) = \bar{x}\bar{y}\bar{z} \vee \bar{x}\bar{y}z \vee \bar{x}yz \vee x\bar{y}\bar{z} \vee xy\bar{z} \vee xyz \vee \bar{x}\bar{y} \vee \bar{x}z \vee \bar{y}\bar{z} \vee xy.$$

3. Последовательно применим правило поглощения:

$$\bar{x}\bar{y}\bar{z} \vee \bar{x}\bar{y} = \bar{x}\bar{y},$$

$$x\bar{y}\bar{z} \vee \bar{y}\bar{z} = \bar{y}\bar{z}$$

и т.д.

Сокращенная ДНФ выглядит следующим образом::

$$f(x,y,z) = \bar{x}\bar{y} \vee \bar{x}z \vee \bar{y}\bar{z} \vee xy.$$

4. Повторно применив правило обобщенного склеивания

$$\bar{x}\bar{y} \vee \bar{x}z \vee \bar{y}\bar{z} = \bar{x}z \vee \bar{y}\bar{z},$$

получаем искомую тупиковую ДНФ:

$$f(x,y,z) = \bar{x}z \vee \bar{y}\bar{z} \vee xy.$$

5.4.2. Геометрический метод

Рассмотрим булев куб $\mathbb{B}^n$ и булеву функцию $f(x_1,\ldots,x_n)$. Обозначим через N_f множество вершин булева куба $(x_1,\ldots,x_n) \in \mathbb{B}^n$ таких, что $f(x_1,\ldots,x_n) = 1$. Очевидно, что отображение $f \to N_f$ биективно. На рис. 5.4 изображен булев куб $\mathbb{B}^3$ и множество N_f для функции $f(x,y,z)$ из рассмотренного выше примера.

Подмножество вершин $\mathbb{B}^n$, r координат которых совпадают $(r \le n)$, называется $(n-r)$-*мерной гранью* булева куба. В частности, вершины куба $\mathbb{B}^3$ является его 0-мерными гранями, а ребра – 1-мерными.

Рассмотрим элементарную конъюнкцию

$$K = K(x_1,\ldots,x_n) = x_{i_1}^{\sigma_1}\ldots x_{i_r}^{\sigma_r}$$

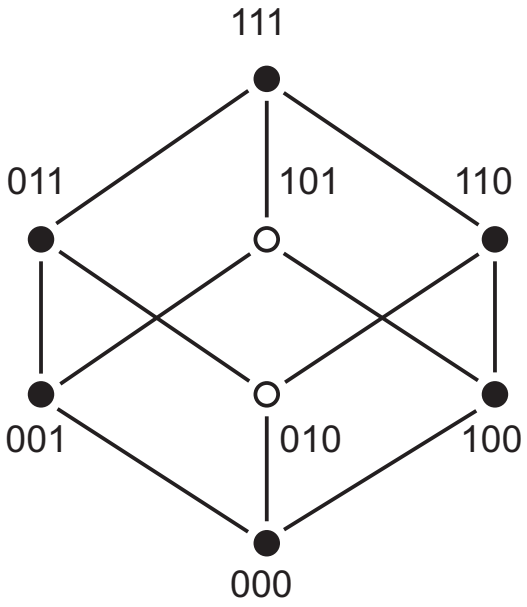


Рис. 5.4. Булев куб $\mathbb{B}^3$ и подмножество вершин N_f (обозначены черным цветом)

и соответствующее ей множество N_K вершин $\mathbb{B}^n$. Заметим, что $K = 1$ тогда и только тогда, когда $x_{i_j} = \sigma_j$, $1 \le j \le r$, поэтому N_K представляет собой $(n-r)$-мерную грань.

Заметим, что чем меньше переменных в конъюнкции, тем выше размерность соответствующей грани. В качестве примера рассмотрим три конъюнкции (рис. 5.5):

$$\begin{array}{lll} K_1(x,y,z) = \bar{x}yz & N_{K_1} & = \{(0,1,1)\} \\ K_2(x,y,z) = \bar{x}\bar{y} & N_{K_2} & = \{(0,0,0),(0,0,1)\} \\ K_3(x,y,z) = x & N_{K_3} & = \{(1,0,0),(1,0,1),(1,1,0),(1,1,1)\} \end{array}$$

Геометрический метод минимизации ДНФ основан на следующем простом утверждении.

Теорема 5.29. *Если* $f(x_1,\ldots,x_n) = g(x_1,\ldots,x_n) \vee h(x_1,\ldots,x_n)$, *то* $N_f = N_g \cup N_h$.

Другими словами, система множеств $\{N_g, N_h\}$ есть покрытие множества N_f. В частности, если функция f представима в виде ДНФ $f = K_1 \vee \ldots \vee K_m$, то $N_f = N_{K_1} \cup \ldots \cup N_{K_m}$. Отсюда следует, что всякой ДНФ взаимно-однозначно соответствует некоторое покрытие множества N_f некоторыми гранями булева куба.

Обозначим через $\dim K_i$ размерность грани N_{K_i}. Число $\sum_{i=1}^{m} \dim K_i$ называется *размерностью покрытия*. Тогда задача о минимизации

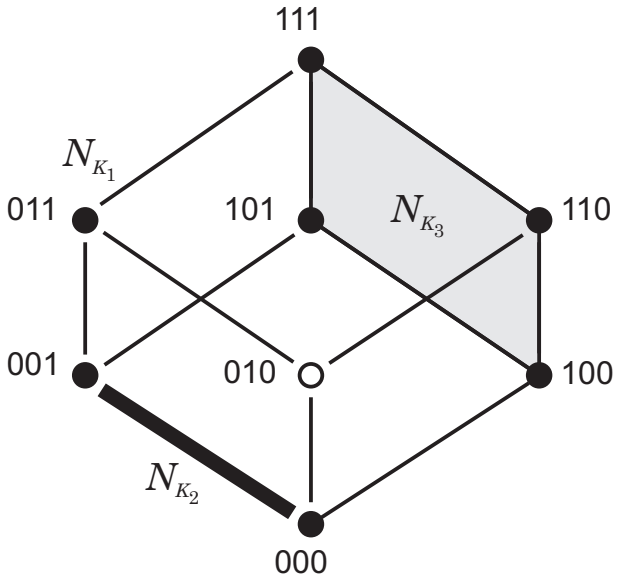


Рис. 5.5. Булев куб $\mathbb{B}^3$ и грани, соответствующие конъюнкциям K_1, K_2 и K_3

булевой функции имеет следующую геометрическую интерпретацию: *для данного множества N_f необходимо найти такое его покрытие гранями $N_{K_1} \cup \ldots \cup N_{K_m}$, которое имеет максимальную размерность.*

Из этой интерпретации следует алгоритм построения тупиковой ДНФ, применимый для небольшого числа переменных:

1. На кубе $\mathbb{B}^n$ для данной булевой функции f отметить множество вершин N_f.
2. Найти покрытие множества N_f гранями N_i, имеющее минимальный ранг.
3. Каждой из полученных граней N_i поставить в соответствие элементарную конъюнкцию искомой ДНФ.

Пример. Для функции из предыдущего примера (раздел 5.4.1.) обозначим на булевом кубе $\mathbb{B}^3$ множество вершин N_f (рис. 5.6 *а*). Нетрудно заметить, что существует всего два способа покрытия выделенного множества вершин гранями N_i, все вершины которых принадлежат N_f (рис. 5.6 *б* и *в*), а именно:

$$\begin{array}{rclcrcl} N_1 & = & \{(0,0,0),(0,0,1)\} & & N_4 & = & \{(0,0,0),(1,0,0)\} \\ N_2 & = & \{(1,0,0),(1,1,0)\} & \text{и} & N_5 & = & \{(0,0,1),(0,1,1)\} \\ N_3 & = & \{(0,1,1),(1,1,1)\} & & N_6 & = & \{(1,1,0),(1,1,1)\} \end{array}$$

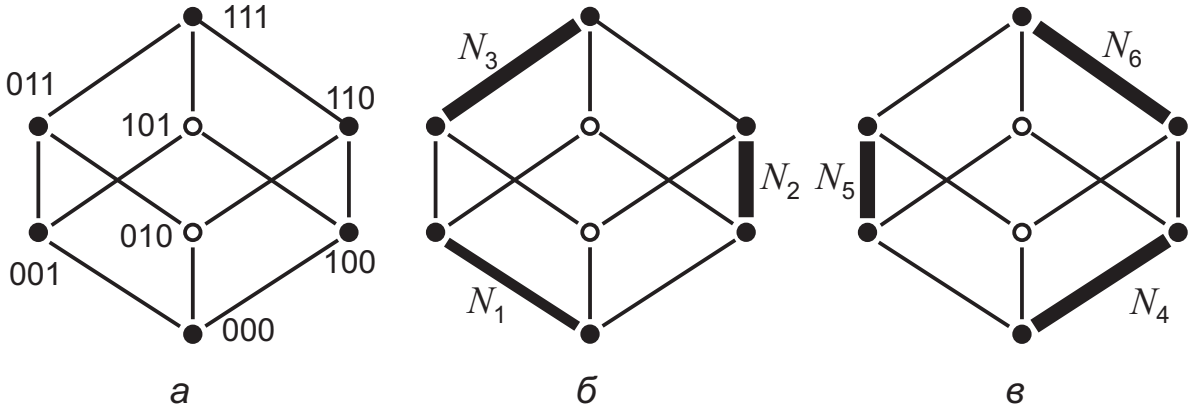


Рис. 5.6. Множество вершин N_f ($а$) и два варианта его покрытия – ($б$) и ($в$)

Данным покрытиям соответствуют следующие тупиковые ДНФ (одна из них построена с помощью метода Блейка в предыдущем разделе):

$$\bar{x}_1\bar{x}_2 \vee x_1\bar{x}_3 \vee x_2x_3 \quad \text{и} \quad \bar{x}_2\bar{x}_3 \vee \bar{x}_1x_3 \vee x_1x_2.$$

5.5. Функциональная полнота и замкнутость

5.5.1. Базис и замыкание

Из единственности СДНФ вытекает следующая теорема:

Теорема 5.30. *Любая булева функция может быть представлена в виде суперпозиции только трех элементарных функций: конъюнкции, дизъюнкции и отрицания.*

Определение 5.30. *Функционально полным базисом* (или просто *базисом*) называется множество булевых функций, суперпозицией которых может быть представлена любая булева функция.

