ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
«ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

А.В. Воронин

ДИСКРЕТНАЯ МАТЕМАТИКА

Рекомендовано в качестве учебного пособия
Редакционно-издательским советом
Томского политехнического университета

Издательство
Томского политехнического университета
2009

УДК 519
ББК 22.1
В60

Воронин А.В.

В60 Дискретная математика: учебное пособие / А.В. Воронин. – Томск: Изд-во Томского политехнического университета, 2009. – 116 с.

Учебное пособие составлено на основе лекций, разработанных автором.

…

Пособие подготовлено на кафедре интегрированных компьютерных ситем управления и предназначено для студентов ИДО, обучающихся по специальности 220301 «Автоматизация технологических процессов и производств».

УДК 519
ББК 22.1

Рецензенты

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Термин «дискретная математика» начал входить в научный обиход на рубеже 50-х и 60-х гг. XX в. для обозначения ряда разделов математики, таких, как теория булевых функций, теория конечных автоматов, теория графов, теория кодирования и др., которые стали интенсивно развиваться в связи с необходимостью создания сложных управляющих систем и бурным прогрессом вычислительной техники.

Иногда в него вкладывают и более широкий смысл, определяя дискретную математику как область математики, занимающуюся изучением дискретных структур, которые возникают как в пределах самой математики, так и в её приложениях, т.е. включая в дискретную математику все математические дисциплины имеющие дело с дискретными множествами. В этом смысле к дискретной математике относят и теорию чисел, и всю конечную алгебру, и некоторые другие классические разделы математики.

Исторически дискретная математика значительно старше своей сестры – математики непрерывных величин, т.к. с момента своего зарождения математика являлась в основном дискретной, ее знания представляли собой набор конкретных правил и служили для решения практических задач. Таковой она в значительной степени и оставалась, пока в XVII веке запросы естествознания и техники не привели к созданию методов, позволяющих математически изучать движение, процессы изменения величин, преобразование геометрических фигур. С употребления переменных величин в аналитической геометрии и создания дифференциального и интегрального исчисления начинается период математики переменных величин. Великим открытиям XVII века является введенное Ньютоном и Лейбницем понятие бесконечно малой величины, создание основ анализа бесконечно малых (математического анализа), разработка понятий предела, производной, дифференциала, интеграла. От Ньютона математика пошла в основном по непрерывному пути, так как обслуживала нужды физики, которая изучала непрерывные процессы (движение планет, процессы в жидкостях и газах и т. д.).

Возрождение дискретной математики в форме работ по теории множеств, математической логике, теории графов, комбинаторике относится к середине XIX в. и было вызвано исследованиями в области электрических сетей, моделей кристаллов и структур молекул, хотя отдельные работы появлялись и ранее. Например, известное рассуждение Эйлера о Кенигсбергских мостах, считающееся началом теории графов, было опубликовано в 1736 г.

Однако наиболее интенсивное развитие всех разделов дискретной математики связано с возникновением кибернетики как науки об общих процессах управления в природе, технике и обществе, а также с появлением мощной вычислительной техники, способной эти процессы исследовать. Знание теории множеств, алгебры, математической логики и теории графов совершенно необходимо для четкой формулировки понятий и постановок различных прикладных задач, их формализации и компьютеризации, а также для усвоения и разработки современных информационных технологий. Понятия и методы теории алгоритмов и алгебры логики лежат в основе современной теории и практики программирования.

В отличие от традиционной математики (математического анализа, линейной алгебры и др.), методы и конструкции которой имеют в основном числовую интерпретацию, дискретная математика имеет дело с объектами нечисловой природы: множествами, логическими высказываниями, алгоритмами, графами. Благодаря этому обстоятельству дискретная математика впервые позволила распространить математические методы на сферы и задачи, которые ранее были далеки от математики. Примером могут служить методы моделирования различных социальных и экономических процессов.

Одной из особенностей дискретной математики является ее «разбросанность», в ней нет такого ядра, какое представляют в математике непрерывных величин разделы дифференциального и интегрального исчисления. Поэтому содержание конкретных курсов в сильной степени зависит от того, для студентов каких специальностей предназначается курс.

Данное пособие ориентировано на подготовку специалистов в области управления техническими объектами. Из множества разделов дискретной математики в него включены лишь наиболее употребительные в теории управления: теория множеств как базовый раздел всей дискретной математики, а также математическая логика, теория графов и некоторые вопросы теории конечных автоматов и теории кодирования.

Глава 1

МНОЖЕСТВА И ОТНОШЕНИЯ

Теория множеств является основой всего здания дискретной математики, так как определяет и упорядочивает круг объектов, с которыми работает дискретная математика.

1.1. Множества

1.1.1. Основные определения

Фундаментальным понятием теории множеств является понятие **множества**. Как всякому фундаментальному понятию, ему нельзя дать четкого определения через элементарные понятия.

Под множеством интуитивно понимают совокупность определенных, вполне различимых объектов, рассматриваемых как единое целое.

Отдельные объекты, из которых состоит множество, называются **элементами** множества.

Из определения следует, что элементы множества должны быть:

- вполне различимыми;
- иметь общее свойство.

Будем обозначать множества большими буквами латинского алфавита, а элементы – малыми буквами с индексами или без.

Отношение принадлежности элемента a множеству A обозначается $a \in A$.

Отношение непринадлежности элемента множеству обозначается как $a \notin A$.

Множество B называется **подмножеством** множества A, если всякий элемент B принадлежит A. Такое отношение обозначается как $B \subseteq A$. Если $B \subseteq A$ и $B \neq A$, то $B \subset A$. В этом случае говорят, что B есть **собственное** подмножество A.

Множествами являются, например:

A – множество студентов группы 8А03. Иванов $\in A$. Сидоров $\notin A$;

B – множество мужчин в группе 8А03, $B \subset A$;

C – множество студентов выше 174 см в группе 8А83, $C \subseteq A$.

Однако мы не будем считать множеством «множество капель в стакане воды» или «множество мыслей в голове». И не из-за их количества, а из-за того, что эти капли-мысли-элементы невозможно четко разделить, разложить по полочкам и разметить.

Множество может содержать **любое** число элементов, в том числе и ни одного элемента. Множество, не содержащее элементов, называется **пустым** и обозначается $\varnothing$. Пустое множество $\varnothing$ является подмножеством любого множества.

Множество, состоящее из конечного числа элементов, называется **конечным**, в противном случае **– бесконечным.** Например, множество натуральных чисел N, т. е. чисел 1, 2, 3, … – бесконечно. Число элементов в конечном множестве A называется его **мощностью** и обозначается $|A|$.

Для бесконечных множеств родоначальником теории множеств Георгом Кантором введены и рассмотрены два типа бесконечности.

Множества, равномощные множеству натуральных чисел N, называются **счетными.**

Множества, равномощные множеству вещественных чисел R, называются **континуальными.**

Разница между счетными и континуальными множествами в том, что между соседними элементами множества N, например числами 2 и 3, нельзя вставить какого-либо числа. А между любыми элементами множества R можно вставить сколько угодно чисел.

Если все рассматриваемые в ходе данного рассуждения множества являются подмножествами некоторого множества U, то такое множество называется **универсальным**.

Пример. Установить истинность или ложность следующих выражений:

1. $2 \in \{1,2,3,4,5\}$.
2. $\{2,6\} \subseteq \{1,2,3,4,5\}$.
3. $\{1,2,3\} \in \{1,2,3,\{1,2,3\}\}$.
4. $\{2,6\} \subseteq \{6,2\}$.
5. $\{2\} \subseteq \{1,2,3,4,5\}$.

Ответ. Выражения 1, 3, 4, 5 истинны. Выражение 2 ложно.

Пример. Справедливо ли равенство $\{\{1,2\},\{2,3\}\}=\{1,2,3\}$?

Ответ – нет; первое множество содержит два элемента, являющихся подмножествами; второе – три простых элемента.

Пример. Определить мощность множеств:

$A = \{1,2,3\}$, $|A| = 3$;

$B = \{1,2,3,\{1,2,3\}\}$, $|B| = 4$.

Основоположником теории множеств Г. Кантором были сформулированы несколько интуитивных принципов, играющих роль аксиом. Нас интересуют два таких принципа.

Принцип объемности. Множества A и B считаются равными $(A = B)$, если они состоят из одних и тех же элементов.

Чтобы сформулировать второй принцип, введем понятие «формы от x». Под формой от x будем понимать конечную последовательность, состоящую из слов и символа x, такую, что если каждое вхождение x в эту последовательность заменить одним и тем же именем некоторого предмета, то в результате получится истинное или ложное предложение. Например, формами от x являются следующие предложения: «x делится на 3», «$x^2 + 2x + 1 > x$», «x – валюта США». А такое предложение, как «существует такое x, что $x>0$», не является формой от x.

Принцип абстракции. Любая форма $P(x)$ определяет некоторое множество A, а именно множество тех и только тех элементов $a \in A$, для которых $P(a)$, – истинное предложение.

Для множества A, определяемого формой $P(x)$, принято обозначение $A = \{x / P(x)\}$ или $A = \{x : P(x)\}$. **Пример**: $A = \{x / x > 0\}$.

Поскольку $P(x)$ позволяет определить, принадлежит некоторый элемент данному множеству или нет, она иногда называется **распознающей процедурой**.

1.1.2. Способы задания множеств

Множество может быть задано следующими способами.

Перечислением, т. е. списком своих элементов. Перечислением можно задать лишь конечные множества. Например, множество студентов группы A={Иванов, Петров, Сидоров}; множество B устройств ПЭВМ B={процессорный блок, монитор, клавиатура}.

Описанием характеристических свойств, которыми должны обладать его элементы, т. е. некоторой распознающей процедурой. Например, множество A периферийных устройств ПЭВМ может быть определено A={x / x – периферийное устройство ПЭВМ}, или $B = \{x / x = 2n, n \in N\}$.

Порождающей процедурой, которая описывает способ получения элементов множества из уже имеющихся элементов либо других объектов. В таком случае элементами множества являются все объекты, которые могут быть получены с помощью такой процедуры.

Например, множество M целых чисел, являющихся степенями двойки, может быть представлено порождающей процедурой, заданной двумя рекуррентными (индуктивными) правилами:

а) $1 \in M$; б) если $m \in M$, то $2m \in M$.

Подобным же образом можно задать множество клеток шахматной доски, доступных коню за два хода, если в начальном состоянии он находится в клетке $c1$:

а) $m_1 = c1 \in M$; б) $m_i \in M$, если в m_i можно попасть из m_{i-1} ходом коня; $i \le 3$.

Порождающей процедурой удобно задавать элементы бесконечных множеств.

Пример. Задать различными способами множество N всех натуральных чисел.

1. Списком задать это множество нельзя.

2. Описанием характеристических свойств: $N = \{x / x$ – целое положительное число$\}$.

3. Порождающей процедурой: а) $1 \in N$; б) если $n \in N$, то $n + 1 \in N$.

Пример. Задать разными способами множество M всех положительных четных чисел ≤ 100.

1. Списком: $M = \{2, 4, 6, \ldots, 100\}$.

2. Описанием характеристических свойств:

$$M = \{n / n \in N, n / 2 \in N, n \le 100\}.$$

3. Порождающей процедурой: $2 \in M$; если $n \in N$, то $(n + 2) \in N$; $n \le 98$.

1.1.3. Диаграммы Эйлера – Венна

Диаграммы Эйлера – Венна – геометрические представления множеств [4, 5]. Построение диаграмм заключается в изображении большого прямоугольника, представляющего универсальное множество U, а внутри его – кругов или других замкнутых фигур, представляющих множества. Фигуры должны пересекаться в наиболее общем случае, требуемом в задаче. Точки внутри фигур обозначают элементы множеств. Пример диаграммы Эйлера – Венна для двух пересекающихся множеств – A и B – представлен на рис. 1.1.

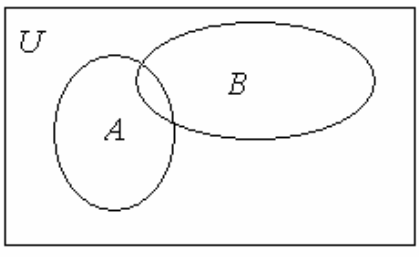


Рис. 1.1. Диаграмма Эйлера – Венна

1.1.4. Операции над множествами

Объединением множеств A и B (обозначается $A \cup B$) называется множество, состоящее из всех тех элементов, которые принадлежат хотя бы одному из множеств A или B: $A \cup B \quad \{x \mid x \in A \text{ или } x \in B\}$.

Пересечением множеств A и B (обозначается $A \cap B$) называется множество, состоящее из всех тех и только тех элементов, которые принадлежат и A, и B: $A \cap B \quad \{x \mid x \in A \text{ и } x \in B\}$.

Объединение и пересечение произвольной совокупности множеств определяются аналогично: $A \cup B \cup C$, $K \cap M \cap P$.

Разностью множеств A и B (обозначается $A \setminus B$) называется множество всех тех элементов A, которые не содержатся в B. Разность – операция строго двухместная и некоммутативная, в общем случае $A \setminus B \neq B \setminus A$, т. е. операнды нельзя менять местами.

Дополнением (до U) множества A (обозначается $\overline{A}$) называется множество всех элементов $\overline{A} = U \setminus A$, не принадлежащих A (но принадлежащих U).

Операции $\{\cup, \cap, -\}$ будем называть **булевыми операциями** над множествами.

Пример. Пусть U – множество сотрудников некоторой фирмы, A – множество сотрудников старше 30 лет, B – множество мужчин, C – множество программистов.

Тогда $\overline{B}$ – множество женщин; $\overline{A} \cap B \cap C$ – множество мужчин – программистов младше 30 лет; $A \cup (B \cap \overline{C})$ – множество сотрудников старше 30 лет или мужчин не программистов; $B \setminus C$ – множество мужчин, не являющихся программистами; $C \setminus B$ – множество программистов – женщин.

Пример. Пусть U={1,2,3,4}; A={1,3,4}; B={2,3}; C={1,4}. Найти $\overline{A} \cup \overline{B}$, $\overline{A \cap B}$, $A \cap \overline{B}$, $(\overline{B} \setminus A) \cup \overline{C}$.

Ответы:

$\overline{A} \cup \overline{B} = \{1,2,4\}$; $\overline{A \cap B} = \{1,2,4\}$;

$A \cap \overline{B} = \{1,4\}$; $(\overline{B} \setminus A) \cup \overline{C} = \{2,3\}$.