Таким образом, конъюнкция, дизъюнкция и отрицание образуют базис, который называется *базисом Буля* (Дж. Буль (Bool), 1815-1864).

Ответим на следующие вопросы:

1. Существуют ли другие системы булевых функций, отличные от базиса Буля и обладающие свойством полноты?
2. Какими такими свойствами обладают функции конъюнкция, дизъюнкция и отрицание, что они образуют базис?

3. Является ли базис Буля минимальным (по количеству входящих в него функций)?

Определение 5.31. *Замыканием* множества F булевых функций, обозначается $[F]$, называется множество всех булевых функций, которые могут быть получены суперпозицией функций из F.

Стало быть, F является базисом, если $[F] = P_n$.

Свойства замыкания:

1. $F \subseteq [F]$.
2. $[[F]] = [F]$, $[P_n] = P_n$.
3. $F \subseteq G \Rightarrow [F] \subseteq [G]$.
4. $[F] \subseteq P_n$.
5. Если F – базис и $F \subseteq [G]$, то G – тоже базис.

Доказательство первого свойства очевидно: любая функция из F является суперпозицией, состоящей из одной этой функции. Из первого свойства следует , что $[F] \subseteq [[F]]$. Но суперпозиция суперпозиций над F также является суперпозицией над F, поэтому $[[F]] \subseteq [F]$. Свойства (3) и (4) также следуют непосредственно из определений. Из условия свойства (5) следует, что любая функция из F есть суперпозиция над G, поэтому любая булева функция может быть представлена суперпозицией функций из G.

Рассмотрим другие функционально полные системы булевых функций, отличные от базиса Буля $\{\wedge, \vee, \neg\}$.

1. Система $\{\neg, \wedge\}$ является полной, поскольку $x \wedge y = \overline{\bar{x}\bar{y}}$.
2. Система $\{\neg, \vee\}$ является полной, поскольку $x\, y = \overline{\bar{x} \vee \bar{y}}$.
3. Система $\{\,|\,\}$ является полной, поскольку

$$\bar{x} = x \,|\, x,\ x\, y = \overline{x \,|\, y} = (x \,|\, y) \,|\, (x \,|\, y).$$

4. Система $\{\downarrow\}$ является полной, поскольку

$$\bar{x} = x \downarrow x,\ x \vee y = \overline{x \downarrow y} = (x \downarrow y) \downarrow (x \downarrow y).$$

5.5.2. Полином Жегалкина и линейные функции

Рассмотрим отдельно систему $\{\wedge, \oplus, 1\}$, где 1 обозначает функцию «тождественная единица». Если в любой формуле, включающей только функции данного множества, раскрыть скобки, то получим сумму по модулю 2 логических произведений (конъюнкций) переменных, то есть полином. Такой полином называется *полиномом Жегалкина*.

Теорема 5.31. *Любая булева функция может быть представлена в виде полинома Жегалкина, причем единственным образом. Таким образом, система функций* $\{\wedge, \oplus, 1\}$ *образует базис.*

Доказательство. Действительно, отрицание представимо в виде следующей суперпозиции: $\bar{x} = 1 \oplus x$. Для дизъюнкции имеем:

$$\begin{aligned} x \vee y &= \overline{\bar{x}\bar{y}} \\ &= 1 \oplus (1 \oplus x)(1 \oplus y) \\ &= 1 \oplus (1 \oplus x \oplus y \oplus x\,y) \\ &= x \oplus y \oplus x\,y. \end{aligned}$$

■

Пример. Функция $f(x, y, z) = (\bar{x} \vee y) \oplus y\,z\,(x \vee z)$ в базисе Жегалкина имеет следующее представление: $1 \oplus x \oplus x\,y \oplus y\,z$.

Определение 5.32. Булева функция, полином Жегалкина которой имеет вид

$$a_0 \oplus \bigoplus_{i=1}^{n} a_i\,x_i,\ a_i \in \{0, 1\}$$

называется *линейной*.

5.5.3. Замкнутые классы булевых функций

Определение 5.33. Класс F булевых функций называется *замкнутым*, если $[F] = F$.

Рассмотрим следующие классы булевых функций:

1. Класс функций, *сохраняющих* 0:
$$T_0 = \{f \mid f(0, \ldots, 0) = 0\}.$$
2. Класс функций, *сохраняющих* 1:
$$T_1 = \{f \mid f(1, \ldots, 1) = 1\}.$$

3. Класс *самодвойственных* функций:

$$T_* = \{f \mid f^* = f\}.$$

4. Класс *монотонных* функций:

$$T_\leq = \{f \mid \forall (x, y \in \mathbb{B}^n)\ (x \leq y) \Rightarrow f(x) \leq f(y)\},$$

где $(x \leq y) \Leftrightarrow (x_i \leq y_i),\ 1 \leq i \leq n.$

5. Класс *линейных* функций:

$$T_L = \{f \mid f(x_1, \ldots, x_n) = a_0 \oplus \bigoplus_{i=1}^{n} a_i\, x_i,\ a_i \in \{0, 1\}\}.$$

Теорема 5.32. *Классы T_0, T_1, T_*, $T_\leq$ и T_L замкнуты.*

Идея доказательства: нужно показать, что если булева функция реализована в виде формулы над классом T, то она принадлежит этому классу.

Пример.

- Конъюнкция принадлежит классам T_0, T_1 и $T_\leq$:

$$0 \wedge 0 = 0,\ 1 \wedge 1 = 1,$$
$$(x_1, \ldots, x_n) \leq (y_1, \ldots, y_n) \Rightarrow \bigwedge_{i=1}^{n} x_i \leq \bigwedge_{i=1}^{n} y_i.$$

- Отрицание не является монотонной функцией ($0 \leq 1$, но $\bar{0} > \bar{1}$), но является линейной ($\bar{x} = 1 \oplus x$).

- Дизъюнкция не является линейной функцией:

$$x \vee y = x \oplus y \oplus x\, y.$$

В следующей теореме сформулировано то самое специфическое свойство, которое отличает базисы от остальным множеств булевых функций.

Теорема 5.33 (Поста). *Система булевых функций F полна тогда и только тогда, когда она содержит:*

1. хотя бы одну функцию, не сохраняющую 0,

2. *хотя бы одну функцию, не сохраняющую* 1,

3. *хотя бы одну немонотонную функцию,*

4. *хотя бы одну несамодвойственную функцию,*

5. *хотя бы одну нелинейную функцию.*

Пример. В следующей таблице отмечена принадлежность элементарных булевых функций указанным замкнутым классам:

	T_0	T_1	T_*	$T_\leq$	T_L	Примечания
0	•			•	•	
1		•		•	•	
$\bar{x}$			•		•	
$x \wedge y$	•	•		•		
$x \vee y$	•	•		•		
$x \Rightarrow y$		•				
$x \oplus y$	•				•	
$x \downarrow y$						$\leftarrow$ базис!
$x \mid y$						$\leftarrow$ базис!
x	•	•	•	•	•	$\leftarrow$ избыточная функция в любом базисе

На основе данной таблицы легко построить функционально полные системы булевых функций, включая уже известные базисы, например: $\{1, \wedge, \oplus\}$, $\{\neg, \Rightarrow\}$, $\{\Rightarrow, 0\}$ и другие.

Глава 6

Формальные грамматики и языки

6.1. Основные определения

Пусть задано конечное множество $V = \{a_1, \ldots, a_n\}$, которое называется *алфавитом*. Элементы $a_i \in V$ называются *буквами*, а последовательность букв – *словами*. Множество слов в алфавите V обозначается V^*. Через $\Lambda \in V^*$ (лямбда) обозначается пустое слово. Подмножество $L \subseteq V^*$ называется *языком*.

В общем случае язык это очень большое множество, которое не может быть описано простым перечислением элементов. Поэтому для задания языка необходим некий конечный механизм. Под *грамматикой* понимают совокупность правил, которые используются для описания языка. Правила позволяют строить одни слова из других с помощью замены некоторых цепочек символов на другие. Правила замены обычно записываются в виде $\alpha \to \beta$, где α и β – слова из некоторых алфавитов.

Рассмотрим пример. Пусть символ S означает термин «сумма». Тогда соответствующее грамматическое правило можно записать в виде $S \to A + B$, где символы A и B обозначают термин «слагаемое». Введем также правила, уточняющие термин «слагаемое»:

$$A \to 0,\ A \to 1, \ldots, A \to 9,\ B \to 0,\ B \to 1, \ldots, B \to 9.$$

Для краткости эти правила записывают в виде:

$$A \to 0|1|\ldots|9,\ B \to 0|1|\ldots|9.$$

Если символы в правой части правила не подлежат дальнейшему уточнению, то заданная совокупность правил описывает язык, который включает множество всех формальных выражений сумм двух целых неотрицательных чисел, меньших десяти. Действительно, используя

указанные правила, запишем последовательность подстановок, позволяющих построить выражение $8 + 5$:

$$S \to A + B \to 8 + B \to 8 + 5.$$

Заметим, что в приведенных правилах символы S, A и B можно заменять другими, а символы $+$, $0, \ldots, 9$ – нельзя. Символы, для которых не существует правила замены, называются *терминальными* (то есть окончательными) или *основными*, а символы, которые можно заменить – *нетерминальными* или *вспомогательными*.

Дадим теперь формальное определение грамматики.

Определение 6.1. *Порождающей грамматикой* или *П-грамматикой* называется упорядоченная четверка $\Gamma = \langle V, W, J, R \rangle$, где

- V – непустой конечный алфавит терминальных символов,
- W – непустой конечный алфавит нетерминальных символов,
- J – начальный символ,
- R – конечное множество слов вида $\varphi \to \psi$, называемых *правилами*, где φ и ψ – различные слова в алфавите $V \cup W$ и символ $\to$ не принадлежит $V \cup W$.

В нашем примере

$$\begin{aligned}\Gamma_1 \;=\; \{&\{+, 0, \ldots, 9\}, \{S, A, B\}, S,\\ &\{S \to A + B,\ A \to 0|1|\ldots|9,\ B \to 0|1|\ldots|9\}\}.\end{aligned}$$

Грамматика позволяет получать (выводить) одни слова из других. Понятие *выводимого слова* в грамматике определяется рекурсивно следующим образом.

Определение 6.2. Пусть $\alpha, \beta, \varphi, \psi \in (V \cup W)^*$.

1. J – выводимое слово грамматики $\Gamma = \langle V, W, J, R \rangle$.
2. Если $\alpha\varphi\beta$ – выводимое слово грамматики Γ и $\varphi \to \psi \in R$, то $\alpha\psi\beta$ есть также выводимое слово. При этом пишут $\alpha\varphi\beta \vdash \alpha\psi\beta$.
3. Других правил построения выводимых слов нет.

Последовательность слов $\varphi_1, \ldots, \varphi_n$ называется *выводом* слова φ_n из слова φ_1 в грамматике Γ, если $\varphi_i \vdash \varphi_{i+1}$, $i = 1, \ldots, n-1$. При этом используется запись $\varphi_1 \models \varphi_n$.

Примерами выводимых слов в грамматике Γ_1 являются: S, $A + B$, $8 + 5$, $1 + 4$ и т.д.

Определение 6.3. Множество всевозможных выводимых в грамматике Γ слов, состоящих только из терминальных символов, называется *языком*, порожденным грамматикой Γ и обозначается $L(\Gamma)$.

Например, $L(\Gamma_1) = \{a + b \mid a, b \in \{0, \ldots, 9\}\}$.

Определение 6.4. П-грамматика $\Gamma = \langle V, W, J, R\rangle$, в которой каждое правило из R имеет вид

$$\alpha A \beta \to \alpha \omega \beta, \tag{6.1}$$

где $\alpha, \beta, \omega \in (V, W)^*$, $\omega \neq \Lambda$, $A \in W$, называется *контекстно-зависимой* грамматикой или *КЗ-грамматикой*. В правиле (6.1) слова α и β называются контекстами.

Пример. Грамматика

$$\Gamma_2 = \{\{a, b, c\}, \{J, A, B, C\}, J, R\}$$

где множество R состоит из правил:

$$\begin{array}{lcl} J & \to & aJAC \mid abC, \\ CA & \to & BA, \\ BA & \to & BC, \\ BC & \to & AC, \\ bA & \to & bb, \\ C & \to & c, \end{array}$$

является КЗ-грамматикой. Можно доказать, что

$$L(\Gamma_2) = \{a^n b^n c^n \mid n \geq 1\}.$$

Определение 6.5. П-грамматика $\Gamma = \langle V, W, J, R\rangle$, в которой каждое правило из R имеет вид $A \to \omega$, где $A \in W$, $\omega \in (V, W)^*$, называется *контекстно-свободной* или *КС-грамматикой*.

Пример. Грамматика

$$\Gamma_3 = \{\{a, b\}, \{J\}, J, \{J \to aJb \mid ab\}\}$$

является КС-грамматикой, порождающей язык

$$L(\Gamma_3) = \{a^n b^n \mid n \geq 1\}.$$

Определение 6.6. П-грамматика $\Gamma = \langle V, W, J, R\rangle$, в которой каждое правило из R имеет вид $A \to aB$ или $A \to a$, где $A, B \in W,\ a \in V$, называется *автоматной* или *А-грамматикой.*

Пример. Грамматика

$$\Gamma_4 = \langle \{0,1\}, \{J\}, J, \{J \to 0 \mid 0J \mid 1J\}\rangle$$

является А-грамматикой, описывающей множество $L(\Gamma_4)$ четных двоичных чисел.

Легко заметить, что любая А-грамматика является КС-грамматикой, в свою очередь КС-грамматика является КЗ-грамматикой с пустыми контекстами, а КЗ-грамматика является П-грамматикой. Данная классификация грамматик образует так называемую *иерархию Хомского.*

Определение 6.7. Языки, порождаемые П-, КЗ-, КС- и А-грамматиками, называются, соответственно, *П-*, *КЗ-*, *КС-* и *А-языками.*

Обозначим через $\mathcal{L}_{\text{П}}$, $\mathcal{L}_{\text{КЗ}}$, $\mathcal{L}_{\text{КС}}$ и $\mathcal{L}_{\text{А}}$ соответствующие множества языков. Из иерархии Хомского следует, что для этих множеств имеет место соотношение

$$\mathcal{L}_{\text{А}} \subset \mathcal{L}_{\text{КС}} \subset \mathcal{L}_{\text{КЗ}} \subset \mathcal{L}_{\text{П}}.$$

Заметим, что включения множеств являются строгими. Покажем, например, что $\mathcal{L}_{\text{А}} \subset \mathcal{L}_{\text{КС}}$. Для этого рассмотрим, язык

$$L = \{a^n b^n \mid n \geq 1\},$$

который порождается КС-грамматикой

$$\Gamma = \{\{a,b\}, \{J\}, J, \{J \to aJb \mid ab\}\}$$

и, стало быть, является КС-языком.

Необходимо доказать, что язык L не порождается ни одной А-грамматикой. От противного: пусть язык L порождается некоторой А-грамматикой $\Gamma_A = \langle \{a,b\}, W, J, R\rangle$, где множество R состоит только из правил вида

$$A \to \alpha B \text{ или } A \to \alpha,\ \alpha \in \{a,b\}.$$

Значит, каждый вывод в грамматике Γ_A будет иметь вид:

$$J \to \alpha_1 A_1 \to \alpha_1\alpha_2 A_2 \to \ldots$$

Рассмотрим слово $a^n b^n$. Для его вывода А-грамматика Γ_A допускает единственный вариант применения цепочки правил:

$$J \to aA_{i_1} \to aaA_{i_2} \to \ldots \to a^n A_{i_n} \to a^n b A_{i_{n+1}} \to \ldots \to a^n b^n.$$

Пусть $n > |W|$. Тогда среди символов $A_{i_1} \ldots A_{i_n}$ найдутся одинаковые нетерминальные символы A_{i_l} и A_{i_m}, $1 \le l < m \le n$. Это означает, что в процессе вывода слова цепочка правил

$$A_{i_l} \to aA_{i_{l+1}} \to \ldots \to a^{m-l} A_{i_m}$$

может быть применена произвольное число раз, откуда, в свою очередь, следует, что грамматика Γ_A, помимо слов вида $a^n b^n$, допускает слова $a^s b^t$, где $s \neq t$. Это противоречит предположению, что Γ_A порождает все слова из L и только их.

Аналогично можно показать, что язык $L = \{a^n b^n c^n \,|\, n \ge 1\}$ порождается КЗ-грамматикой и не порождается никакой КС-грамматикой, то есть вхождение $\mathcal{L}_{\text{КС}} \subset \mathcal{L}_{\text{КЗ}}$ также строгое.

6.2. А-грамматики и конечные автоматы

6.2.1. Конечные автоматы

Под термином «автомат» подразумевается некий преобразователь информации. В отличие от преобразователя вида

$$\text{Вход} \to \mathbf{F} \to \text{Выход}$$

автомат помнит предысторию входов и формирует выходной сигнал в зависимости от этой предыстории. Понятно, что вариантов предыдущих сигналов может быть очень много, и если автомат будет вести себя по-разному для каждой предыстории, то он должен будет иметь и огромный ресурс памяти. Поэтому всевозможные предыстории объединяют в классы эквивалентности, причем для двух предысторий из одного класса автомат выдает одинаковый выходной сигнал (рис. 6.1). По определению, предыстории, входящие в один класс эквивалентности, считаются эквивалентными и называются *внутренними состояниями автомата*. Если число состояний автомата конечно, то он называется *конечным*.

Рассмотрим пример. Разработаем тактику поведения девушки, ожидающей свидания с молодым человеком. Упростим ситуацию и оставим всего два варианта развития событий:

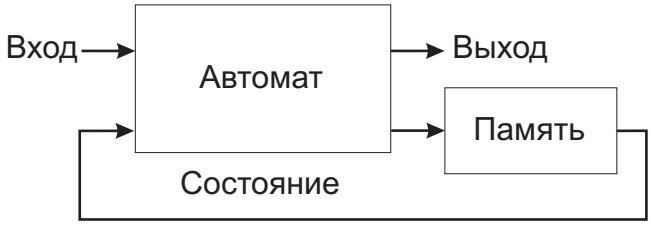


Рис. 6.1. Схема работы конечного автомата

1. Он является с опозданием и с бутылкой пива.
2. Он приходит вовремя и с букетом ее любимых цветов.

В случае, если молодой человек провинился, девушка не разрывает отношения сразу, а выбирает тонкую политику воспитания с учетом предыдущего опыта. Соответствующий автомат изображен на рис. 6.2.