Пример. На диаграммах Эйлера – Венна заштриховать множества $(B \cap C) \setminus A$, $(A \cup B) \cap \overline{C})$, рассмотрев для каждого примера три варианта взаимного размещения множеств A, B, C.

Ответы:

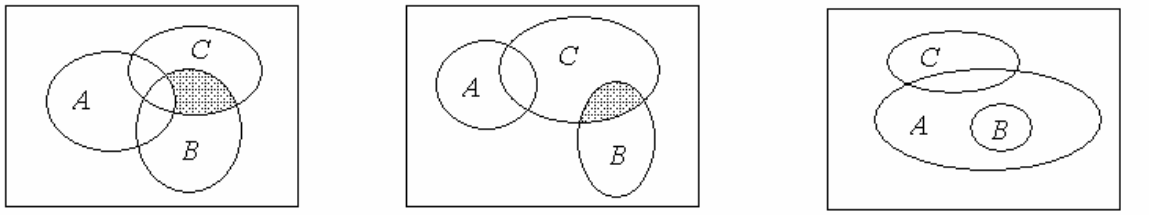


Рис. 1.2. Варианты диаграмм Эйлера – Венна для множества $(B \cap C) \setminus A$

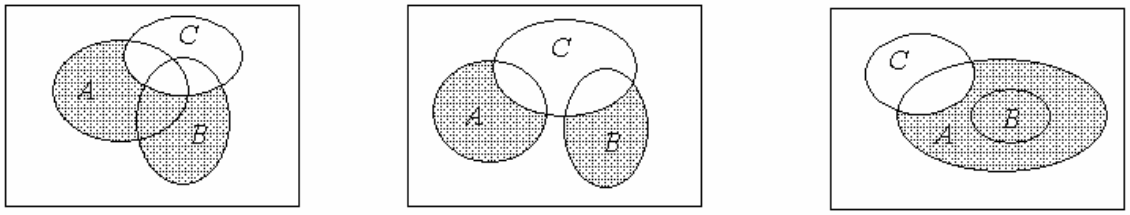


Рис. 1.3. Варианты диаграмм Эйлера – Венна для множества $(A \cup B) \cap \overline{C}$.

1.1.5. Свойства булевых операций над множествами

Коммутативность:
$A \cup B = B \cup A$;
$A \cap B = B \cap A$.
Ассоциативность:
$(A \cup B) \cup C = A \cup (B \cup C)$;
$(A \cap B) \cap C = A \cap (B \cap C)$.
Идемпотентность и свойства констант:
$A \cap A = A$; $A \cup A = A$;
$A \cap U = A$; $A \cup U = U$;
$A \cap \varnothing = \varnothing$; $A \cup \varnothing = A$.
Дистрибутивность:
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C))$;
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.
Теоремы А. де Моргана:
$\overline{A \cap B} = \overline{A} \cup \overline{B}$; $\overline{A \cup B} = \overline{A} \cap \overline{B}$.
Инволюция:
$\overline{\overline{A}} = A$.

Аналитическое доказательство справедливости приведенных формул может опираться на доказательство равенства двух множеств. Однако проще проиллюстрировать свойства булевых операций, используя

диаграммы Эйлера – Венна. Например, справедливость теорем де Моргана вполне очевидна прямо из рис. 1.4.

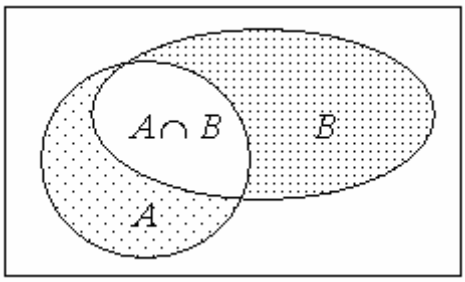


Рис. 1.4. Иллюстрация к теоремам де Моргана

1.2. Отношения

Между элементами множеств могут устанавливаться различные взаимосвязи, обычно называемые **отношениями**. Человек может соотноситься с профессией, зарплата – с должностью, наказание – с преступлением, оценка – со знанием. Для определения конкретного отношения надо определить два множества: множество (область) определения и множество (область) значений. Эти множества могут быть различными (как, например, отношение между множеством студентов группы и множеством полученных ими оценок) или одинаковыми (как, например, «быть братом» на множестве людей). Иногда в литературе для отношений, связывающих различные множества, используется термин «**соответствия**».

Наиболее изученными и чаще всего используемыми являются так называемые бинарные, или двухместные, отношения. Прежде чем их рассматривать, введем понятия упорядоченной пары и прямого произведения множеств.

Пусть a и b – элементы множеств M_1 и M_2 соответственно. Тогда через (a,b) будем обозначать упорядоченную пару, т. е. в общем случае $(a,b) \neq (b,a)$.

Пусть M_1 и M_2 – два множества. **Прямым** (декартовым) **произведением** двух множеств (M_1 и M_2) называется множество всех упорядоченных пар, в котором первый элемент каждой пары принадлежит M_1, а второй принадлежит M_2:

$$M_1 \times M_2 = \{(a,b) / a \in M_1, b \in M_2\}.$$

Пример. Классическим примером упорядоченных пар могут служить координаты точек на плоскости. Если координаты принадлежат

области вещественных чисел, то прямое произведение содержит бесконечное множество упорядоченных пар – координат всех точек плоскости. Если координаты определены на множестве целых чисел и рассматриваемая область плоскости ограничена, то прямое произведение будет содержать конечное множество упорядоченных пар.

Бинарным, или **двухместным,** отношением R называется подмножество упорядоченных пар $(a,в) \in R$ прямого произведения $M_1 \times M_2$, т. е. $R \subseteq M_1 \times M_2$. **Говорят, что отношение R задано на $M_1 \times M_2$.**

Как уже отмечалось, весьма часто все элементы принадлежат одному множеству M, т. е. $R \subseteq M \times M$, или $R \subseteq M^2$. Так, на множестве студентов группы могут быть заданы такие бинарные отношения, как «жить в одной комнате общежития», «быть моложе», «быть земляком» и т. д. **Тогда говорят, что отношение R задано на множестве M.**

Наравне с обозначением $(a,в) \in R$, $R \subseteq M^2$ в литературе используется обозначение $aRв$, $R \subseteq M^2$.

В общем случае могут рассматриваться n-местные отношения, например отношения между тройками элементов (тернарные) и т. д.

Пример. Равенство $x^2 + y^2 = z^2$ задает тернарное отношение на множестве целых чисел. Отношение является множеством, включающим все тройки чисел (x,y,z), удовлетворяющие данному равенству.

Пусть $R \subseteq A \times B$ определено в соответствии с рис. 1.5, из которого следует, что в отношении R задействованы не все, а лишь некоторые элементы исходных множеств A и B.

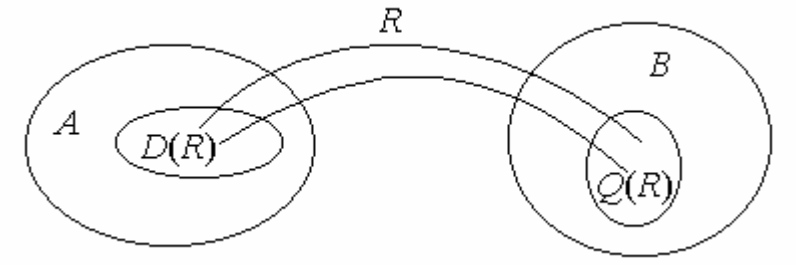


Рис. 1.5. Область определения и область значений отношения R

Тогда подмножество $D(R) = \{a \,/\, (a,b) \in R\}$ называется **областью определения** отношения R, а подмножество $Q(R) = \{b \,/\, (a,b) \in R\}$ – **областью значений** этого отношения. В литературе встречаются и иные обозначения: $D(R) = np_1R$, $Q(R) = np_2R$.

Пример. Пусть группа студентов в составе Иванова, Петрова, Сидорова и Кузнецова сдает экзамен. Процесс сдачи экзамена можно рассматривать как формирование отношения между областью определения (множество студентов группы) и областью значений (множество оценок – отл, хор, уд, неуд). Пусть студенты Иванов, Петров и Сидоров сдали на хорошо, а Кузнецов не явился. Тогда отношение R будет включать пары: (Иванов, хор), (Петров, хор), Сидоров, хор), соответственно, $D(R) = \{\text{Иванов, Петров, Сидоров}\}$, $Q(R) = \{\text{хор}\}$.

1.2.1. Способы задания бинарных отношений

Для задания бинарных отношений могут быть использованы любые, рассмотренные ранее, способы задания множеств. Кроме того, отношения, определенные на конечных множествах, могут задаваться:

1. Списком (перечислением) пар, для которых это отношение выполняется. Например, $R = \{(a,b),\ (a,c),\ (b,d)\}$.

2. Матрицей – бинарному отношению $R \subseteq M_1 \times M_2$, где $M_1 = \{a_1, a_2, \ldots, a_n\}$, $M_2 = \{b_1, b_2, \ldots, b_m\}$, соответствует матрица размера $n \times m$, в которой элемент c_{ij}, стоящий на пересечении *i*–й строки и *j*-го столбца, равен 1, если между a_i и b_j имеет место отношение R, или 0, если нет: $c_{ij} = \begin{cases} 1, \text{если } a_i R a_j; \\ 0 \text{ в противном случае.} \end{cases}$

3. Направленным графом, т. е. структурой, состоящей из вершин и дуг (направленных ребер). Элементы множеств отображаются в виде вершин графа, а отношения – в виде дуг, соединяющих эти вершины.

Два первых способа одинаково применимы как для отношений, заданных на разных множествах, так и для отношений на одном множестве. Третий способ больше подходит для отношений на одном множестве, т. к. позволяет получить более компактный граф.

Пример. Пусть $M = \{1,2,3,4,5,6\}$. Задать в явном виде (списком), описанием характеристических свойств, матрицей и графом отношение $R \subseteq M^2$, если R означает – «быть строго меньше».

1. С использованием распознающей процедуры можно записать $R = \{(a,b) / a,b \in M,\ a < b\}$.

2. Списком $R = \{(1,2),(1,3),(1,4),(1,5),(1,6).(2,3),\ldots\}$.

3. Матрица данного отношения имет вид

R	1	2	3	4	5	6
1	0	1	1	1	1	1
2	0	0	1	1	1	1
3	0	0	0	1	1	1 .
4	0	0	0	0	1	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

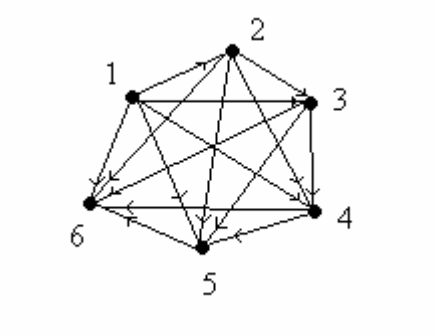


Рис. 1.6

Граф отношения приведен на рис. 1.6.

Пример. Для того же самого множества M={1,2,3,4,5,6} составить матрицы отношений $R_i \subseteq M^2$, если

$R1$ – «быть делителем»,

$R2$ – «иметь один и тот же остаток от деления на 3».

Матрицы имеют вид

R_1	1	2	3	4	5	6
1	1	1	1	1	1	1
2	0	1	0	1	0	1
3	0	0	1	0	0	1 ,
4	0	0	0	1	1	1
5	0	0	0	0	1	1
6	0	0	0	0	0	1

R_2	1	2	3	4	5	6
1	1	0	0	1	0	0
2	0	1	0	0	1	0
3	0	0	1	0	0	1 .
4	1	0	0	1	0	0
5	0	1	0	0	1	0
6	0	0	1	0	0	1

1.2.2. Свойства бинарных отношений

Пусть R – отношение на множестве M, $R \subseteq M^2$, т. е. множество отражается само в себя. Для отношений этого типа могут быть определены следующие свойства:

1. R – **рефлексивно**, если имеет место aRa для любого $a \in M$. Например, отношение $R = \{(a,b) / a \le b\}$ рефлексивно.

2. R – **антирефлексивно**, если ни для какого $a \in M$ не выполняется aRa. Например, отношение «быть сыном».

3. R – **симметрично**, если aRb влечет bRa. Например, отношение «жить в одном городе» симметрично.

4. R – **антисимметрично,** если aRb и bRa влекут $a=b$, т. е. ни для каких различающихся элементов a и b $(a\neq b)$ не выполняется одновременно aRb и bRa. Например, отношение «быть начальником» антисимметрично.

5. R – **транзитивно**, если aRb и bRc влекут aRc. Например, отношение «быть моложе» транзитивно.

Для отношений, заданных на прямом произведении различных множеств, т. е. при $R\subseteq M_1\times M_2$, свойства рефлексивности, симметричности и транзитивности не определяются.

Пример. Каковы свойства отношения «Быть не больше» ($R_1=\{(a,b)/a\leq b\}$), заданного на множестве натуральных чисел *N*?

Рефлексивно, т. к. $a\leq a$ для всех $a\in N$.

Антисимметрично, поскольку если $a\leq b$ и $b\leq a$, то $a=b$.

Транзитивно, т. к. если $a\leq b$ и $b\leq c$, то $a\leq c$.

Пример. Пусть $A=\{\pm,\oplus,\times,*\}$ и пусть $R\subseteq A\times A$ определено в виде $R=\{(\pm,\pm),(\pm,\oplus),(\pm,*),(\oplus,\pm),(*,\pm),(*,*),(\times,*),(\times,\times)\}$. Каковы свойства отношения?

R не является рефлексивным, т. к. $\oplus\in A$, но $(\oplus,\oplus)\notin R$..

R не является симметричным, поскольку $(\times,*)\in R$, но $(*,\times)\notin R$.

R не является антисимметричным, поскольку $(\oplus,\pm)\in R$ и $(\pm,\oplus)\in R$, но $\pm\neq\oplus$.

R не является транзитивным, т. к. $(\oplus,\pm)\in R$ и $(\pm,*)\in R$, но $(\oplus,*)\notin R$.

1.2.3. Эквивалентность и порядок

Рассматриваемые ниже отношения представляют собой формально определенные типы отношений, заданных на одном множестве, и отличающиеся фиксированным набором свойств.

Отношением **эквивалентности** (или просто эквивалентностью) называют бинарное отношение на множестве, если оно рефлексивно, симметрично, транзитивно. Например, отношение «жить в одном городе» на множестве людей – эквивалентность.

Отношение эквивалентности имеет важную особенность: эквивалентность R разбивает множество M, на котором оно задано, на непересекающиеся подмножества так, что элементы одного и того же подмножества находятся в отношении R, а между элементами разных подмножеств оно отсутствует. В этом случае говорят, что отношение R задает **разбиение** на множестве R, или **систему классов эквивалентно-**

сти по отношению R. Мощность этой системы называется **индексом разбиения**.

В то же время, любое разбиение множества на классы определяет некоторое отношение эквивалентности, а именно отношение «входить в один и тот же класс данного разбиения».

Это очень важное утверждение, т. к. дает основание строго определить свойства рефлексивности, симметричности и транзитивности некоторых отношений. Например, что сказать об отношении, заданном нуль-графом с петлями? Таким будет отношение «жить в одном городе» на множестве, где все люди живут в разных городах, или отношение «быть равным» на множестве натуральных чисел. Сам факт, что данное отношение разбивает множество на классы эквивалентности говорит о том, что оно симметрично и транзитивно, хотя граф не имеет ни одного ребра.

Пример. Пусть множество A – это набор разноцветных шаров, а отношение R задается условием $(a,b) \in R$ тогда и только тогда, когда a и b имеют одинаковый цвет. Поскольку R – отношение эквивалентности, каждый класс эквивалентности будет состоять из шаров, имеющих одинаковый цвет.

Замечание. Из того, что отношение «жить в одном городе» разбивает множество людей на непересекающиеся подмножества, т. е. является эквивалентностью, следует, что с точки зрения математики вполне естественно утверждение «Иванов живет в одном городе с самим собой» как условие рефлексивности данного отношения. Такие утверждения, не вызывающие сомнения при работе с числами, например $1=1$, выглядят непривычно, если элементами множеств являются люди.

Отношением нестрогого порядка (или нестрогим порядком) называют бинарное отношение на множестве, если оно рефлексивно, антисимметрично, транзитивно, и **отношением строгого порядка** (строгим порядком), – если оно антирефлексивно, антисимметрично, транзитивно. Оба эти отношения называются отношениями порядка.

Например, отношение «быть не старше» на множестве людей, «быть не больше» на множестве натуральных чисел – нестрогий порядок. Отношения «быть моложе», «быть прямым потомком» на множестве людей – строгий порядок.

Элементы a,b **сравнимы** по отношению порядка R на M, если выполняется aRb или bRa.

Множество M, на котором задано отношение порядка R, может быть:

• **полностью упорядоченным множеством**, если любые два элемента этого множества сравнимы по отношению порядка R;

• **частично упорядоченным множеством**, если сравнимы лишь некоторые элементы этого множества.

Пример. Отношения полного порядка на множестве людей – быть моложе, быть выше и т. д.

Отношение частичного порядка на множестве людей – быть начальником.

Пример. Каков индекс разбиения и мощности классов эквивалентности по отношению R, если R – отношение равенства (тождества) на любом множестве?

Ответ: Все классы эквивалентности по отношению равенства $R = \{(a,b) / a = b\}$ на любом множестве M состоят из одного элемента. Индекс разбиения M по отношению равенства равен мощности множества, т. е. $|M|$.

Пример. Каков индекс разбиения и мощности классов эквивалентности по отношению R, если R – отношение «иметь один и тот же остаток от деления на 5» на множестве натуральных чисел N?

Ответ: Индекс разбиения множества N по заданному отношению R равен 5. Множества натуральных чисел, составляющие каждый класс эквивалентности, счетны.

1.2.4. Операции над бинарными отношениями

Так как отношения являются обычными подмножествами $R \subseteq M_1 \times M_2$, то для них определены все те же операции, что и для любых множеств, т. е. объединение, пересечение, дополнение, разность. Кроме того, над отношениями определены и некоторые другие операции.

Обратное отношение R^{-1}.

Отношение $aR^{-1}b$ имеет место тогда и только тогда, когда имеет место aRb, соответственно $R^{-1} = \{(a,b) / (b,a) \in R\}$.

Пример. Если R – «быть моложе», то R^{-1} – быть старше ; если R – «быть подчиненным», то R^{-1} – «быть начальником».

Пример. Пусть A={1,2,3,4,5}, B={6,7,8,9}, C={10,11,12,13}. Пусть $R \subseteq A \times B$, $S \subseteq B \times C$ определены следующим образом: R={(1,7), (4,6), (5,6), 2,8)}, S={(6,10), (6,11), (7,10), (8,13)}. Определить отношения R^{-1}, S^{-1}.

Ответ: R^{-1}={(7,1), (6,4), (6,5), (8,2)};

S^{-1}={10,6), (11,6), (10,7), (13,13)}.

Составное отношение (композиция) – $R_1 \circ R_2$.

Пусть заданы множества M_1, M_2, M_3 и отношения $R_1 \subseteq M_1 \times M_2$, $R_2 \subseteq M_2 \times M_3$. Составное отношение действует из M_1 в M_2 посредством R_1 и из M_2 в M_3 посредством R_2.

Составное отношение может быть определено и на одном множестве. В частности, если $R \subseteq M^2$, то составное отношение $R \circ R = \{(a,b) / (a,c),(c,b) \in R\}$.

Пример: если R – «быть сыном», то $R \circ R$ – «быть внуком».

Транзитивное замыкание R^0.

Транзитивное замыкание R^0 состоит из таких и только таких пар элементов a,b из M, для которых в M существует цепочка из $k+2$ элементов M, $k \geq 0$, $a, c_1, c_2, \ldots, c_k, b$, между соседними элементами которой выполняется R, т. е.

$R^0 = \{(a,b) / \ (a,c_1),(c_1,c_2),\ldots,(c_k,b) \in R\}$.

Например, если R – отношение «быть сыном», то R^0 – «быть прямым потомком».

Если отношение R транзитивно, то $R = R^0$.

Пример. Пусть R – отношение «быть руководителем» на множестве M. Определить $\overline{R}$, R^{-1}, R^0. Каковы свойства отношений?

$\overline{R}$ – «не быть руководителем»;

R^{-1} – «быть подчиненным»;

$R^0 = R$ – «быть руководителем», т. к. R – транзитивно.

Отношение R – «быть руководителем»:

- не является рефлексивным, т. к. выражение «быть руководителем по отношению к самому себе» вряд ли имеет смысл;
- антирефлексивно, т. к. ни для какого члена организации не выполняется «A – руководитель A»;
- не симметрично, т. к. если A – руководитель B, то B не может быть руководителем A;
- антисимметрично, т. к. ни для каких членов организации не выполняется одновременно «A – руководитель B» и «B – руководитель A»;
- Транзитивно, т. к. если A – руководитель B и B – руководитель C, то A – руководитель C.

Таким образом, отношение «быть руководителем» антирефлексивно, антисимметрично и транзитивно, т. е. является отношением строгого частичного порядка на множестве M сотрудников фирмы.

1.2.5. Функциональные отношения

Отношение $R \subseteq A \times B$ называется **всюду (полностью) определенным**, если $D(R) = A$ (рис. 1.7, *а*). В противном случае отношение частично определенное.

Отношение $R \subseteq A \times B$ называется **сюръективным**, если $Q(R) = B$ (рис. 1.7, *б*).

Образом элемента a в множество B при отношении R называется множество всех $b \in B$, соответствующих элементу $a \in A$ (рис. 1.7, *в*).

Прообразом элемента b в множество A при отношении R называется множество всех $a \in A$, которым соответствует $b \in B$ (рис. 1.7, *г*).

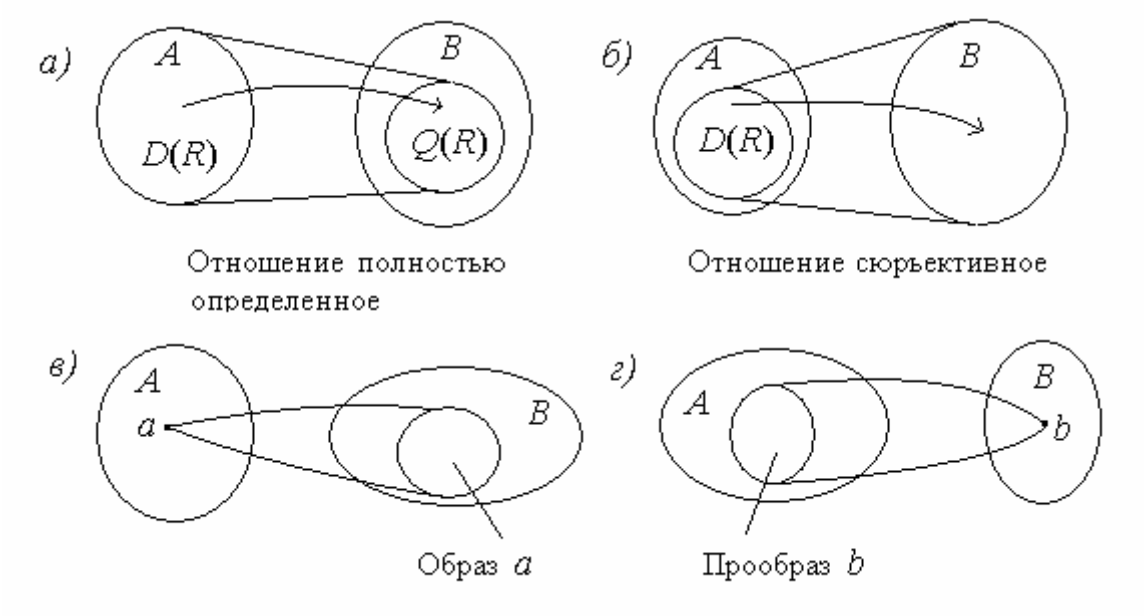


Рис. 1.7

Отношение $R \subseteq A \times B$ называется **функциональным (однозначным)**, или просто **функцией**, если образом любого элемента a из области определения $D(R)$ является единственный элемент b из области значений $Q(R)$.

Отношение $R \subseteq A \times B$ называется **инъективным (инъекцией)**, если прообразом любого элемента b из области значений $Q(R)$ является единственный элемент a из области определения $D(R)$.

Отношение называется **взаимно однозначным**, если оно:

- всюду определено;
- сюръективно;
- функционально;

• инъективно.

Примеры:

1. Матрица задает функциональное отношение, если в любой строке содержится только одна единичка.

2. Матрица задает взаимнооднозначное отношение, если в любой строке и любом столбце содержится одна и только одна единичка.

3. Отношение $R \subseteq X^2$, $R = \{(x, y) / x, y \in N,\ x^2 = y\}$ функционально.

4. Отношение $R \subseteq X^2$, $= R$ $\{(x, y) / x, y \in N,\ y = \sqrt{x}\}$ функционально. Но то же самое отношение, если x и y принадлежат множеству всех целых чисел (положительных и отрицательных), не является функциональным, т. к. $y = \pm\sqrt{x}$.

5. Позиция на шахматной доске представляет собой взаимно однозначное отношение между множеством оставшихся на доске фигур и множеством занятых ими полей.

1.2.6. Функции и отображения

Для всякого функционального соответствия R определим функцию f, связанную с этим соответствием. Пусть функция f устанавливает соответствие между множествами A и B. Говорят, что функция f имеет тип $A \to B$ (обозначается $f : A \to B$). Каждому элементу a из своей области определения функция f ставит в соответствие единственный элемент b из области значений. Это обозначается хорошо известной записью $f(a) = b$.

Отображением A **в** B называется всюду определенное функциональное отношение $f : A \to B$ (рис. 1.8, *а*).

Отображением A **на** B называется всюду определенное и сюръективное функциональное отношение $f : A \to B$) (рис. 1.8, *б*).

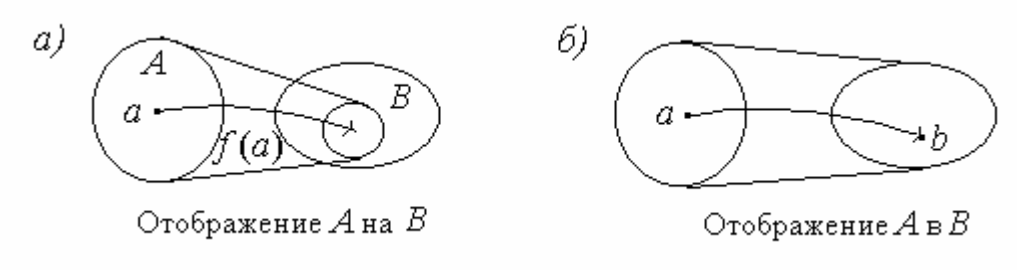


Рис.1.8

Пример. Пусть A – множество значений углов, B – множество вещественных чисел. Отношение $R = \{(a,b) / b = \sin a, a \in A, b \in B\}$ является функциональным, т. к. любое значение угла имеет только единственное значение синуса и, соответственно, определяет функцию $f : A \to B$. Данная функция является отображением, т. к. $D(R) = A$.

Является ли данное отображение отображением A в B? Да, т. к. оно всюду определено, т. е. $D(R) = A$.

Является ли данное отображение отображением A на B? Нет, т. к. оно не сюръективно, т. е. $Q(R) \neq B$.

1.2.7. Операции

Операцией называют функцию, все аргументы и значения которой принадлежат одному и тому же множеству. Функция одного аргумента называется унарной операцией.

Примеры унарных операций – элементарные функции $\sin x$, e^x, $\log x$; дополнение множества $\overline{A}$, обратное отношение R^{-1}.

Функция двух аргументов $\varphi(x, y) = z$, имеющая тип $\varphi: M \times M \to M$, называется **бинарной операцией**.

Примеры бинарных операций – арифметические операции сложения, умножения; операции над множествами – пересечение, объединение.

Множество M вместе с заданными на нем операциями $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ называется **алгеброй** и обозначается $A = \{M; \varphi_1, \ldots, \varphi_n\}$. Здесь M – называется **основным множеством**, а $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ – **сигнатурой алгебры**.

Пример: булева алгебра.

Глава 2

МАТЕМАТИЧЕСКАЯ ЛОГИКА

Математическая логика – современный вид формальной логики, т. е. науки, изучающей умозаключения с точки зрения их формального строения. Логика изучает правильные способы рассуждений – такие способы рассуждений, которые приводят к верным результатам в тех случаях, когда верны исходные предпосылки.

«Если все вороны черные, то все нечерные предметы не вороны». Данное высказывание несомненно истинно, и для того чтобы это утверждать, вообще не нужно знать что ворон – это птица. Или другой пример: «Все граждане России имеют право на образование. Иванов – гражданин России. Значит, он имеет право на образование».