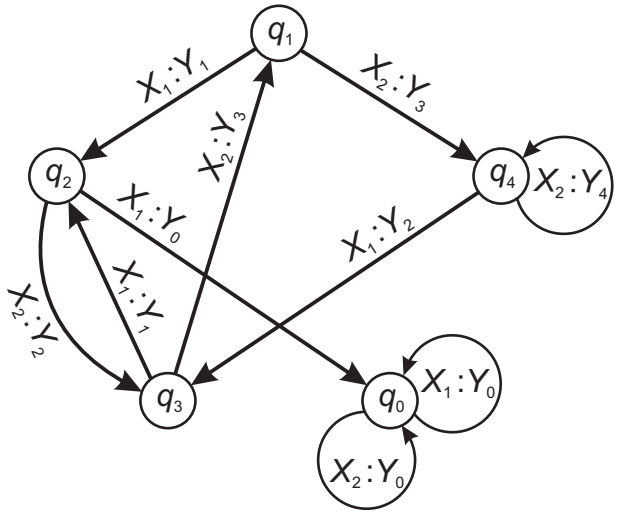


Рис. 6.2. Тактика поведения девушки на свидании.

Обозначения:

X_1 – опоздал,
X_2 – пришел вовремя,
Y_0 – разорвать отношения,
Y_1 – обидеться,
Y_2 – простить,
Y_3 – надеяться,
Y_4 – радоваться.

В этом орграфе вершины соответствуют состояниям автомата, а дуги имеют пометки, отражающие событие, связанное с появлением приятеля девушки, и ее реакцию.

Определение 6.8. *Детерминированным конечным автоматом* (КА) называется шестерка $A = \langle Q, X, Y, \delta, \lambda, q_1 \rangle$, где

Q – непустое конечное множество состояний,
X – непустое конечное множество входных сигналов,
Y – непустое конечное множество выходных сигналов,
$q_1 \in Q$ – начальное состояние,
$\delta : Q \times X \to Q$ – функция переходов,
$\lambda : Q \times X \to Y$ – функция выходов.

Функции переходов и входов можно представлять как в виде орграфа, так и в виде таблицы:

	q_1	q_2	q_3	q_4	q_0
X_1	$q_2 : Y_1$	$q_0 : Y_0$	$q_2 : Y_1$	$q_3 : Y_2$	$q_0 : Y_0$
X_2	$q_4 : Y_3$	$q_3 : Y_2$	$q_1 : Y_3$	$q_4 : Y_4$	$q_0 : Y_0$

6.2.2. Распознающие А-грамматики

Порождающие грамматики языка L позволяют построить все правильные предложения языка L и ни одного неправильного. В отличие от порождающей грамматики, *распознающая* грамматика задает критерий принадлежности произвольной цепочки символов $\alpha \in V^*$ данному языку. Фактически, это есть алгоритм, который разделяет все входные цепочки на два класса: принадлежащие языку L и не принадлежащие языку L. Роль распознающей грамматики может выполнить конечный автомат без выхода.

Определение 6.9. *Конечным автоматом-распознавателем* называется пятерка $A = \langle Q, V, \delta, q_1, Q_0 \rangle$, где

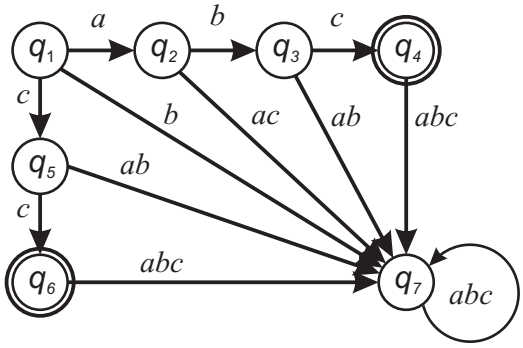
Q – непустое конечное множество состояний,
V – непустое конечное множество входных символов (входной алфавит),
$\delta : Q \times V \to Q$ – функция переходов,
$q_1 \in Q$ – начальное состояние,
$Q_0 \subseteq Q$ – множество заключительных состояний.

Иногда вместо функции перехода δ удобно использовать *расширенную* функцию перехода $\delta^* : Q \times V^* \to Q$, которая определяется следующим образом:

1. $\delta^*(q, \Lambda) = q$,
2. $\delta^*(q, a\alpha) = \delta^*(\delta(q, a), \alpha)$.

Если $\delta^*(q_1, \alpha) = q_2$, то говорят, что слово α *переводит* КА из состояния q_1 в состояние q_2. Если $\delta^*(q_1, \alpha) \in Q_0$, то есть слово α переводит КА из начального состояния в одно из заключительных состояний, то говорят, что КА *допускает* входное слово α. Говорят также, что КА *распознает* язык L, если он допускает те и только те слова из V^*, которые принадлежат языку L.

Пример: Пусть $V = \{a, b, c\}$ и $L = \{abc, cc\}$. Соответствующий конечный автомат-распознаватель A выглядит следующим образом:



Здесь цепочки abc и cc и только они переводят автомат в одно из заключительных состояний: q_4 или q_6.

Из данного примера видно, что любой конечный язык может быть задан конечным автоматом, поскольку существует взаимно-однозначное соответствие между словами языка и множеством конечных состояний.

Предложение 6.1. *Все конечные языки – автоматные.*

Пример (неавтоматный язык). Рассмотрим язык $L = \{a^n b^n \mid n \geq 0\}$ с бесконечно большим числом слов. Попытка построить КА для языка L приводит к тому, что число состояний бесконечно возрастает (рис. 6.3).

Определение 6.10. *Недетерминированным* конечным автоматом-распознавателем (НКА) называется пятерка $A = \langle Q, V, \delta, q_1, Q_0 \rangle$, где

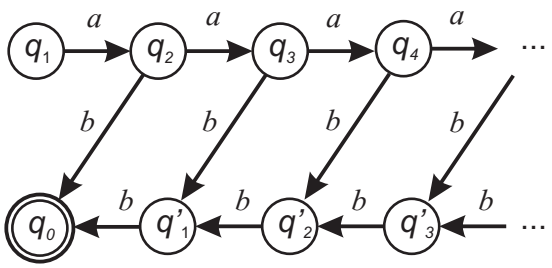
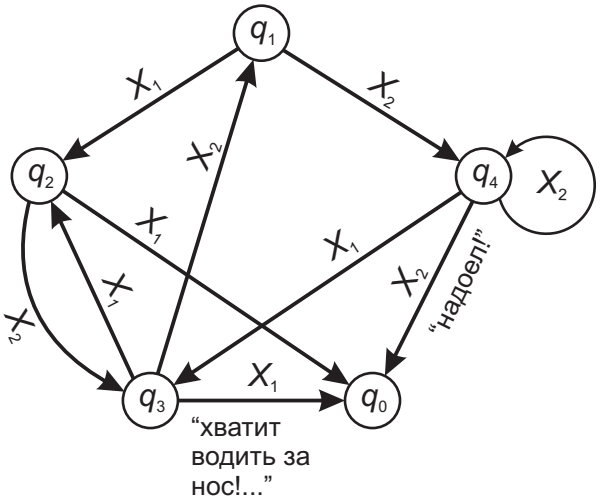


Рис. 6.3. Попытка построения КА, распознающего бесконечный язык

Q	–	непустое конечное множество состояний,
V	–	непустое конечное множество входных сигналов (входной алфавит),
$\delta : Q \times V \to 2^Q$	–	функция переходов, отображающая пару $(q, a) \in Q \times V$ в некоторое множество состояний,
$q_1 \in Q$	–	начальное состояние,
$Q_0 \subseteq Q$	–	множество заключительных состояний.

Таким образом, НКА, находясь в состоянии q, при получении входного сигнала может перейти в *одно из нескольких* состояний, определенных недетерминированной функцией перехода δ.

Пример («Недетерминированная девушка»):



Недетерминированная девушка-автомат, будучи в определенном состоянии, может поступать непредсказуемо, то есть совершать *один из нескольких* поступков, как показано на рисунке. Данный НКА имеет следующее табличное представление:

	q_1	q_2	q_3	q_4	q_0
X_1	q_2	q_0	q_2	q_3	
			q_0		
X_2	q_4	q_3	q_1	q_4	
				q_0	

Чтобы формально описать работу НКА, определим *недетерминированную расширенную функцию переходов* $\delta_N^* : Q \times V \to 2^Q$:

1. $\delta_N^*(q, \Lambda) = \{q\}$.
2. $\delta_N^*(q, a\,\alpha) = \bigcup\limits_{q' \in \delta(q,a)} \delta_N^*(q', \alpha)$.

Определение 6.11. Говорят, что НКА *допускает* входное слово $\alpha \in V^*$, если множество $\delta_N^*(q_1, \alpha)$ содержит хотя бы одно заключительное состояние из Q_0. Говорят, что НКА *распознает* язык L, если он допускает все слова этого языка и только их.

Следующая теорема устанавливает связь между автоматными грамматиками и недетерминированными КА.

Теорема 6.1. *Для любого А-языка существует распознающий его НКА.*

Доказательство. Пусть $\Gamma = \langle V, W, J, R\rangle$ – А-грамматика. Нужно доказать, что существует НКА, который допускает слова языка $L(\Gamma)$ и только их. Построим НКА следующим образом. Определим

- множество состояний $Q = W \cup \{q_0\}$, где $q_0 \notin V \cup W$;
- начальное состояние $q_1 = J$;
- множество заключительных состояний $Q_0 = \{q_0\}$;
- если в R существует правило $A \to aB$, то $B \in \delta(A, a)$;
- если в R существует правило $A \to a$, то $q_0 \in \delta(A, a)$.

Теперь покажем, что данный НКА распознает язык $L(\Gamma)$. Для этого нужно доказать, что:

1. если $\alpha \in L$, то $q_0 \in \delta^*(J, \alpha)$, и наоборот

2. если $q_0 \in \delta^*(J, \alpha)$, то $\alpha \in L$.