Легко заметить, что эти примеры составлены по одной формальной схеме: «Все M суть P. S есть M. Следовательно, S есть P». Содержание терминов S, M, P для справедливости этих умозаключений безразлично.

Традиционная логика берет начало в древней Греции и Риме. Недаром ее называют Аристотелевой. До начала XIX в. формальная логика практически не выходила за рамки такого рода силлогических умозаключений. Однако, начиная с работ Джона Буля (труд «Законы мышления», 1854), можно говорить о превращении ее в математическую логику. Особенности математической логики заключаются в ее математическом аппарате, в преимущественном внимании к умозаключениям, применяемым в самой математике.

В 1910 г. появились первые работы, связанные с применением логики высказываний для описания переключательных цепей в телефонной связи. А в 1938–1940 гг. почти одновременно в СССР, США и Японии появились работы о применении математической логики в цифровой технике.

Современная математическая логика включает два основных раздела: **логику высказываний** и **логику предикатов**. Логика предикатов является охватывающей по отношению к логике высказываний (как алгебра к арифметике).

Изучать математическую логику можно, пользуясь двумя подходами (языками) – алгеброй логики и логическими исчислениями. Между основными понятиями этих языков формальной логики существует взаимно одно-

значное соответствие, т. е. они изоморфны. Это объясняется единством законов логики, лежащих в их основе. В данной работе, при изучении математической логики, будет использовать в основном язык алгебры логики, т. к. он чаще используется в технических приложениях.

2.1. Логические операции

2.1.1. Основные определения математической логики

Высказывание – повествовательное предложение (утверждение, суждение), о котором имеет смысл говорить, что оно истинно или ложно. Истинность или ложность высказываний определяется отношением содержания утверждения к действительному положению вещей.

При изучении высказываний предполагается, что выполняются следующие законы традиционной логики:

- **закон исключенного третьего** – каждое высказывание либо истинно, либо ложно;
- **закон противоречия** – никакое высказывание не является одновременно истинным и ложным.

Эти предложения, очевидно, абсолютизируют свойства реальности и в действительности не всегда выполняются.

Примеры высказываний: «дважды два – пять», «Сидоров – студент», «на дворе сентябрь», «зимой холодно». В ряде случаев истинность или ложность высказываний зависит от конкретной обстановки.

Будем называть высказывание **простым** (элементарным), если оно рассматривается нами как некое неделимое целое (аналогично элементу множества). Простым высказываниям в алгебре логики ставятся в соответствие **переменные**, принимающие значения «истина» или «ложь» и называемые по этой причине логическими переменными. Для упрощения записи мы будем использовать **вместо слова «истина» символ 1**, а **вместо слова «ложь» символ 0**. Обычно определение истинности или ложности простых высказываний не представляет проблемы.

Примеры простых высказываний: «Земля вращается вокруг Солнца», «На улице идет дождь».

Сложным называется высказывание, составленное из простых с помощью логических связок. В естественном языке роль связок при составлении сложных предложений играют грамматические средства – союзы «и», «или», «не»; слова «если…то» и др. В математической логике логические связки определены точно, как некоторые логические операции.

Примеры сложных высказываний: «На улице холодно и идет дождь», «Если вечером допоздна работаешь за компьютером и пьешь много кофе, то утром встаешь в плохом настроении или с головной болью».

Буквенные обозначения переменных, логические связки и скобки составляют **алфавит** логики высказываний. С помощью элементов алфавита можно построить разнообразные слова, которыми в логике высказываний являются **логические формулы**.

Логические формулы – алгебраические выражения, которые можно преобразовывать по определенным правилам, реализующим логические законы.

Пусть $B=\{0,1\}$ – бинарное множество, элементами которого являются формальные символы 0 и 1, не имеющие арифметического смысла и интерпретируемые как «нет», «да» или «ложь», «истина». На этом множестве заданы операции, имеющие смысл логических связок. Результатом является **алгебра логики**.

Таким образом, алгебра логики – это алгебра, образованная множеством $B=\{0,1\}$ со всеми возможными логическими операциями на нем.

Функцией алгебры логики, или логической функцией f от n переменных $f(x_1,x_2,\ldots,x_n)$, называется n-арная логическая операция на B, т. е. $f:B^n \to B$.

Любая логическая функция является сложным высказыванием, любая логическая переменная – простым высказыванием.

Рассмотрим некоторые примеры построения логических формул.

Пример. Пусть имеется сложное высказывание – «Если социологические исследования показывают, что потребитель отдает предпочтение удобству и многообразию выбора, то фирме следует сделать упор на усовершенствование товара или увеличение многообразия новых форм».

Разобьем исходное высказывание на простые и поставим им в соответствие логические переменные:

A – « социологические исследования показывают, что потребитель отдает предпочтение удобству»;

B – «социологические исследования показывают, что потребитель отдает предпочтение многообразию выбора»;

C – «фирме следует сделать упор на усовершенствование товара»;

D – «фирме следует сделать упор на увеличение многообразия новых форм».

Тогда логическая формула $f(A,B,C,D)$, эквивалентная исходному высказыванию, имеет вид «если A и B, то C или D».

Пример. Для высказывания «Если при выполнении программы отклонение контролируемых параметров превышает предусмотренные нормы, то требуется оперативная корректировка программы или уточнение стандартов», при обозначениях:

A – «отклонение контролируемых параметров превышает предусмотренные нормы»;

B – «требуется оперативная корректировка программы»;

C – «требуется уточнение стандартов»,

логическую формулу $f(A,B,C)$ можно записать в виде «если A, то B или C».

2.1.2. Таблицы истинности

Функциональная зависимость истинности сложного высказывания $f(x_1,x_2,\ldots,x_n)$ от истинности входящих в него элементарных высказываний $x_1,x_2,\ldots,x_n$ может быть описана построением **таблицы истинности** сложного высказывания.

Так как логические функции не имеют памяти, их удобно представлять как некоторый оператор, на который поступают входные сигналы $x_1,x_2,\ldots,x_n$, как это показано на рис. 2.1.

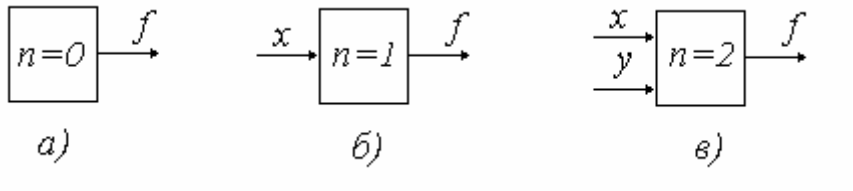


Рис. 2.1. Представление логических функций

Каждому набору входных сигналов соответствует некоторое значение выходной логической переменной f. Получив эти значения для всех возможных входных наборов, будем иметь полную информацию об истинностных значениях логической функции.

Так как каждая переменная может принимать только два значения, возможны 2^n различных двоичных наборов $x_1,x_2,\ldots,x_n$, каждому из которых ставится в соответствие истинностное значение сложного высказывания (логической функции).

Важно отметить, что число различных логических операторов (истинностных функций) конечно и зависит от числа аргументов логической формулы.

Истинностных функций от $n=0$ аргументов (рис. 2.1, а) всего две: это ноль-местные функции 0 и 1, называемые также логическими константами, т. е. логический оператор на рис. 2.1, а может быть реализован лишь в двух вариантах, либо как источник сигнала «истина» (1), либо как источник сигнала «ложь» (0).

Истинностных функций от $n=1$ аргументов всего четыре:

- функция–константа «ложь»: $0(x)=0$ при любом x;
- функция–константа «истина»: $1(x)=1$ при любом x;
- функция повторения: $r(x)=x$ при любом x;
- функция отрицания: $\overline{x}=0$ при $x=1$ и $\overline{x}=1$ при $x=0$.

Указанные функции могут быть также заданы табл. 2.1.

Таблица 2.1

Значения переменных	Значения функций			
x	Константа «0»	Константа «1»	Функция повторения $r(x)$	Функция отрицания $\overline{x}$
0	0	1	0	1
1	0	1	1	0

Значения двух первых функций не зависят от переменной x. Говорят, что переменная x является для данных функций фиктивной.

Истинностных функций от $n=2$ аргументов всего 16. Не все они одинаково важны для практики, но, чтобы иметь представление, представим эти функции в виде табл. 2.2.

Таблица 2.2

Значения переменных		Значения функций							
x	y								
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1
		Константа 0	Конъюнкция, «и»		Переменная x		Переменная y	Неравнозначность	Дизъюнкция, «или»
Обозначения		0	$\wedge,\cdot$		x		y	$\oplus$	$\vee,+$
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

		Стрелка Пирса	Эквивалентность	Отрицание y	Функция запрета	Отрицание x	Импликация	Штрих Шеффера	Константа 1
Обозначения		$\downarrow$	$\sim$	$\overline{y}$	$\leftarrow$	$\overline{x}$	$\rightarrow$	$\mid$	1

Из 16 логических функций 6 имеют фиктивные переменные.

2.1.3. Основные логические операции

Операция, заданная на некотором множестве, называется **унарной**, если она действует на один элемент этого множества и ее результатом является элемент этого же множества. Основная унарная операция – **отрицание (инверсия)**.

Отрицанием высказывания P называется высказывание, истинное, когда высказывание P ложно, и ложное в противном случае. Обозначение отрицания – $\overline{P}$.

Остальные операции, о которых далее говорится, – бинарные. Операция, заданная на некотором множестве, называется **бинарной**, если она действует на два элемента этого множества и ее результатом является элемент этого же множества.

Конъюнкцией (операцией «И», логическим произведением) двух высказываний – P и Q – называется высказывание, истинное, когда оба высказывания истинны, и ложное во всех других случаях. Обозначения: $P \wedge Q,\ P \cdot Q$.

Пример: Высказывание «На улице – дождь со снегом» можно рассматривать как конъюнкцию двух простых высказываний – «на улице идет снег», «на улице идет дождь».

Дизъюнкцией (операцией «ИЛИ», логической суммой) двух высказываний – P и Q – называется высказывание, ложное, когда оба высказывания ложны, и истинное во всех других случаях. Обозначения: $P \vee Q,\ P + Q$.

Пример: высказывание «идет дождь или снег» также состоит из двух простых высказываний, соединенных логической связкой «или». Его можно записать логической формулой $P \vee Q$ и оно истинно, если истинно хотя бы одно простое высказывание или оба вместе.

Импликацией (логическим следованием) двух высказываний – P и Q – называется высказывание, ложное, когда P истинно, а Q ложно; во всех других случаях – истинное. Обозначение: $P \rightarrow Q$. Здесь выска-

зывание P называется посылкой импликации, а высказывание Q – заключением (выводом).

Понятие импликации несколько сложнее для понимания, чем ранее рассмотренные операции. Из двух высказываний – P и Q – можно составить высказывание «P влечет Q» (другие варианты: «из P следует Q»; «если P, то Q»). Поэтому импликацию часто называют операцией логического следования, хотя важно понимать, что связь между P и Q не обязательно носит причинно–следственный характер. Высказывания P и Q могут логически не следовать одно из другого, более того, могут не иметь между собой никакой логической связи. Истинность приведенных ниже сложных высказываний зависит только от истинности их частей и не зависит от наличия связей между ними.

Пример: «если $2 \cdot 2 = 4$, то Москва – столица России» – истинно;
«если $2 \cdot 2 = 5$, то Москва – столица России» – истинно;
«если $2 \cdot 2 = 5$, то Москва – столица США» – истинно;
«если $2 \cdot 2 = 4$, то Москва – столица США» – ложно.

Анализируя истинность каждого из данных высказываний, необходимо отвлечься от его смысла, а обращать внимание только на форму. Все высказывание ложно, если посылка истинна, а вывод ложен.

Эквивалентностью двух высказываний – P и Q – называется высказывание, истинное, когда истинностные значения P и Q совпадают, и ложное в противном случае. Например, эквивалентность может быть использована для записи в виде логической формулы высказывания «Что в лоб, что по лбу». Обозначив A – «в лоб», B – «по лбу» получим общую формулу $A \equiv B$.

Неравнозначностью (исключающим «ИЛИ», сложением по модулю 2) двух высказываний – P и Q – называется высказывание, истинное, когда истинностные значения P и Q не совпадают, и ложное в противном случае.

Пример: Высказывание «сегодня понедельник или вторник» истинно, если истинно «сегодня понедельник» или «сегодня вторник». Из смысла самого высказывания следует, что должна использоваться именно операция «исключающее ИЛИ», т. к. данные простые высказывания не могут выполниться одновременно.

Примеры:

1. Правило заключения, формулируемое как «Если из высказывания A следует высказывание B и справедливо высказывание A, то справедливо B», может быть записано логической формулой $((A \to B) \wedge A) \to B$.

2. Аналогично правило отрицания «Если из высказывания A следует B, но высказывание B неверно, то неверно A» в виде логической формулы выглядит как $((A \to B) \wedge \overline{B}) \to \overline{A}$.

3. Пусть имеем сложное высказывание «Если при выполнении программы отклонение контролируемых параметров превышает предусмотренные нормы, то требуется оперативная корректировка программы или уточнение стандартов». Введя логические переменные: A – «отклонение контролируемых параметров превышает предусмотренные нормы», B – «требуется оперативная корректировка программы», C – «требуется уточнение стандартов», получим следующую логическую формулу: $A \to (B \vee C)$.

Как и в непрерывной математике, для сокращения числа скобок при записи формул придерживаются следующих соглашений по приоритетам операций:

- предполагается, что последовательность связок одного типа, записанная без скобок, вычисляется слева направо;
- принимается, что при отсутствии скобок высшим приоритетом обладает отрицание, затем следуют конъюнкция, дизъюнкция, импликация и эквивалентность, и в формуле опускаются те скобки, без которых возможно восстановление формулы с учетом данной субординации связок.

Пример. Формула $A \vee \overline{B} \to C \equiv A$ может быть записана так: $((A \vee (\overline{B})) \to C) \equiv A$.

Логические функции трех и более переменных также могут задаваться таблицами истинности или формулами, состоящими из символов переменных и знаков унарных и бинарных логических операций. Таким образом, формула наряду с таблицей служит способом задания и вычисления функций. В общем случае формула описывает логическую функцию как суперпозицию других, более простых, функций.

Пример: составить таблицу истинности функции трех переменных, заданной формулой $f(x_1, x_2, x_3) = (\overline{x}_1 \vee x_2) \to (x_1 \wedge x_3)$. Для построения таблицы истинности f вычислим ее значение на каждом из 8 наборов значений.

Таблица 2.3

x_1, x_2, x_3	$\overline{x}_1$	$\overline{x}_1 \vee x_2$	$x_1 \wedge x_3$	$(\overline{x}_1 \vee x_2) \to (x_1 \wedge x_3)$
000	1	1	0	0
001	1	1	0	0

010	1	1	0	0
011	1	1	0	0
100	0	0	0	1
101	0	0	1	1
110	0	1	0	0
111	0	1	1	1

2.1.4. Функционально полные системы (базисы)

Важной задачей математической логики являются преобразования логических формул. **Эквивалентными, или равносильными,** называют формулы, представляющие одну и ту же функцию. Стандартный метод установления эквивалентности двух формул состоит в следующем:

- по каждой формуле восстанавливается таблица истинности;
- полученные таблицы сравниваются по каждому набору значений переменных;
- если на всех наборах формулы дают одинаковые истинностные значения, они эквивалентны.

Пример: доказать эквивалентность формул

$$x_1 \mid x_2 = \overline{x_1 \wedge x_2} = \overline{x}_1 \vee \overline{x}_2 .$$

Воспользуемся стандартным методом, т. е. построим таблицу истинности для всех трех формул (табл.2.4).

Таблица 2.4

x_1, x_2	$x_1 \mid x_2$	$x_1 \wedge x_2$	$\overline{x_1 \wedge x_2}$	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_1 \vee \overline{x}_2$
00	1	0	1	1	1	1
01	1	0	1	1	0	1
10	1	0	1	0	1	1
11	0	1	0	0	0	0

Полученные результаты говорят о том, что формулы эквивалентны.

Как видно из примера, одна и та же логическая функция может быть задана формулами, включающими различные наборы логических операций. Существуют наборы логических операций, с помощью которых можно выразить любые другие логические операции. Такие наборы называются **функционально полными системами, или базисами**. Примерами таких базисов логических операций являются: $\{\wedge, \vee, -\}$, $\{\wedge, -\}$, $\{\vee, -\}$, $\}$, $\{|\}$, $\{\downarrow\}$, $\{\wedge, \oplus, 1\}$.

Наиболее хорошо изученным является **булев** базис $\{\wedge,\vee,-\}$. Формулы, содержащие только операции конъюнкции, дизъюнкции и отрицания называются **булевыми**.

Следующие две теоремы, приведенные без доказательств, устанавливают правила перехода от одного базиса к другому.

Теорема 1

Всякая логическая формула может быть представлена булевой формулой.

Теорема 2

Если все функции функционально полной системы Σ^* представимы формулами над Σ, то Σ также функционально полна.

Таким образом, чтобы перейти в записи логической формулы от одного базиса к другому, нужно просто заменить все операции первого базиса через операции второго базиса.

Алгебра $(P;\wedge,\vee,-)$, основным множеством которой является множество всех логических функций P, а операциями (т. е. сигнатурой Σ**)** – конъюнкция, дизъюнкция и отрицание, называется **булевой алгеброй логических функций**.

2.1.5. Совершенная дизъюнктивная нормальная форма

Система операций булевой алгебры полна, и переход от табличного задания любой логической функции к формуле булевой алгебры всегда возможен. Сформулируем очень важный для практики способ перехода от табличного задания логической функции к булевой формуле. Он включает следующие действия:

- для каждого набора значений переменных $x_1,x_2,\ldots,x_n$, на котором функция $f(x_1,x_2,\ldots,x_n)$ равна 1, выписываются конъюнкции всех переменных;
- над теми переменными, которые на этом наборе равны 0, ставятся отрицания;
- все такие конъюнкции соединяются знаками дизъюнкции.

Полученная таким образом формула называется **совершенной дизъюнктивной нормальной формой** (СДНФ) логической функции. Для каждой функции СДНФ единствена.

Таким образом, СДНФ функции $f(x_1,x_2,\ldots,x_n)$ представляет собой дизъюнкцию элементарных конъюнкций: $D = K_1 \vee K_2 \vee \ldots \vee K_m$, где все конъюнкции имеют одинаковое число сомножителей, равное числу логических переменных, а число конъюнкций равно числу набо-

ров значений переменных $x_1, x_2, \ldots, x_n$, на которых функция $f(x_1, x_2, \ldots, x_n)$ равна 1. Любые другие записи логической функции, вида $D = K_1 \vee K_2 \vee \ldots \vee K_m$, не отвечающие этим условиям, называются **дизъюктивными нормальными формами** (ДНФ) этой функции.

Пример: для логической функции, заданной в табл. 2.3, СДНФ имеет вид

$$f(x_1, x_2, x_3) = x_1 \wedge \overline{x}_2 \wedge \overline{x}_3 \vee x_1 \wedge \overline{x}_2 \wedge x_3 \vee x_1 \wedge x_2 \wedge x_3 .$$

Переход от логической формулы произвольного вида или формулы, записанной в некоторой не булевой алгебре с сигнатурой Σ^*, возможен не только через таблицу истинности, но и на основе теоремы 2. Для этого необходимо лишь выразить элементы Σ^* через дизъюнкцию, конъюнкцию и отрицание.

Пример. Перевести в булев базис следующую логическую формулу: $((A \to (B \wedge (C \vee Д))) \wedge A) \to (C \vee Д)$.

Для решения задачи воспользуемся соотношением $x_1 \to x_2 = \overline{x}_1 \vee x_2$, правильность которого легко проверить через построение таблицы истинности. Последовательно проводя преобразования, будем получать

$$\begin{aligned}
&((A \to (B \wedge (C \vee Д))) \wedge A) \to (C \vee Д) = \\
&= ((\overline{A} \vee (B \wedge (C \vee Д))) \wedge A) \to (C \vee Д) = \\
&= \overline{\overline{((\overline{A} \vee (B \wedge (C \vee Д))) \wedge A)}} \to (C \vee Д) = \\
&= \overline{\overline{((\overline{A} \vee (B \wedge (C \vee Д))) \vee \overline{A})}} \to (C \vee Д) = \\
&= \overline{\overline{A} \vee (B \wedge (C \vee Д))} \vee \overline{A} \vee C \vee Д = \\
&= A \wedge \overline{(B \wedge (C \vee Д))} \vee \overline{A} \vee C \vee Д .
\end{aligned}$$

Пример. В алгебре Жигалкина $(P_2; \wedge, \oplus, 1)$ ее сигнатура $\Sigma = \{\wedge, \oplus, 1\}$ является функционально полной системой. Убедиться в этом, используя теоремы 1 и 2.

Решение. Из теоремы 1 следует, что набор булевых функций полон. Тогда, в соответствии с теоремой 2, для доказательства функциональной полноты набора $\{\wedge, \oplus, 1\}$ достаточно доказать следующие равенства:

$$\overline{x} = x \oplus 1;$$

$$x_1 \vee x_2 = x_1 \wedge x_2 \oplus x_1 \oplus x_2 .$$

Используя обычный подход, построим таблицы истинности (табл. 2.5 и 2.6).

Таблица 2.5

x	$\overline{x}$	1	$x \oplus 1$
0	1	1	1
1	0	1	0

Таблица 2.6

x_1	x_2	$x_1 \vee x_2$	$x_1 \wedge x_2$	$(x_1 \wedge x_2) \oplus x_1$	$((x_1 \wedge x_2) \oplus x_1) \oplus x_2$
0	0	0	0	0	0
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	1	0	1

Из полученных таблиц истинности следует, что алгебра Жигалкина функционально полна.

Замечание. При описании булевых операций отмечалось, что наряду с символами $\wedge, \vee$ для обозначения конъюнкции и дизъюнкции используются традиционные символы умножения и сложения: $\cdot, +$, поэтому далее, там, где это ясно из контекста, будут преимущественно использоваться знаки $\cdot, +$. Таким образом, записи $x \vee (x \wedge z) \vee (\overline{x} \wedge y \wedge z) \vee (y \wedge \overline{z})$ и $x + x \cdot z + \overline{x} \cdot y \cdot z + y \cdot \overline{z}$ эквивалентны, однако последняя значительно короче.

2.1.6. Основные эквивалентные соотношения в булевой алгебре

1. Ассоциативность конъюнкции и дизъюнкции:
   $x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3 = x_1 \cdot x_2 \cdot x_3$;
   $x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3 = x_1 + x_2 + x_3$.
2. Коммутативность конъюнкции и дизъюнкции:
   $x_1 \cdot x_2 = x_2 \cdot x_1$;
   $x_1 + x_2 = x_2 + x_1$.
3. Дистрибутивность конъюнкции относительно дизъюнкции:
   $x_1 \cdot (x_2 + x_3) = x_1 \cdot x_2 + x_1 \cdot x_3$.
4. Дистрибутивность дизъюнкции относительно конъюнкции:
   $x_1 + (x_2 \cdot x_3) = (x_1 + x_2) \cdot (x_1 + x_3)$.
5. Идемпотентность:
   $x \cdot x = x, \qquad x + x = x.$
6. Закон двойного отрицания:
   $\overline{\overline{x}} = x$.
7. Свойства констант 0 и 1:

$x \cdot 1 \quad x, \quad = x + 1 \quad 1, \quad = \overline{0} \quad 1;$

$x \cdot 0 \quad 0, \; = x + 0 \quad x, \; = \overline{1} \quad 0. \; =$

8. Правила де Моргана:

$\overline{x_1 \cdot x_2} = \overline{x}_1 + \overline{x}_2, \qquad \overline{x_1 + x_2} = \overline{x}_1 \cdot \overline{x}_2.$

9. Закон противоречия:

$x \cdot \overline{x} = 0.$

10. Закон исключенного третьего:

$x + \overline{x} = 1.$

Особенность данных эквивалентных соотношений в том, что:

- они не выводимы друг из друга. Убедиться в их справедливости можно путем построения таблиц истинности;
- этих соотношений достаточно для выполнения любых эквивалентных преобразований.

Кроме основных соотношений, часто используются следующие соотношения, выводимые из основных:

- $x + x \cdot y \; = x, \qquad x \cdot (x + y) = x$ (поглощение);
- $x \cdot y + x \cdot \overline{y} = x$ (склеивание);
- $x + \overline{x} \cdot y = x + y.$

На основе основных и дополнительных эквивалентных соотношений булевой алгебры возможно проведение преобразований логических формул с целью получения более простых выражений.

Примеры:

1. Упростить булеву формулу $f_1(x, y, z) = x + x \cdot z + \overline{x} \cdot y \cdot z + y \cdot \overline{z}.$

Решение:

$f_1(x, y, z) = x + x \cdot z + \overline{x} \cdot y \cdot z + y \cdot \overline{z} = x + \overline{x} \cdot y \cdot z + y \cdot \overline{z} = x + y \cdot z + y \cdot \overline{z} = x + y$

.

2. Упростить булеву формулу $f_2(x, y, z) = x \cdot (\overline{y} + z) \cdot (\overline{x} + y + z).$

Решение: $f_2(x, y, z) = x \cdot (\overline{y} + z) \cdot (\overline{x} + y + z) = x \cdot z.$

2.1.7. Метод расщепления

Иногда возникает задача перехода от ДНФ произвольного вида, описывающей некоторую логическую функцию, к совершенной дизъюнктивной нормальной форме, реализующей ту же логическую функцию. Этот переход возможен, конечно, через таблицу истинности, однако более простым является так называемый метод расщепления. Рас-

смотрим его на примере. Пусть логическая функция трех переменных представлена следующей логической формулой:

$f_1(x,y,z) = x + x \cdot z + \overline{x} \cdot y \cdot z + y \cdot \overline{z}$.

Переход к СДНФ проведем путем следующих очевидных преобразований:

$$f_1(x,y,z) = x + x \cdot z + \overline{x} \cdot y \cdot z = x \cdot (y + \overline{y}) \cdot (z + \overline{z}) + x \cdot z \cdot (y + \overline{y}) + \overline{x} \cdot y \cdot z =$$

$$= (x \cdot y + x \cdot \overline{y}) \cdot (z + \overline{z}) + x \cdot y \cdot z + x \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z =$$

$$= x \cdot y \cdot z + x \cdot y \cdot \overline{z} + x \cdot \overline{y} \cdot z + x \cdot \overline{y} \cdot \overline{z} + x \cdot y \cdot z + x \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z =$$

$$= x \cdot y \cdot z + x \cdot y \cdot \overline{z} + x \cdot \overline{y} \cdot \overline{z} + x \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z.$$

2.2. Формы представления булевых функций

Выше рассмотрены две формы представления булевых функций – таблицы истинности и логические формулы. Познакомимся с некоторыми другими формами.

2.2.1. Геометрическое представление булевых функций

В геометрическом смысле каждый набор значений переменных $(x_1, x_2, \ldots, x_n)$ можно рассматривать как n-мерный вектор, определяющий точку в n-мерном пространстве [1]. Все множество двоичных наборов значений аргументов образует геометрическое множество вершин n-мерного единичного куба. Выделяя вершины, на которых значение функции равно 1, можно получить геометрический образ истинностной функции.

Пример. Функция задана таблицей истинности (табл. 2.7). Геометрическим представлением ее области истинности являются вершины куба, отмеченные на рис. 2.2 черными точками.

Таблица 2.7

x_1, x_2, x_3	$f(x_1, x_2, x_3)$

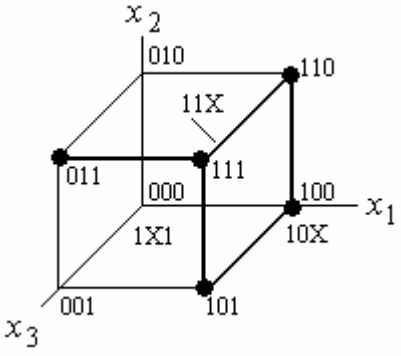


Рис. 2.2 . Геометрическое представление области истинности

000	0
001	0
010	0
011	1
100	1
101	1
110	1
111	1

Расстоянием между двумя вершинами называется число переменных, значения которых следует изменить, чтобы преобразовать вектор координат одной вершины в вектор координат другой.