Доказательства обоих утверждений схожи между собой. Докажем (1). Пусть $\alpha = \langle a_{i_1} a_{i_2} \dots a_{i_k} \rangle \in L$, то есть, по определению языка, в грамматике Γ существует вывод $J \models_\Gamma \alpha$:

$$J \to a_{i_1} A_1 \to a_{i_1} a_{i_2} A_2 \to \dots \to a_{i_1} \dots a_{i_{k-1}} A_{k-1} \to a_{i_1} \dots a_{i_k}.$$

Значит, Γ содержит правила

$$J \to a_{i_1} A_1,\ A_1 \to a_{i_2} A_2,\ \dots,\ A_{k-2} \to a_{i_{k-1}} A_{k-1},\ A_{k-1} \to a_{i_k}.$$

Тогда, по определению данного НКА имеем:

$$A_1 \in \delta(J, a_{i_1}),\ \dots,\ A_{k-1} \in \delta(A_{k-2}, a_{i_{k-1}}),\ q_0 \in \delta(A_{k-1}, a_{i_k}),$$

откуда $q_0 \in \delta^*(J, \alpha)$. ∎

Пример. Пусть

$$\Gamma = \langle \{a, b\}, \{J, A, B\}, J, \{J \to aJ|bJ|aA,\ A \to bB,\ B \to a\} \rangle .$$

Определим НКА с функцией перехода:

	J	A	B	J_1
a	J, A		J_1	
b	J	B		

Грамматика Γ и построенный НКА распознают язык

$$L = \{\alpha aba, \alpha \in V^*\},$$

то есть множество всех слов, заканчивающихся на $'aba'$.

Глава 7

Алгоритмы

7.1. Понятие алгоритма

Часто решение какой-либо задачи наводит на мысль, что полученный способ решения можно распространить и на другие задачи, похожие на только что решенную. Например, при решении *одной конкретной* системы линейных уравнений путем складывания уравнений, умноженных на правильно подобранные коэффициенты, можно прийти к выводу, что действуя аналогичным образом, то есть по строго определенным правилам, можно решить *любую* систему линейных уравнений. Система таких правил называется *алгоритмом*. Более точно, алгоритм – это строгое предписание, задающее последовательность некоторых элементарных операций, выполненных в определенном порядке для решения задач одного типа.

Заметим, что такое интуитивно понятное определение алгоритма не является формальным. В самом деле, что такое «элементарная операция»? И какие задачи считать задачами «одного типа»? Все эти детали требуют уточнения, если мы хотим применять к алгоритмам строгие методы исследования и доказательства. Формализация понятия алгоритма требуется для сравнения трудоемкости алгоритмов, их классификации, доказательства корректности и т. д.

К настоящему времени предложен ряд формальных моделей алгоритма. Курт Гёдель определил алгоритм как последовательность правил построения сложных математических функций из более простых. Алонсо Чёрч использовал формализм, называемый λ-исчислением. Алан Тьюринг предложил виртуальное автоматическое устройство, которое сейчас называется машиной Тьюринга, и определил алгоритм как программу для этой машины. А.А.Марков определил алгоритм как конечный набор правил подстановок цепочек символов и т. д.

В дальнейшем выяснилось, что все известные алгоритмы могут быть представлены алгоритмами, определенными в рамках этих формальных моделей. Мало того, перечисленные формальные определения алгоритмов являются эквивалентными: любой класс проблем, которые можно решить с помощью алгоритмов одного типа, решается также алгоритмами другого типа!

На основании этих результатов исследователи пришли к выводу, что *определение алгоритма в интуитивном смысле и его формальные определения эквивалентны.* Это положение является гипотезой, которую невозможно доказать строго, хотя бы потому что определение алгоритма в интуитивном смысле не является формальным.

Исторически Алонсо Чёрч первый предложил отождествлять понятие алгоритма с одним из его эквивалентных точных определений. Независимо от него Алан Тьюринг высказал предположение, что любой алгоритм в интуитивном смысле может быть представлен Машиной Тьюринга. Это предположение, известное как гипотеза Чёрча-Тьюринга, имеет важное практическое значение: поскольку каждый компьютер, независимо от его мощности, способен моделировать машину Тьюринга, а значит и алгоритм в любом другом понимании, все компьютеры эквивалентны с точки зрения принципиальной возможности алгоритмического решения проблем.

7.2. Машина Тьюринга

Алан Тьюринг определил алгоритм как механический процесс обработки информации, введя в рассмотрение гипотетическое автоматическое устройство, которое интуитивно моделирует действия чаловека, решающего задачу. Тьюринг рассуждал примерно так. Человек имеет конечную память, то есть представляет собой систему с конечным числом состояний. Исходная информация к алгоритму поставляется в виде цепочек символов – слов из конечного словаря, записанных, например, на бумаге. Выполняя алгоритм, человек использует дополнительную внешнюю память (например, листы бумаги), для записи промежуточной информации, причем запись ведется последовательно, символ за символом. При решении задачи человек может возвращаться к ранее записанной информации, стирать некоторую информацию и т. д. Таким образом, элементарными операциями при выполнении алгоритма можно считать запись и стирание символа, а также перенесение внимания с одного участка записи на другой.

Схематично предложенная Тьюрингом модель устройства, решающего задачу, изображена на рис. 7.1. Машина Тьюринга состоит из бесконечной ленты с записанными на ней символами из некоторого конечного алфавита и автоматического устройства – головки чтения-записи, способной считывать символы с ленты, записывать на нее новые символы и перемещаться вдоль ленты в обоих направлениях. Автоматическое устройство может находиться в одном из состояний, число которых конечно. В отличие от конечного автомата машина

Рис. 7.1. Машина Тьюринга

Тьюринга использует практически неограниченную память, записывая необходимую информацию на ленту.

Формально, *детерминированная машина Тьюринга* (ДТМ) есть шестёрка

$$T = \langle Q, A, \delta, \Lambda, q_1, q_0 \rangle ,$$

где

Q	–	непустое конечное множество состояний,
A	–	непустой конечный алфавит,
Λ	–	пустой символ, $\Lambda \notin A$,
δ	–	функция переходов, $\delta : Q \times A \to Q \times A \times \{R, L, S\}$
$q_1 \in Q$	–	начальное состояние,
$q_0 \in Q$	–	конечное состояние.

Машина Тьюринга работает по тактам. В начале работы машина находится в состоянии q_1, а ее головка обозревает некоторый символ на ленте. На каждом такте машина читает символ из обозреваемой головкой ячейки ленты и, в зависимости от текущего состояния и прочитанного символа, выполняет следующие действия:

1. изменяет или не изменяет свое состояние;

2. записывает символ в обозреваемую ячейку ленты, который может совпадать с только что прочитанным;

3. смещает головку влево или вправо на одну позицию относительно ленты или оставляет ее на прежнем месте.

Действие машины на каждом такте однозначно определяется ее текущим состоянием и читаемым символом на ленте. Для записи одного действия используется команда вида

$$q\,a \to q'\,a'\,d$$

или без стрелки

$$q\,a\,q'\,a'\,d$$

где через q и a обозначены состояние машины и читаемый символ, а через q', a' и d – новое состояние машины, записанный на ленту новый символ и направление движения головки. Список всевозможных команд машины Тьюринга, описывающих ее действия для всех состояний и входных символов, называется ее *программой*.

Рассмотрим пример. Следующая машина Тьюринга T обрабатывает строку, содержащую открывающие и закрывающие скобки, и проверяет их взаимный баланс. Баланс скобочного выражения определяется рекурсивно:

1. пустое слово является сбалансированным;
2. если α – сбалансированное слово, то слова $()\alpha$, (α) и $\alpha()$ тоже сбалансированные.

Например, слово $(())()$ является сбалансированным, а слово $(()$ – нет. Если записанное на ленте слово сбалансировано, то машина записывает на ленте символ (и останавливается. В противном случае машина также останавливается и на ленте остается символ).

Машина Тьюринга T определяется следующим образом:

$$T = \langle \{q_0, \ldots, q_4\}, \{(,), *\}, \delta, \Lambda, q_1, q_0 \rangle\,.$$

Программа δ состоит из следующих команд:

$$\begin{array}{ll}
q_1\,(\,q_1\,(\,R & q_3\,(\,q_4\,\Lambda\,L \\
q_1\,)\,q_2 * L & q_3 * q_3\,\Lambda\,L \\
q_1 * q_1 * R & q_3\,\Lambda\,q_0\,(\,S \\
q_1\,\Lambda\,q_3\,\Lambda\,L & q_4\,(\,q_4\,\Lambda\,L \\
q_2\,(\,q_1 * R & q_4 * q_4\,\Lambda\,L \\
q_2 * q_2 * L & q_4\,\Lambda\,q_0\,)\,L \\
q_2\,\Lambda\,q_0\,)\,S &
\end{array}$$

Часто программу записывают в виде таблицы:

	q_1	q_2	q_3	q_4
(	$q_1\,(\,R$	$q_1 * R$	$q_4\,\Lambda\,L$	$q_4\,\Lambda\,L$
)	$q_2 * L$			
$*$	$q_1 * R$	$q_2 * L$	$q_3\,\Lambda\,L$	$q_4\,\Lambda\,L$
Λ	$q_3\,\Lambda\,L$	$q_0\,)\,S$	$q_0\,(\,S$	$q_0\,)\,S$

Определение 7.12. Говорят, что машина Тьюринга T *применима* к слову на ленте, если после выполнения конечного числа шагов она переходит в конечное состояние q_0 и останавливается. Если в процессе обработки данного слова машина не переходит в конечное состояние и не останавливается, то говорят, что машина T *неприменима* к данному слову.

В данном примере машина T применима ко всем скобочным выражениям, однако ее легко модифицировать таким образом, чтобы она стала неприменимой к несбалансированным выражениям. Предлагаем читателю проделать это самостоятельно в качестве упражнения.

7.3. Функции, вычислимые по Тьюрингу

Функция вида $f : \mathbb{N}^n \to \mathbb{N}$ называется *арифметической функцией* n аргументов.