2.2.2. Интервальное представление булевых функций

Интервальное представление устанавливает более тесную связь между геометрическим представлением логической функции и ее записью в виде логической формулы в булевом базисе [1]. Обозначим E^n – множество всех вершин единичного *n*-мерного куба. Пусть $N_f \subseteq E^n$ – множество всех таких вершин куба, на которых функция $f(x_1, x_2, \ldots, x_n) = 1$. Для вышеприведенного примера $N_f = \{(1\,0\,0), (1\,1\,0), (1\,0\,1), (1\,1\,1), (0\,1\,1)\}$. Множество N_f является фактически n-арным отношением на E^n вида $N_f = \{x_1, \ldots, x_n \,/\, \not\equiv (x_1, \ldots, x_n) \quad 1,\ x_1, \ldots, x_n \in E^n\}$.

Сформулированное в разд. 2.1.5 правило получения СДНФ связывает с каждой вершиной n-мерного куба конъюнкцию n членов, а со всей областью истинности функции – дизъюнкцию этих конъюнкций. Пусть для некоторой функции $f(x_1, x_2, x_3)$ $N_f = \{(1\,0\,0), (1\,1\,0)\}$. Тогда СДНФ имеет вид $f(x_1, x_2, x_3) = x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 + x_1 \cdot x_2 \cdot \overline{x}_3$. Применив к полученной формуле операцию склеивания по x_3, получим $f(x_1, x_2, x_3) = x_1 \cdot \overline{x}_2$. Легко заметить, что исходные вершины соединены ребром. Если связать с этим ребром конъюнкцию $x_1 \cdot \overline{x}_2$, то можно построить цепочку: конъюнкция трех составляющих – вершина куба, конъюнкция двух составляющих – ребро куба и т. д.

Для формализации полученной связи между конъюнкциями и элементами гиперкуба введем несколько определений [1].

Элементарной конъюнкцией называется конъюнкция переменных или их отрицаний $K = x_1^{a_1} \cdot x_2^{a_2} \cdot \ldots \cdot x_r^{a_r}$, где $x^a = \begin{cases} x & \text{при } a = 1; \\ \overline{x} & \text{при } a = 0, \end{cases}$ в которой каждая переменная встречается не более одного раза. Часто в литературе переменную или ее отрицание вида x^a называют **первичным термом**. Число r называется рангом конъюнкции. В случае $r = 0$ конъюнкция называется **пустой** и полагается равной 1.

Подмножество $N_k \subseteq E^n$ называется **интервалом *r*-го ранга**, если оно соответствует элементарной конъюнкции K r-го ранга.

Очевидно, каждой конъюнкции $K = x_1^{a_1} \cdot x_2^{a_2} \cdot \ldots \cdot x_r^{a_r}$ соответствует интервал N_k, состоящий из всех вершин гиперкуба, у которых $x_1 = a_1, \ldots, x_r = a_r$, а значения остальных координат произвольны. Таким образом, каждая вершина куба E^n является интервалом n-го ранга, множество всех вершин E^n – интервалом нулевого ранга.

Пример. В трехмерном кубе конъюнкции $x_2 \cdot x_3$ соответствует ребро $N_{x_2 x_3} = \{(0\,1\,1), (1\,1\,1)\}$, являющееся интервалом 2-го ранга (такой интервал часто обозначают $(X\,1\,1)$), а конъюнкции x_1 – грань $N_{x_1} = \{(1\,0\,1), (1\,0\,0), (1\,1\,0), (1\,1\,1)\}$, являющаяся интервалом 1–го ранга.

Для некоторой логической функции $f(x_1, x_2, \ldots, x_n)$ можно ввести понятие комплекса интервалов r-го ранга. Например, для функции, заданной выше таблицей истинности 2.7 и рис. 2.2, можно выделить следующие комплексы:

$$N_f^3 = \{(0\,1\,1), (1\,0\,0), (1\,0\,1), (1\,1\,0), (1\,1\,1)\} = N_f;$$

$$N_f^2 = \{(X\,1\,1), (1\,0\,X), (1X\,0), (1X\,1), (1\,1X)\};$$

$$N_f^1 = \{(1XX)\}.$$

2.3. Синтез логических схем

Для того чтобы оценить полезность применения логических функций в решении практических задач и сформулировать вопросы для дальнейшего изучения, рассмотрим задачу проектирования системы управления табло для соревнований штангистов. Логика работы системы управления следующая.

Подсвечивание табло «Вес взят» происходит по сигналу «1», которое выдает устройство, обрабатывающее сигналы трех судей – A, B, C.

Судья A – старший. Сигнал на подсвечивание выдается только тогда, когда все судьи или два из них нажали свои кнопки, но при этом одним из них должен быть старший судья.

Идея решения состоит в переходе от словесного описания логики работы устройства управления к логической формуле с последующим ее преобразованием, упрощением и реализацией с помощью типовых логических компонентов. Фактически словесное описание рассматривается как сложное высказывание.

Решение данной задачи начнем с построения таблицы истинности, отражающей сформулированные выше требования к работе системы управления (таб. 2.8).

Таблица 2.8

A	B	C	f	Примечания
1	1	1	1	Все судьи «за»
1	1	0	1	Двое судей «за», в том числе и судья A
1	0	1	1	Двое судей «за», в том числе и судья A
1	0	0	0	Двое судей «против»
0	1	1	0	Судья «A» против
0	1	0	0	
0	0	1	0	
0	0	0	0	

Далее, используя алгоритм разд.2.1.5, проведем переход к логической формуле в СДНФ, которая имеет следующий вид:

$$f(A,B,C) = A \cdot B \cdot C + A \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C . \qquad (2.1)$$

Заключительным этапом может являться техническая реализация полученной формулы в виде схемы, включающей логические устройства, реализующие булевы функции. Таких устройств три. Они носят следующие названия: элемент «И», элемент «ИЛИ», элемент «НЕ», или инвертор. В частности, они могут быть реализованы на релейных элементах, как показано на рис. 2.3, где Pa, Pb – катушки реле, a, b – контакты реле.

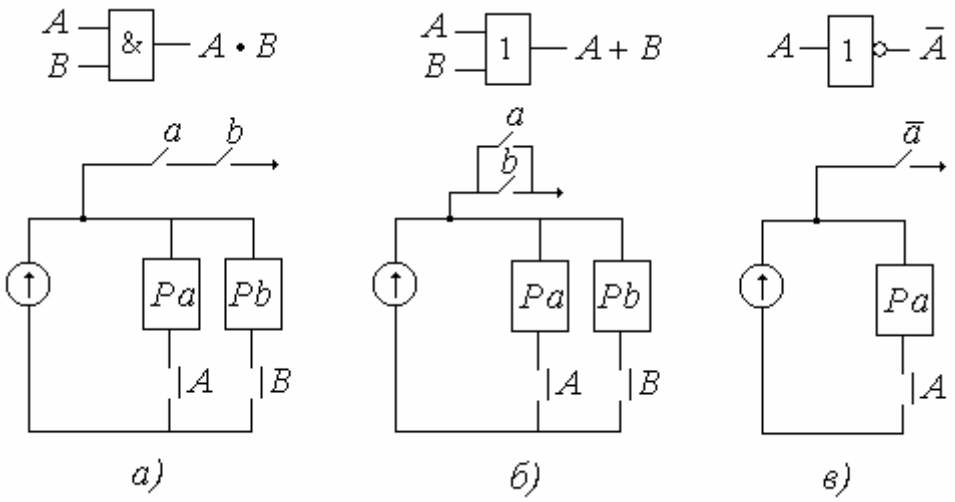


Рис. 2.3. *а* – элемент «И»; *б* – элемент «ИЛИ»; *в* – элемент «НЕ»

В общем случае элементы «И» и «ИЛИ» могут иметь не два, а произвольное число входов.

Общая схема управления табло, полученная по СДНФ, представлена на рис. 2.4. В ней использованы два элемента «НЕ», три трехвходовых элемента «И» и один трехвходовый элемент «ИЛИ».

Дело, однако, в том, что использованное формульное представление данной логической функции не единственно. Той же самой таблице истинности соответствует еще по крайней мере одна булевская формула. Правило ее получения следующее:

- для каждого набора значений переменных $x_1, x_2, \ldots, x_n$, на котором функция $f(x_1, x_2, \ldots, x_n)$ равна 0, выписываются дизъюнкции всех переменных;
- над теми переменными, которые на этом наборе равны 1, ставятся отрицания;
- все такие дизъюнкции соединяются знаками конъюнкции.

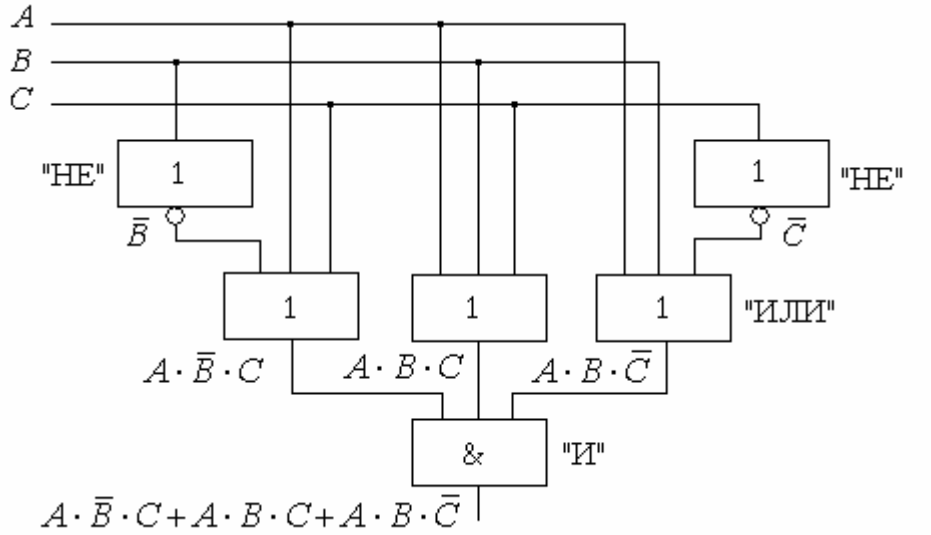


Рис. 2.4. Схема управления табло

Полученная таким образом формула называется **совершенной конъюнктивной нормальной формой** (СКНФ) логической функции.

Формула, соответствующая табл. 2.8, в СКНФ имеет вид

$$f(A,B,C)=(\bar{A}+B+C)\cdot(A+\bar{B}+\bar{C})\cdot \\ \cdot(A+\bar{B}+C)\cdot(A+B+\bar{C})\cdot(A+B+C). \qquad (2.2)$$

Построение для нее таблицы истинности (табл. 2.9) показывает, что формулы (2.1) и (2.2) эквивалентны.

Таблица 2.9

ABC	$\bar{B}$	$\bar{C}$	$\bar{A}$	$\bar{A}+B+C$	$A+\bar{B}+\bar{C}$	$A+\bar{B}+C$	$A+B+\bar{C}$	$A+B+C$	f
111	0	0	0	1	1	1	1	1	1
110	0	1	0	1	1	1	1	1	1
101	1	0	0	1	1	1	1	1	1
100	1	1	0	0	1	1	1	1	0
011	0	0	1	1	0	1	1	1	0
010	0	1	1	1	1	0	1	1	0
001	1	0	1	1	1	1	0	1	0
000	1	1	1	1	1	1	1	0	0

Соответственно, схема логического устройства, реализующего формулу (2.2), будет иной. Легко посчитать, что для реализации этой формулы необходимы три инвертора, один блок перемножения на четыре входа и четыре трехвходовых блока логического сложения. Таким

образом, с точки зрения технической реализации схемы отнюдь не равнозначны.

Более того, легко проверить (табл. 2.10), что ту же самую таблицу истинности имеет формула $f(A,B,C) = A \cdot (B + C)$, которая требует для своей реализации всего один логический сумматор (элемент «ИЛИ») и один блок перемножения (элемент «И»).

Таблица 2.10

ABC	$B + C$	$(B + C) \cdot A$
111	1	1
110	1	1
101	1	1
100	0	0
011	1	0
010	1	0
001	1	0
000	0	0

Пример. Провести синтез (до уровня логической формулы) устройства, предназначенного для включения и выключения света в длинной подземной галерее, имеющей два входа. В систему входят два выключателя (A и B), установленные у входов в галерею, и устройство управления лампами. Если в галерее никого нет, она не освещается. Входя в галерею через любой вход, можно зажечь лампы, выходя через любой вывод – выключить.

Решение. Таблица истинности проектируемого устройства представлена ниже (табл.2.11).

Таблица 2.11

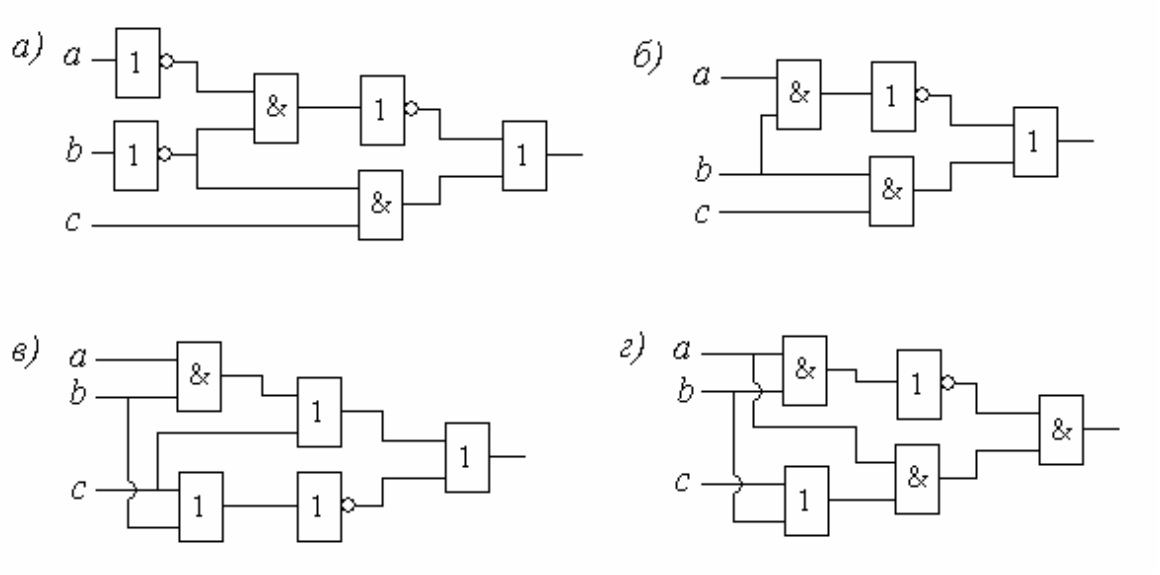
A B	$F(A,B)$	Примечания
0 0	0	В галерее никого нет. Выключатели выключены.
0 1	1	Кто-то зашел через вход B и включил освещение.
1 1	0	Выйдя через вход A, – выключил
1 0	1	Кто-то зашел через вход A и включил освещение

Соответствующая ей совершенная ДНФ имеет вид

$$F(A,B) = A \cdot \overline{B} + \overline{A} \cdot B .$$

Примеры

Приведите булевы выражения, соответствующие коммутационным схемам *а, б, в, г*.



Ответ: $f_a = (\overline{\overline{a}\cdot\overline{b}}) + (\overline{b}\cdot c)$, $f_б = \overline{(a\cdot b)} + (\overline{b}\cdot c)$;
$f_в = ((a\cdot b)+c) + \overline{(b+c)}$, $f_г = \overline{(a\cdot b)}\cdot((b+c)\cdot a)$.

2.4. Минимизация дизъюнктивных нормальных форм

Рассмотренные в предыдущем разделе примеры показывают, что булева алгебра дает систематический подход к построению комбинационных схем – простейших преобразователей информации, реализующих функциональное отображение конечных множеств. При этом актуальной является задача получения наилучших в некотором смысле технических решений.

2.4.1. Приведение к дизъюнктивной нормальной форме

Ранее определено, что всякая функция алгебры логики, отличная от 0, может быть представлена совершенной дизъюнктивной нормальной формой

$$f(x_1, x_2, \ldots, x_n) = \bigvee_{a_1, a_2, \ldots, a_n} x_1^{a_1}\cdot x_2^{a_2}\cdot\ldots\cdot x_n^{a_n},$$

где $x^a = \begin{cases} x & \text{при } a = 1; \\ \overline{x} & \text{при } a = 0. \end{cases}$

Однако СДНФ обычно допускает упрощения, в результате которых получается формула, также реализующая f, но содержащая меньшее число символов (термов). Рассмотрим методы представления функций

алгебры логики простейшими формулами в классе так называемых дизъюнктивных нормальных форм (ДНФ).

Введем ряд определений. Назовем выражение

$$K = x_1^{a_1} \cdot x_2^{a_2} \cdot \ldots \cdot x_k^{a_k},$$

где $x^a = \begin{cases} x & \text{при } a = 1; \\ \overline{x} & \text{при } a = 0, \end{cases}$

элементарной конъюнкцией.

Дизъюнктивной нормальной формой (ДНФ) называется формула, имеющая вид дизъюнкции элементарных конъюнкций $D = K_1 + K_2 + \ldots + K_m$, в которой все K_j различны.

В таком случае СДНФ (совершенная дизъюнктивная нормальная форма) – это ДНФ, в которой каждая конъюнкция содержит все переменные или их отрицания.

Количество первичных термов, которые образуют форму, задающую булеву функцию $f(x_1, x_2, \ldots, x_n)$, называют сложностью $L(f)$ этой формы. Например, для $f(a,b,c) = a \cdot b + \overline{c} + \overline{b} \cdot c$ $L(f) = 5$.

Минимальной (наименее сложной) ДНФ функции $f(x_1, x_2, \ldots, x_n)$ называется ДНФ, реализующая $f(x_1, x_2, \ldots, x_n)$ и содержащая наименьшее число термов по сравнению со всеми другими ДНФ, реализующими $f(x_1, x_2, \ldots, x_n)$ [1].

Существует тривиальный алгоритм построения минимальной ДНФ. Все ДНФ, составленные из переменных $x_1, x_2, \ldots, x_n$, упорядочиваются по возрастанию числа термов и по порядку для каждой ДНФ проверяется соотношение $D = f(x_1, x_2, \ldots, x_n)$. Первая по порядку ДНФ, для которой это соотношение выполняется, есть, очевидно, минимальная ДНФ. Однако уже при $n > 2$ этот метод приводит к перебору огромного числа ДНФ, т. к. известно, что число различных элементарных конъюнкций, составленных из $x_1, x_2, \ldots, x_n$, равно 3^n. Число же всех возможных ДНФ, составленных из $x_1, x_2, \ldots, x_n$, т. е. мощность множества, из которого требуется сделать выбор, равно 2^{3^n}. Таким образом, тривиальный алгоритм построения минимальной ДНФ является чрезвычайно трудоемким, по крайней мере при ручной обработке.

Существуют различные способы повышения эффективности алгоритма синтеза минимальных ДНФ. Большинство из них заключается в том, что из множества всех элементарных конъюнкций некоторым (сравнительно нетрудоемким) способом удаляются конъюнкции, которые заведомо не входят в минимальные ДНФ. Это приводит к сниже-

нию мощности множества ДНФ, в котором находятся минимальные ДНФ заданной функции. Вторая группа способов связана с более экономичным перебором ДНФ из этого множества.

2.4.2. Геометрическая интерпретация задачи минимизации ДНФ

Для каждой ДНФ $K_1 + K_2 + \ldots + K_m$ функции $f(x_1, x_2, \ldots, x_n)$ выполняется соотношение $N_f = \bigcup_{j=1}^{m} N_{k_j}$.

Говорят [1], что с каждой ДНФ функции f связано покрытие подмножества N_f такими интервалами $N_{k_1}, N_{k_2}, \ldots, N_{k_m}$, что $N_{k_j} \subseteq N_f$, где $j = \overline{1,m}$. Обозначим через r_j ранг интервала N_{k_j}. Тогда $r = \sum_{j=1}^{m} r_j$ совпадает с числом символов в ДНФ. Задача отыскания минимальной ДНФ сводится, очевидно, к отысканию такого покрытия N_f интервалами $N_{k_j} \subseteq N_f$, чтобы выражение $r = \sum_{j=1}^{m} r_j$ было минимальным.

Простейший известный нам способ покрытия – покрытие СДНФ. Здесь каждый $r_j = n$. Для ранее рассмотренного графического примера имели

$$f(x_1, x_2, x_3) = \bar{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 +$$
$$+ x_1 \cdot \bar{x}_2 \cdot x_3 + x_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot x_2 \cdot x_3.$$

Графически из рис. 2.2 получим покрытие $f(x_1, x_2, x_3) = x_1 + x_2 \cdot x_3$.

2.4.3. Допустимые конъюнкции

При построении минимальных ДНФ для функции f достаточно рассматривать только те интервалы N_k, для которых $N_k \subseteq N_f$, т. е. $N_k \cap (E^n \setminus N_f) = \varnothing$. Такие интервалы и соответствующие им конъюнкции называются **допустимыми** [1].

Возможен следующий способ повышения эффективности тривиального алгоритма. Выделяются все допустимые конъюнкции K_j и к множеству этих конъюнкций $\{K_j\}$ применяется тривиальный алгоритм. Этот простой прием существенно увеличивает эффективность тривиального алгоритма.

Множество всех допустимых конъюнкций $\{K_j\}$может быть получено путем удаления из числа 3^n конъюнкций «лишних». Заметим, что конъюнкции, обращающиеся в единицу в вершине $(a_1,a_2,\ldots,a_n)$, есть произведения некоторых из символов $(x_1^{a_1},x_2^{a_2},\ldots,x_n^{a_n})$, поэтому если $(a_1,a_2,\ldots,a_n)\in(E^n \setminus N_f)$, то из списка всех 3^n конъюнкций следует удалить 2^n конъюнкций, составленных из сомножителей, входящих в множество $(x_1^{a_1},x_2^{a_2},\ldots,x_n^{a_n})$. Иными словами, **если некоторая нулевая вершина *n*-куба связана с конъюнкцией $(x_1^{a_1},x_2^{a_2},\ldots,x_n^{a_n})$, то недопустимыми являются все конъюнкции, которые могут быть получены из данного набора символов**.

Пример

Рассмотрим функцию, заданную таблицей истинности (табл. 2.12).

Таблица 2.12

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

Множество $E^n \setminus N_f$ состоит из трех точек – (011), (100), (110). Составим таблицу конъюнкций, связанных с этими точками (табл. 2.13).

Таблица 2.13

Точки множества $E^n \setminus N_f$	Конъюнкции, обращающиеся в единицу в точках на $E^n \setminus N_f$
011	$\overline{x},\ y,\ z;\ \overline{x}\cdot y,\ \overline{x}\cdot z,\ y\cdot z;\ \overline{x}\cdot y\cdot z$
100	$x,\ \overline{y},\ \overline{z};\ x\cdot\overline{y},\ x\cdot\overline{z},\ \overline{y}\cdot\overline{z};\ x\cdot\overline{y}\cdot\overline{z}$
110	$x,\ y,\ \overline{z};\ x\cdot y,\ x\cdot\overline{z},\ y\cdot\overline{z};\ x\cdot y\cdot\overline{z}$

Все конъюнкции, указанные в данной таблице, должны быть удалены из множества 27 конъюнкций, составленных из переменных x,y,z. После удаления останутся конъюнкции $x\cdot\overline{y}\cdot\overline{z},\ x\cdot\overline{y}\cdot z,\ x\cdot y\cdot\overline{z},\ x\cdot\overline{y}\cdot z,\ x\cdot y\cdot z,\ \overline{x}\cdot\overline{y},\ \overline{x}\cdot\overline{z},\ x\cdot z,\ \overline{y}\cdot z$.

2.4.4. Сокращенная ДНФ

Определение. Интервал N_k называется максимальным для f, если $N_k \subseteq N_f$ и не существует интервала $N_{k'}$ такого, что $N_k \subset N_{k'} \subseteq N_f$ [1]. При проверке отношения $N_k \subset N_{k'}$ полезно иметь в виду, что оно выполняется тогда и только тогда, когда $K < K'$, т. е. когда конъюнкция K' получается из конъюнкции K вычеркиванием непустого числа сомножителей.

Очевидно, что каждый интервал $N_k \subseteq N_f$ содержится в некотором максимальном интервале $N_{k_j} \subseteq N_f$. Поэтому совокупность $\{N_{k_j}\}, j = 1,\ldots,m$ всех максимальных для f интервалов определяет покрытие подмножества N_f: $N_f = \bigcup_{j=1}^{m} N_{k_j}$.

Определение. ДНФ $\bigvee_{j=1}^{m} K_j$, реализующая функцию $f(x_1,x_2,\ldots,x_n)$ и соответствующая покрытию подмножества N_f всеми максимальными для f интервалами, называется сокращенной ДНФ функции $f(x_1,x_2,\ldots,x_n)$.

Пример. Из рис. 2.5 следует, что область истинности включает четыре интервала 2-го ранга, которые образуют покрытие области истинности $\overline{x}\cdot\overline{z} + \overline{y}\cdot\overline{z} + \overline{x}\cdot y + y\cdot z$. Интервалов 1-го ранга здесь нет. Таким образом, полученная ДНФ является сокращенной ДНФ функции $f(x_1,x_2,\ldots,x_n)$.

Сокращенная ДНФ не является, вообще говоря, минимальной ДНФ. В частности, минимальными для данной $f(x_1,x_2,\ldots,x_n)$ являются ДНФ $\overline{x}\cdot\overline{z} + \overline{y}\cdot\overline{z} + y\cdot z$ и $\overline{y}\cdot\overline{z} + \overline{x}\cdot y + y\cdot z$. Геометрически легко заметить, что эти формы соответствуют покрытию подмножества N_f минимальным числом максимальных для f интервалов. Алгебраически же может быть сформулирована следующая теорема:

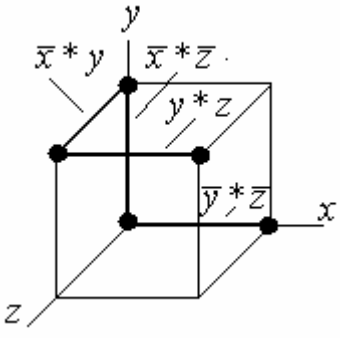


Рис. 2.5

Теорема. Минимальная ДНФ функции $f(x_1,x_2,\ldots,x_n)$ получается из сокращенной ДНФ функции $f(x_1,x_2,\ldots,x_n)$ путем удаления некоторых элементарных конъюнкций.

Из данной теоремы следует, что при построении минимальных форм нет необходимости рассматривать все допустимые конъюнкции – достаточно ограничиться теми, которые входят в сокращенную ДНФ.

2.4.5. Построение сокращенной ДНФ

Один из методов построения сокращенной ДНФ – геометрический – ранее уже был изложен. Он нагляден, однако его применимость ограничена функциями трех аргументов. Для функций с большим числом аргументов необходимы аналитические способы.

Существует целый ряд методов синтеза сокращенной ДНФ. Суть всех этих методов – в последовательном упрощении логического выражения, обычно заданного в виде СДНФ. В процессе упрощения используются следующие преобразования:

1) склеивание – $x\cdot K+\overline{x}\cdot K=K$;

2) поглощение – $K'+K'\cdot K''=K'$;

3) неполное склеивание – $x\cdot K+\overline{x}\cdot K=x\cdot K+\overline{x}\cdot K+K$;

4) обобщенное склеивание – $x\cdot K'+\overline{x}\cdot K''=x\cdot K'+\overline{x}\cdot K''+K'\cdot K''$.

Рассмотрим один из методов получения сокращенной ДНФ из СДНФ, известный как метод Блейка – Порецкого [1, 5], который заключается в неполном попарном склеивании всех элементарных конъюнкций СДНФ между собой и затем – использовании правила поглощения. Эта процедура повторяется для элементарных конъюнкций меньшего числа переменных до тех пор, пока склеивание станет невозможным. Не вдаваясь глубоко в теорию, рассмотрим пример.