Обозначим через $\nu(a) \in A^*$ представление натурального числа a в алфавите V. Говорят, что машина Тьюринга $T = \langle Q, A, \delta, \Lambda, q_1, q_0\rangle$ *вычисляет* функцию f, если выполняются два условия:

1. если $f(a_1, \ldots, a_n) = b$, то машина Тьюринга T применима к слову $\nu(a_1), \ldots, \nu(a_n)$ и в результате его обработки оставляет на ленте слово $\nu(b)$;

2. если значение $f(a_1, \ldots, a_n)$ не определено, то машина T неприменима к слову $\nu(a_1), \ldots, \nu(a_n)$.

Если для функции f существует вычисляющая ее машина Тьюринга, то функция f называется *вычислимой по Тьюрингу.*

Пример. Докажем, что функция $f(n, m) = n + m$, $n, m \in \mathbb{N}$, вычислима по Тьюрингу. Для простоты построения соответствующей

машины Тьюринга будем записывать натуральные аргументы функции в *унарной* системе счисления, в которой число $n \in \mathbb{N}$ представляется в виде слова $\nu(n)$ в алфавите $A = \{1\}$:

$$\nu(n) = 1^n = \underbrace{1\,1\,\ldots\,1}_{n \text{ раз}}$$

Записанные на ленте аргументы будем разделять пустым символом Λ. Таким образом, требуется построить машину Тьюринга T, которая переводит строку $1^n\,\Lambda\,1^m$ в строку 1^{n+m}.

Программа для машины T выглядит следующим образом:

$$\begin{array}{c} q_1\,1\,q_2\,\Lambda\,R \\ q_2\,1\,q_2\,1\,R \\ q_2\,\Lambda\,q_0\,1\,S \end{array}$$

Машина T стирает первую встреченную единицу и начинает движение по ленте вправо, пока не встретит пустой символ, разделяющий слова 1^n и 1^m. Пустой символ заменяется на 1, после чего машина останавливается, оставляя на ленте слово 1^{n+m}.

Обозначим через $\mathcal{T}$ множество всех арифметических функций, вычислимых по Тьюрингу, а через $\mathcal{J}$ – множество всех арифметических функций, для которых существует алгоритм в интуитивном смысле, который ее вычисляет. Очевидно, что $\mathcal{T} \subseteq \mathcal{J}$, так как в качестве вычисляющего функцию алгоритма можно использовать машину Тьюринга.

Обратное вхождение $\mathcal{J} \subseteq \mathcal{T}$ нельзя ни доказать, ни опровергнуть, поскольку понятие алгоритма в интуитивном смысле не является формальным. Однако до сих пор не найдено ни одной арифметической функции, для которой не удалось построить вычисляющую ее машину Тьюринга. Это дало основание Алонзо Чёрчу и Алану Тьюрингу сформулировать в середине 1930-х годов гипотезу, известную как *тезис Чёрча-Тьюринга*, утверждающую, что $\mathcal{J} = \mathcal{T}$, и означающую, по сути, что машина Тьюринга представляет собой ту искомую модель, которая адекватно формализует понятие алгоритма в интуитивном смысле.

7.4. Абстрактные задачи и языки

Определение 7.13. *Абстрактной задачей* называется произвольное бинарное отношение $Q \subseteq I \times S$ между множеством I условий и множеством S решений.

Например, в задаче поиска гамильтонова цикла в графе условием является граф, а решением – последовательность вершин, задающих гамильтонов цикл.

Бинарное отношение $i \times S'$, где $i \in I$ и $S' \subseteq S$, называется *индивидуальной задачей*. Для задачи поиска гамильтонова цикла i это конкретный граф, а S' – множество всех гамильтоновых циклов этого графа.

Пусть исходные данные абстрактной задачи Q представлены словами в алфавите A. Задача Q называется *задачей распознавания*, если ее решением является один из ответов *да* или *нет*. Таким образом, решение задачи распознавания Q сводится к вычислению предиката $Q : I \to \mathbb{B}$.

Говорят, что ДТМ T *вычисляет* предикат $Q(x)$, $x \in A^*$, если выполняются два условия:

1. если $x \in I$, то машина T применима к слову x и в результате его обработки на ленте остается один из символов – 0 или 1;
2. если $x \notin I$, то машина T неприменима к слову x.

Определение 7.14. Пусть ДМТ T вычисляет предикат $Q(x)$, где $x \in A^*$. Будем говорить, что машина T *допускает* слово $x \in A^*$, если T применима к x и $Q(x) = 1$. Если T применима к x и $Q(x) = 0$, то говорят, что T *отвергает* слово x. Говорят также, что машина T *распознает язык* $L \subseteq A^*$, если T допускает все слова из L и отвергает все остальные слова. (Заметим, что распознающая ДМТ применима к любому слову $x \in A^*$, то есть всегда останавливается.)

Таким образом, каждой задаче распознавания $Q : I \to \mathbb{B}$ соответствует некоторый язык L, а решение задач в привычном смысле сводится к распознаванию языков. Разработка алгоритма решения задачи Q означает построение такой ДМТ, которая за конечное число шагов сможет определить принадлежность или непринадлежность любого входного слова языку L.

7.5. Алгоритмически неразрешимые проблемы

Долгое время считалось, что неразрешимых проблем не существует, то есть для любой четко сформулированной проблемы всегда можно построить алгоритм ее решения. В 1931 году немецкий математик Курт Гёдель впервые сформулировал неразрешимую проблему и показал,

что решающий ее алгоритм *никогда* не может быть найден. В дальнейшем оказалось, что таких проблем много. С некоторыми оговорками можно даже утверждать, что неразрешимых проблем бесконечно больше, чем разрешимых ([**?**], стр. 322).

В качестве примера неразрешимой проблемы рассмотрим так называемую проблему *самоприменимости*. Заметим, что программа машины Тьюринга представляет собой строку символов из некоторого алфавита. Для удобства формулировки проблемы заменим все символы программы комбинациями только двух символов – 0 и 1 (например, используя кодировку ASCII). Двоичное представление машины Тьюрига M назовем ее кодом и обозначим через $N(M)$.

Теперь условимся рассматривать только машины Тьюринга, входной алфавит которых состоит из двух символов – 0 и 1. Если машина M применима к слову $N(M)$, то она называется *самоприменимой*, в противном случае – *несамоприменимой*. Примером самоприменимой машины Тьюринга является машина, все команды которой содержат в правой части заключительное состояние, а примером несамоприменимой – машина, ни одна команда которой не содержит заключительного состояния.

Проблема *самоприменимости* состоит в определении по коду $N(M)$, является машина Тьюринга M самоприменимой или нет. Другими словами, требуется построить машину Тьюринга S, которая распознает язык $\mathcal{L}_S$, включающий только коды самоприменимых машин. Получив в качестве входной строки на ленте код какой-либо самоприменимой машины Тьюринга, машина S должна перейти в заключительное состояние и записать в текущей позиции на ленте символ 1. В противном случае машина S также должна перейти в заключительное состояние и записать на ленте символ 0.

Теорема 7.2. *Проблема самоприменимости является алгоритмически неразрешимой, то есть не существует машины Тьюринга, распознающей язык* $\mathcal{L}_S$.

Доказательство. Предположим, что машина Тьюринга S, распознающая язык $\mathcal{L}_S$, существует. На основе машины S построим машину S^* по следующим правилам:

1. все команды машины S полагаются командами машины S^*;
2. состояние q_0, которое является конечным в машине S, не является конечным в машине S^*;

3. в машину S^* добавляется новое состояние q_0^*, которое полагается конечным;

4. в программу машины S^* добавляются две новые команды:

$$q_0 0 \to q_0^* 1 S \text{ и } q_0 1 \to q_0 1 S.$$

Проанализируем работу машины S^*. Получив на входе код *несамоприменимой* машины, машина S^* выполнит все команды машины S, перейдет в состояние q_0 и запишет на ленте символ 0. После этого, в соответствии с новой добавленной командой, она перейдет в конечное состояние q_0^*, запишет на ленте символ 1 и остановится.

Если на входе окажется код *самоприменимой* машины, то машина S^* навсегда останется в состоянии q_0, то есть зациклится!

Таким образом, машина Тьюринга S^*

1. применима к кодам *всех* несамоприменимых машин и

2. неприменима к кодам *всех* самоприменимых машин,

то есть не может быть ни самоприменимой, ни несамоприменимой. Из полученного противоречия следует, что машина Тьюринга S^* не может существовать, а значит не может существовать и машина S. ∎

7.6. NP-полные задачи

Основными критериями оценки качества алгоритма служат время и память, затраченные для решения задачи. Свяжем с произвольной задачей некоторое число, называемое ее *размером*. Например, для задачи коммивояжера это число городов, для минимизации булевой функции – число ее аргументов и т.д.

Временной сложностью алгоритма называется зависимость времени выполнения алгоритма от размера задачи. (формальное определение мы дадим ниже с помощью машин Тьюринга). Поведение этой сложности в пределе при увеличении размера задачи называется *асимптотической временной сложностью*. (Аналогично определяется ёмкостная и асимптотическая ёмкостная сложность – как затраты памяти, необходимые для работы алгоритма.)

Для записи асимптотической сложности существует специальное обозначение – так называемая O-символика (или *асимптотическая нотация*).

Определение 7.15. Пусть g и f – неотрицательные функции, определенные на множестве целых неотрицательных чисел. Запись $g(n) = O(f(n))$ означает, что существуют константы $C, N > 0$, для которых соотношение $g(n) \leq Cf(n)$ выполняется для всех $n \geq N$.

Пример. Покажем, что $an + b\log n + c = O(n)$, где $a, b, c > 0$. Действительно, нетрудно заметить, что неравенство

$$an + b\log n + c \leq Cn$$

верно для $C = a + b + c$ и всех $n \geq 1$.

Обозначим через $T(n)$ время выполнения алгоритма. Что означает запись $T(n) = O(f(n))$? Во-первых, она дает краткую и удобную характеристику поведения этого алгоритма при достаточно больших значениях n. Кроме того, если рассматривать $O(f(n))$ как *класс функций* с одинаковой асимптотической оценкой, то все алгоритмы также могут быть разделены на классы в соответствии с их асимптотической сложностью.

7.6.1. Полиномиальные алгоритмы

Мы уже знаем, что существуют задачи, для которых не существует алгоритма решения. Однако на практике задача не всегда может быть решена, даже если алгоритм ее решения существует.

Рассмотрим пять классов алгоритмов $A_1, \ldots, A_5$, имеющих временную асимптотическую сложность, соответственно, n, $n\log n$, n^2, n^3 и 2^n. Предположим, что алгоритм A_1 сможет решить за 1 секунду задачу размера 1000. Следующая таблица содержит максимальный размер задач, решаемых алгоритмами из указанных классов за различные интервалы времени.

Класс алгоритма	Временная сложность	1 сек	1 мин	1 час
A_1	n	1000	6×10^5	3.6×10^6
A_2	$n\log n$	140	4893	2×10^5
A_3	n^2	31	244	1897
A_4	n^3	10	39	153
A_5	2^n	9	15	21

Рассмотрим еще одну любопытную закономерность. Пусть N_i означает максимальный размер задачи, решаемой i-м алгоритмом за один

час с помощью нынешнего поколения процессоров. Следующая таблица демонстрирует, на сколько возрастает размер решаемой задачи при многократном ускорении процессора для алгоритмов различной сложности.

Временная сложность	Современный процессор	В 100 раз мощнее	В 1000 раз мощнее
$O(n)$	N_1	$100\,N_1$	$1000\,N_1$
$O(n^2)$	N_2	$10\,N_2$	$31.6\,N_2$
$O(n^3)$	N_3	$4.64\,N_3$	$10\,N_3$
$O(n^5)$	N_4	$2.5\,N_4$	$4\,N_4$
$O(2^n)$	N_5	$N_5 + 7$	$N_5 + 10$

Приведенные здесь оценки заставляют задуматься. Действительно, алгоритмы с асимптотической временной оценкой сложности, ограниченной сверху полиномом (произвольной степени!), имеют некое принципиальное преимущество перед алгоритмами с экспоненциальной сложностью (заметим, что $2^n < n^{10}$ до $n \leq 59$). Фактически, разница между полиномиальной сложностью алгоритма и сложностью более высокого порядка – есть граница между тем, что можно решить с помощью компьютера, а что нельзя.

Таким образом, принципиальным является вопрос, все ли разрешимые (по Тьюрингу) задачи, могут быть решены за полиномиальное время. Далее мы рассмотрим класс задач, называемый $\mathcal{NP}$, для которых, с одной стороны, не найдены полиномиальные алгоритмы решения, а с другой стороны – *не доказано*, что таких алгоритмов нет. В классе $\mathcal{NP}$ выделен класс $\mathcal{NP}$-полных задач, к которым можно свести любую решаемую проблему. Это означает, что если будет найден полиномиальный алгоритм решения хотя бы одной $\mathcal{NP}$-полной задачи, то *любая* задача автоматически «получит» полиномиальный алгоритм решения.

Для формализации указанных понятий нам потребуется ряд новых определений.

7.6.2. Задачи распознавания и k-ленточные ДМТ

В теории $\mathcal{NP}$-полных задач рассматриваются только задачи распознавания, то есть задачи, решением которых является один из ответов – да или нет. Для классификации задачи по критерию временной сложности необходимо преобразовать ее к соответствующей задаче распознавания. Например, для задачи коммивояжера (ЗК) это следующая задача

(обозначим ее через Q_1): *по заданному числу k и графу G установить, существует ли в графе G гамильтонов цикл, длина которого не превосходит k.*

Пусть пара $i = \langle G, k\rangle$ является условием задачи распознавания Q_1. Тогда $Q_1(i) = 1$, если такой цикл существует, и $Q_1(i) = 0$, если такого цикла в графе нет. Задача распознавания Q_1 обладает одним интересным свойством: если ЗК имеет полиномиальную временную сложность, то и задача Q_1 также имеет полиномиальную временную сложность. Действительно, если можно за полиномиальное время найти кратчайший гамильтонов цикл в графе G, то сравнив его длину с k, мы сможем ответить, существует ли в графе гамильтонов цикл не длиннее k.

Напомним, что решение задач в привычном смысле сводится к распознаванию языков с помощью детерминированных машин Тьюринга (ДМТ). Ранее мы рассматривали ДМТ с одной лентой. Обобщением одноленточной ДМТ является ДМТ, имеющая $k \geq 1$ лент, на каждой из которых имеется по одной считывающей головке. В начальный момент времени входное слово записано на первой ленте, остальные ленты имеют по одной пустой ячейке. Если машина применима к входному слову, то результат вычисления записывается на первой ленте. Дадим формальное определение.

Определение 7.16. *k-ленточной детерминированной машиной Тьюринга*, обозначается k-ДМТ, называется шестерка следующего вида

$$M = \langle Q, A, \delta, \mathbf{\Lambda}, q_1, q_0\rangle,$$

где символы Q, A, $\mathbf{\Lambda}$, q_1 и q_0 определяются так же как и для одноленточных ДМТ, а функция перехода

$$\delta : Q \times A^k \to Q \times (A \times \{L, R, S\})^k$$

по текущему состоянию и набору текущих символов на k лентах задает новое состояние, новые k символов и направления движения каждой из k считывающих головок.

Дадим формальное определение временной сложности алгоритма.

Определение 7.17. Арифметическая функция $T_M(n)$ называется *временной сложностью* вычисления k-ДМТ M, если получив на вход слово длины n машина M остановится не более чем через $T_M(n)$ шагов.

Если на каком-либо слове длины n машина M не остановится, то для этого n функция $T_M(n)$ не определена[1].

Можно показать, что временная сложность k-ДМТ и временная сложность машины с *произвольным доступом к памяти* (*random access memory* – RAM), которая является моделью современного компьютера, совпадают с точностью до полинома: если RAM допускает некоторое слово a длины n за время $T(n)$, то существует константа $C > 0$ и k-ДМТ, которая допускает a за время не более чем $C\,T^2(n)$.

7.6.3. Класс $\mathcal{P}$ и полиномиальная сводимость

Определение 7.18. Говорят, что язык L распознается за *полиномиальное время*, если существуют k-ДМТ M и полином $p(n)$, такие что $T_M(n) = O(p(n))$. Множество всех языков, распознаваемых за полиномиальное время, образует класс $\mathcal{P}$.

Другими словами, класс $\mathcal{P}$ включает только те задачи распознавания, для которых существует полиномиальный алгоритм решения.

Примерами полиномиально разрешимых задач являются поиск кратчайшего пути между любыми двумя вершинами графа, нахождение минимального потока в сети, перемножение матриц, все виды сортировок, различные геометрические задачи и т.д. – одним словом, это все задачи, которые можно решить с помощью компьютера независимо от их размера.

Одним из способов поиска полиномиального решения является сведение данной задачи к такой задаче, для которой полиномиальный алгоритм известен. Говорят, что задача A *полиномиально сводится* к задаче B, если существует полиномиальный алгоритм, который преобразует входные данные задачи A в такие входные данные задачи B, что обе задачи на этих входах имеют одинаковое решение (т.е. *да* или *нет*). Проще говоря, свести (полиномиально) задачу A к задаче B означает указать полиномиальный алгоритм решения задачи A с помощью полиномиального алгоритма решения задачи B.

Формальное определение сводимости такое.

Определение 7.19. Язык $L_1 \subseteq A_1^*$ *полиномиально сводится* к языку $L_2 \subseteq A_2^*$, если существует k-ДМТ M такая, что $T_M(n) = O(p(n))$ и

[1] Арифметическая функция $S_M(n)$ называется *емкостной сложностью*, если при обработке слова x СГ отступят не более чем на $S_M(n)$ ячеек от начальной позиции. Если на какой-либо ленте СГ движется вправо или влево неограниченно долго, то функция $S_M(n)$ не определена ни для какого n.

которая преобразует любое слово $x_1 \in A_1^*$ в такое слово $x_2 \in A_2^*$, что $(x_1 \in L_1) \Leftrightarrow (x_2 \in L_2)$.

Пример. Рассмотрим две задачи.

Задача 7.1 (РАЗДЕЛИМОСТЬ МНОЖЕСТВ). *Пусть $A = \{a_1, \ldots, a_n\}$ и $B = \{b_1, \ldots b_n\}$ – два множества действительных неотрицательных чисел. Необходимо определить, пересекаются они или нет.*

Задача 7.2 (ДИАМЕТР МНОЖЕСТВА ТОЧЕК). *Задано множество S из n точек на плоскости и действительное число $d \geq 0$. Необходимо определить, существует ли в S пара точек, расстояние между которыми не меньше d.*

Выполним полиномиальное сведение задачи 7.1 к задаче 7.2 (то есть решим задачу 7.1 с помощью задачи 7.2). Сначала нормируем числа из A и B разделив их на максимальное число среди всех чисел обоих множеств. Получим числа, лежащие в интервале от 0 до 1. Теперь поставим в соответствие множеству A точки, лежащие на пересечении единичной окружности с прямыми $y_i = a_i x$. Указанные прямые пересекают окружность в двух точках, мы же выбираем только те из них, которые лежат в первом октанте. Аналогично, поставим в соответствие множеству B точки, лежащие в третьем октанте и на пересечении единичной окружности и прямых $y_i = b_i x$.

Очевидно, максимальное расстояние между построенными точками в точности равно $d = 2$ тогда и только тогда, когда существуют равные числа $a_i \in A$ и $b_j \in B$. Построение множеств точек выполнено за время $O(n)$, поэтому мы можем говорить о полиномиальном сведении задачи 7.1 к задаче 7.2.

Теорема 7.3. *Если язык L_1 полиномиально сводится к языку L_2 и $L_2 \in \mathcal{P}$, то $L_1 \in \mathcal{P}$.*

Доказательство. Из условия теоремы следует, что существует k-ДМТ M_2, распознающая язык L_2 за время $T_2(n) = O(p_2(n))$. Кроме того, существует машина M_{12}, преобразующая за полиномиальное время $T_{12}(n) = O(p_{12}(n))$ любое слово x_1 из алфавита языка L_1 в слово x_2 из алфавита языка L_2. Значит, объединив команды обеих машин, можно построить новую машину M_1, которая принимает на вход слово x_1, затем преобразует его в слово x_2, и возвращает 1 тогда и только

тогда, когда $x_2 \in L_2$. Получаем, что временная сложность машины M_1 ограничена сверху полиномом $p_1(n)$:

$$T_1(n) = T_{12}(n) + T_2(n) = O(p_{12}(n) + p_2(n)) = O(p_1(n)).$$

■

7.6.4. Недетерминированные машины Тьюринга

Мы уже обсуждали недетерминированные конечные автоматы (КА), каждый шаг которых мог перевести КА в одно из нескольких различных состояний. По аналогии, для каждой конфигурации недетерминированной машины Тьюринга (НМТ) существует конечное число вариантов следующих конфигураций, из которых выбирается какой-то один. НМТ применима к входному слову x, если по крайней мере одна последовательность конфигураций для слова x приводит к конфигурации, содержащей конечное состояние. Дадим строгое определение.

Определение 7.20. *k-ленточной недетерминированной машиной Тьюринга*, обозначается k-НМТ, называется шестёрка $\langle Q, A, \delta, \Lambda, q_1, q_0 \rangle$, элементы которой определяются так же, как для k-ДМТ, за исключением функции перехода

$$\delta : Q \times A^k \to 2^{Q \times (A \times \{L,S,R\})^k}$$

которая по текущему состоянию и списку из k символов на лентах выдает конечное множество вариантов следующего шага.

Определение 7.21. Говорят, что k-НМТ имеет *временную сложность* $T(n)$, если для произвольного входного слова длины n найдется последовательность, состоящая не более чем из $T(n)$ конфигураций, переводящая машину в состояние q_0.

Понятие *распознаваемого языка* для НМТ определяется так же, как для ДМТ.

Можно показать, что любой язык L_M, распознаваемый некоторой НМТ M, распознается также и некоторой ДМТ M', но временная сложность такой ДМТ может сильно возрасти[2].

[2]Доказательство проводится путем построения ДМТ M', моделирующей НМТ M перебором всех вариантов ее поведения в каждом из состояний.

Утверждение 7.5. *Если $T_M(n)$ есть временная сложность k-НМТ M, то существуют константа $C \geq 1$ и k-ДМТ M' такие, что $L_M = L_{M'}$ и $T_{M'}(n) = O(C^{T_M(n)})$.*

Таким образом, $O(C^{T_M(n)})$ есть *верхняя* оценка временной сложности решения задач с помощью k-ДМТ (а значит и с помощью обычного компьютера), для которой существует k-НМТ M с временной сложностью $T_M(n)$. Другими словами, *если существует полиномиальный недетерминированный алгоритм, то соответствующий ему детерминированный алгоритм может иметь не более чем экспоненциальную сложность*!

Следствие 7.1. *Проблема Q является алгоритмически неразрешимой тогда и только тогда, когда не сушествует k-НМТ, которая ее решает.*

Доказательство. Необходимость. От противного: если существует k-НМТ M, решающая задачу Q, то существует также некоторая k-ДМТ M', решающая ее, возможно, за более долгое время. Значит, Q алгоритмически разрешима. Противоречие.

Достаточность. Любая ДМТ есть НМТ, каждый следующий шаг которой определяется однозначно. Значит, если не существует НМТ, решающей задачу Q, то ее не решает и ни одна ДМТ. ∎

7.6.5. Класс $\mathcal{NP}$

Итак, мы разделили множество всех языков на два класса: класс языков, для которых не существует распознающей машины Тьюринга (ни ДМТ, ни НМТ), и класс языков, для которых существует ДМТ, распознающая язык за полиномиальное время. Промежуточным классом в данной иерархии является класс $\mathcal{NP}$.

Определение 7.22. Множество языков, распознаваемых некоторой k-НМТ за полиномиальное время, образует сложностный класс $\mathcal{NP}$.

Утверждение 7.6. $\mathcal{P} \subseteq \mathcal{NP}$.

Действительно, если $L \in \mathcal{P}$, то по определению существует ДМТ M, распознающая L за полиномиальное время, а значит существует и распознающая НМТ M', равная M.

Обратное включение $\mathcal{NP} \subseteq \mathcal{P}$ до сих пор не доказано: существует большое количество проблем, полиномиальный алгоритм решения которых не найден.

Таким образом, проблема $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ остается открытой. Поскольку непрерывные поиски полиномиальных алгоритмов для многих задач из класса $\mathcal{NP}$ до сих пор не принесли результатов, существуют веские основания полагать, что $\mathcal{P} \neq \mathcal{NP}$.

Возможные соотношения между классами $\mathcal{P}$ и $\mathcal{NP}$ представлены на рис. 7.2.

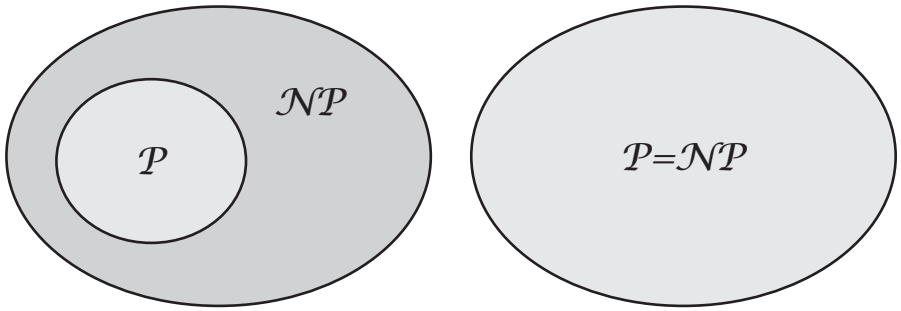


Рис. 7.2. Возможные соотношения между классами $\mathcal{P}$ и $\mathcal{NP}$

7.6.6. $\mathcal{NP}$-полные задачи

Наиболее убедительным аргументом в пользу того, что классы $\mathcal{P}$ и $\mathcal{NP}$ различны, является существование $\mathcal{NP}$-полных задач. Каждая $\mathcal{NP}$-полная задача обладает удивительным свойством: если она разрешима за полиномиальное время, то и *все* задачи из класса $\mathcal{NP}$ также разрешимы за полиномиальное время, то есть $\mathcal{P} = \mathcal{NP}$. Но несмотря на многолетние исследования, ни для одной $\mathcal{NP}$-полной задачи не найден полиномиальный алгоритм.

Определение 7.23. Абстрактная задача Q называется *$\mathcal{NP}$-полной*, если выполняются два условия:

1. $Q \in \mathcal{NP}$,
2. Любая задача из класса $\mathcal{NP}$ полиномиально сводится к Q.

Если для задачи Q выполняется только свойство (2), то она называется *$\mathcal{NP}$-трудной*.

Из определения $\mathcal{NP}$-полной задачи следует важное ее свойство.

Теорема 7.4. *Если некоторая $\mathcal{NP}$-полная задача разрешима за полиномиальное время, то $\mathcal{P} = \mathcal{NP}$. Другими словами, если в классе*

$\mathcal{NP}$ существует задача, неразрешимая за полиномиальное время, то и все $\mathcal{NP}$-полные задачи таковы.

Таким образом, гипотеза $\mathcal{P} \neq \mathcal{NP}$ означает, что $\mathcal{NP}$-полные задачи не могут быть решены за полиномиальное время. Большинство экспертов считают, что так оно и есть. На рис. 7.3 показано предполагаемое соотношение между классами $\mathcal{P}$, $\mathcal{NP}$ и $\mathcal{NP}$-полных задач.

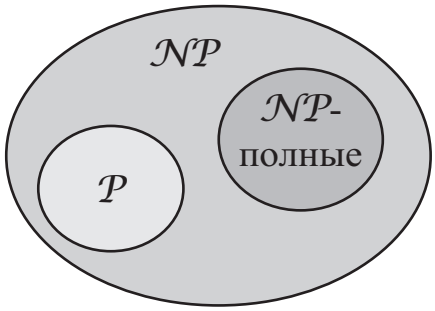


Рис. 7.3. Предполагаемое соотношение между классами $\mathcal{P}$, $\mathcal{NP}$ и классом $\mathcal{NP}$-полных задач

Первой обнаруженной $\mathcal{NP}$-полной задачей является задача выполнимости КНФ.

Задача 7.3 (ВЫПОЛНИМОСТЬ). *По заданной КНФ, реализующей булеву функцию $f(x_1, \ldots, x_n)$, установить, существует ли набор значений переменных $a_1, \ldots, a_n$ такой, что $f(a_1, \ldots, a_n) = 1$.*

Пример.

$$K_1 = (x_1 \vee \bar{x}_2)(x_2 \vee x_3 \vee \bar{x}_4)$$

принимает значение 1 при $x_i = 1$, $i = 1, \ldots, 4$, в то время как

$$K_2 = \bar{x}_1(x_1 \vee x_2)(x_1 \vee \bar{x}_2) \equiv 0.$$

В 1971 году С.Кук сформулировал и доказал следующую теорему:

Теорема 7.5 (Кук, 1971). *Задача выполнимости $\mathcal{NP}$-полна.*

Без доказательства.