Пример. Получить сокращенную ДНФ функции, заданной в виде совершенной дизъюнктивной нормальной формы

$$f(x,y,z,v)=x\cdot y\cdot z\cdot v+\overline{x}\cdot\overline{y}\cdot\overline{z}\cdot v+\overline{x}\cdot y\cdot\overline{z}\cdot v+$$

$$+\overline{x}\cdot\overline{y}\cdot z\cdot v+x\cdot\overline{y}\cdot\overline{z}\cdot v+x\cdot\overline{y}\cdot z\cdot v+\overline{x}\cdot y\cdot z\cdot v.$$

С целью формализации процесса пронумеруем все конъюнкции СДНФ:

$$f(x,y,z,v)=\underset{1}{x\cdot y\cdot z\cdot v}+\underset{2}{\overline{x}\cdot\overline{y}\cdot\overline{z}\cdot v}+\underset{3}{\overline{x}\cdot y\cdot\overline{z}\cdot v}+$$

$$+\underset{4}{\overline{x}\cdot\overline{y}\cdot z\cdot v}+\underset{5}{x\cdot\overline{y}\cdot\overline{z}\cdot v}+\underset{6}{x\cdot\overline{y}\cdot z\cdot v}+\underset{7}{\overline{x}\cdot y\cdot z\cdot v}.$$

Выполним все возможные неполные попарные склеивания конъюнкций четырех переменных, нумеруя получающиеся конъюнкции трех переменных парой индексов, относящихся к исходным конъюнкциям. Получим

$$f(x,y,z,v) = x\cdot y\cdot z\cdot v + \overline{x}\cdot\overline{y}\cdot\overline{z}\cdot v + \overline{x}\cdot y\cdot\overline{z}\cdot v + \overline{x}\cdot\overline{y}\cdot z\cdot v +$$
$$\qquad\qquad 1 \qquad\qquad 2 \qquad\qquad 3 \qquad\qquad 4$$
$$+x\cdot\overline{y}\cdot\overline{z}\cdot v + x\cdot\overline{y}\cdot z\cdot v + \overline{x}\cdot y\cdot z\cdot v + x\cdot z\cdot v + y\cdot z\cdot v + \overline{x}\cdot\overline{z}\cdot v +$$
$$\quad 5 \qquad\quad 6 \qquad\quad 7 \qquad\quad 1,6 \qquad 1,7 \qquad 2,3$$
$$+\overline{x}\cdot\overline{y}\cdot v + \overline{y}\cdot\overline{z}\cdot v + \overline{x}\cdot y\cdot v + \overline{y}\cdot z\cdot v + \overline{x}\cdot z\cdot v + x\cdot\overline{y}\cdot v.$$
$$2,4 \qquad 2,5 \qquad 3,7 \qquad 4,6 \qquad 4,7 \qquad 5,6$$

Теперь проведем процедуру поглощения конъюнкций. Легко видеть, что все конъюнкции четырех переменных поглощены коньюнкциями трех переменных. В результате получим

$$f(x,y,z,v) = x\cdot z\cdot v + y\cdot z\cdot v + \overline{x}\cdot\overline{z}\cdot v + \overline{x}\cdot\overline{y}\cdot v +$$
$$\qquad\qquad 1,6 \qquad 1,7 \qquad 2,3 \qquad 2,4$$
$$+\overline{y}\cdot\overline{z}\cdot v + \overline{x}\cdot y\cdot v + \overline{y}\cdot z\cdot v + \overline{x}\cdot z\cdot v + x\cdot\overline{y}\cdot v.$$
$$\quad 2,5 \qquad 3,7 \qquad 4,6 \qquad 4,7 \qquad 5,6$$

Аналогичным образом выполним все возможные неполные попарные склеивания полученных элементарных конъюнкций трех переменных и проведем процедуру поглощения. В итоге имеем

$$f(x,y,z,v) = z\cdot v + \overline{x}\cdot v + \overline{y}\cdot v.$$

Дальнейшее склеивание невозможно, следовательно, сокращенная дизъюнктивная нормальная форма получена.

2.4.6. Тупиковые ДНФ

После того как сокращенная ДНФ построена, для получения минимальной ДНФ можно воспользоваться тривиальным алгоритмом. Ясно, что его эффективность будет еще выше. Возможен, однако, другой подход, связанный с перебором лишь так называемых тупиковых ДНФ [1].

Определение. Покрытие подмножества N_f максимальными интервалами называется неприводимым, если после удаления из него любого интервала оно перестает быть покрытием.

Определение. ДНФ функции f называется тупиковой, если ей соответствует неприводимое покрытие подмножества N_f. Очевидно, что всякая минимальная ДНФ является тупиковой.

Пример. Рассмотрим функцию

$$f(x,y,z) = \overline{x}\cdot\overline{y}\cdot\overline{z} + \overline{x}\cdot\overline{y}\cdot z + x\cdot\overline{y}\cdot z +$$
$$+x\cdot y\cdot z + x\cdot y\cdot\overline{z} + \overline{x}\cdot y\cdot\overline{z}.$$

и соответствующее ей подмножество N_f (рис. 2.6). Сокращенную ДНФ этой функции можно записать следующим образом:

$$D(f) = \overline{x}\cdot\overline{y} + \overline{y}\cdot z + x\cdot z + x\cdot y + y\cdot\overline{z} + \overline{x}\cdot\overline{z}\ .$$

Тупиковыми ДНФ являются:

$D_1 = \overline{x}\cdot\overline{y} + x\cdot z + y\cdot\overline{z};$

$D_2 = \overline{y}\cdot z + x\cdot y + \overline{x}\cdot\overline{z};$

$D_3 = \overline{x}\cdot\overline{y} + \overline{y}\cdot z + x\cdot y + y\cdot\overline{z};$

$D_4 = \overline{x}\cdot\overline{y} + x\cdot z + x\cdot y + \overline{x}\cdot\overline{z};$

$D_5 = \overline{y}\cdot z + x\cdot z + y\cdot\overline{z} + \overline{x}\cdot\overline{z}.$

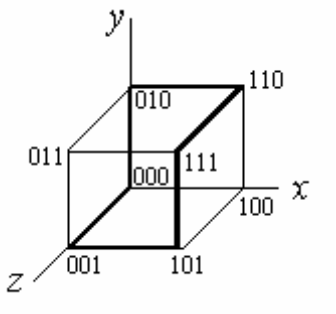


Рис. 2.6

ДНФ D_1 и D_2 являются минимальными для данной функции. ДНФ D_3, D_4, D_5 не являются минимальными. Таким образом, если сокращенная ДНФ строится однозначно, то процесс перехода от сокращенной ДНФ к тупиковой неоднозначен. Удаление одних элементарных конъюнкций из сокращенной ДНФ приводит к минимальной ДНФ, других – к тупиковой ДНФ, не являющейся минимальной, поэтому для построения минимальных ДНФ приходится строить все тупиковые ДНФ и среди них вести отбор. Процесс построения минимальных ДНФ на основе СДНФ или таблицы можно представить схемой, представленной на рис. 2.7.

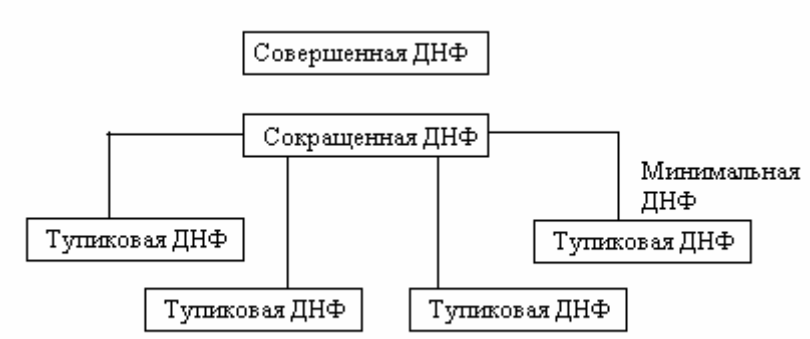


Рис. 2.7. Структура процесса построения минимальных ДНФ

Процедуру перехода от сокращенной ДНФ к тупиковой можно разбить на элементарные шаги, каждый из которых представляет собой удаление из ДНФ, полученной на предыдущем шаге, одной элементар-

ной конъюнкции K. Удаляемая конъюнкция такова, что $N_k \subseteq \bigcup_{j=1}^{m} N_{k_j}$, т. е. представляется суммой оставшихся интервалов. Для этого необходимо установить аналитический критерий покрытия некоторого интервала суммой других интервалов.

Определение. Дизъюнкция $D = K_1 + K_2 + \ldots + K_m$ элементарных конъюнкций поглощает элементарную конъюнкцию K, если $K \le D$, иначе дизъюнкция $D = \bigvee_{j=1}^{m} K_j$ поглощает конъюнкцию K тогда и только тогда, когда $N_k \subseteq \bigcup_{j=1}^{m} N_{k_j}$.

Определение. Элементарные конъюнкции K_1 *и* K_2 называются ортогональными, если $K_1 \cdot K_2 = 0$.

Геометрический смысл ортогональных конъюнкций прост. Они соответствуют интервалам, не имеющим общих вершин. Для проверки ортогональности проще всего пользоваться следующим свойством: две элементарные конъюнкции ортогональны тогда и только тогда, когда есть переменная, входящая в одну конъюнкцию с отрицанием, а во вторую – без отрицания.

Алгоритм проверки поглощения конъюнкции K дизъюнкцией $D = \bigvee_{j=1}^{m} K_j$ включает следующие шаги:

- в дизъюнкции $D = \bigvee_{j=1}^{m} K_j$ выделяются конъюнкции неортогональные K;
- из всех выделенных конъюнкций удаляются термы $x_i^{a_i}$, встречающиеся в K;
- выясняется, всегда ли равна 1 полученная после такого удаления дизъюнкция $K_1' + K_2' + \ldots + K_m'$. Если всегда, то конъюнкция K поглощается дизъюнкцией $D = \bigvee_{j=1}^{m} K_j$ и может быть вычеркнута из формулы.

Пример. Рассмотрим функцию $f(x, y, z)$ из предыдущего примера. Сокращенная ДНФ имеет вид

$$D(f) = \bar{y} \cdot z + y \cdot \bar{z} + \bar{x} \cdot \bar{y} + x \cdot z + x \cdot y + \bar{x} \cdot \bar{z}\,.$$

Выделим конъюнкцию $\overline{y}\cdot z$ и все неортогональные к ней конъюнкции: $\overline{x}\cdot\overline{y},\ x\cdot z$. Проверим выполнение условия $N_{\overline{y}z}=N_{\overline{xy}}\bigcup N_{xz}$. Для этого удаляем из $\overline{x}\cdot\overline{y}+x\cdot z$ термы $\overline{y}$ *и* z. Получаем дизъюнкцию $\overline{x}+x$, тождественно равную 1. Таким образом, конъюнкция $\overline{y}z$ поглощается дизъюнкцией $\overline{x}\cdot\overline{y}+x\cdot z$ и ее можно удалить из D. В результате имеем

$D'(f)=y\cdot\overline{z}+\overline{x}\cdot\overline{y}+x\cdot z+x\cdot y+\overline{x}\cdot\overline{z}$.

Во вновь полученной дизъюнкции $D'(f)$ выделим конъюнкцию $y\cdot\overline{z}$ и все неортогональные к ней конъюнкции: $x\cdot y,\ \overline{x}\cdot\overline{z}$. Удаляем из $x\cdot y+\overline{x}\cdot\overline{z}$ термы y и $\overline{z}$. Получаем $\overline{x}+x=1$. Конъюнкцию можно удалить из $D'(f)$. В результате получаем

$D''(f)=\overline{x}\cdot\overline{y}+x\cdot z+x\cdot y+\overline{x}\cdot\overline{z}$.

Полученная формула является тупиковой. Это несложно установить, проверив условия поглощения для каждой из оставшихся в D'' конъюнкций. Рассмотрим, например, конъюнкцию $\overline{x}\cdot\overline{y}$. Неортогональна к ней только конъюнкция $\overline{x}\cdot\overline{z}$. Однако условие $\overline{z}=1$ не является тождеством и, следовательно, $\overline{x}\cdot\overline{y}$ не поглощается D''.

Другой вариант состоит в последовательном исключении конъюнкций $\overline{y}\cdot z,\ x\cdot y,\ \overline{x}\cdot\overline{z}$. Такая последовательность операций ведет к форме $D'''(f)=\overline{x}\cdot\overline{y}+x\cdot z+y\cdot\overline{z}$, которая является минимальной.

2.5. Логика предикатов

2.5.1. Основные понятия логики предикатов

Многие утверждения, имеющие форму высказываний, на самом деле таковыми не являются, т. к. содержат переменные, конкретные значения которых не указаны. Поскольку такое утверждение при одних значениях переменных может быть истинным, а при других – ложным, ему не может быть предписано истинностное значение. Такие утверждения, примерами которых являются

$P(x)$: $\quad 3+x=5;$

$Q(x,y,z)$: $\quad x^2+y^2\geq z^2;$

$S(x)$: $\quad -1\leq\sin(x)\leq 1,$

называются **предикатами**. Логика предикатов представляет собой развитие логики высказываний.

Предикат – повествовательное предложение, содержащее предметные переменные, определенные на соответствующих множествах. При замене переменных конкретными значениями (элементами этих множеств) предложение обращается в высказывание, т. е. принимает значение «истина» или «ложь».

Предикат с одной переменной называется одноместным предикатом, с двумя – двухместным, а предикат, содержащий n переменных, называется n**–местным предикатом**.

n-местный предикат – это функция $P(x_1, x_2, \ldots, x_n)$ от n переменных, принимающих значения из некоторых заданных предметных областей, так, что $x_1 \in M_1, x_2 \in M_2, \ldots, x_n \in M_n$, а функция P принимает два логических значения – «истина» и «ложь». Таким образом, предикат $P(x_1, x_2, \ldots, x_n)$ является функцией типа $P\colon M_1 \times M_2 \times \ldots \times M_n \to B$, где множества $M_1, M_2, \ldots, M_n$ называются предметными областями предиката; $x_1, x_2, \ldots, x_n$ – предметными переменными предиката; B – двоичное множество. Если предикатные переменные принимают значения на одном множестве, то $P\colon M^n \to B$.

В качестве примера рассмотрим три высказывания:

A: «Рубль – валюта России»;

B: «Доллар – валюта России»;

C: «Доллар – валюта США».

Высказывания A и C истинны, высказывание B – ложно. Если вместо конкретных наименований валюты в выражениях A, B, C подставить предметную переменную x и определить ее на множестве наименований денежных единиц $x \in \{$ рубль, доллар, марка, крона и т. д. $\}$, то получим одноместный предикат, например $P(x)$: «x – валюта России».

Если в выражениях A, B, C (или аналогичных им) вместо конкретных наименований валюты и государства подставить переменные x и y, где $y \in \{$Россия, США, Англия, и т. д.$\}$, получим двухместный предикат $P(x, y)$: «x – валюта y». Приписав x и y конкретные значения, получим высказывание, обладающее свойством «истинно» или «ложно».

С помощью логических связок и скобок предикаты можно объединять в логические формулы – предикатные формулы. Исследование предикатных формул и способов установления их истинности является основным предметом логики предикатов. Логика предикатов является важным средством построения развитых логических языков и формальных систем (формальных теорий).

Логика предикатов, как и логика высказываний, может быть построена в виде алгебры логики предикатов и исчисления предикатов. Далее везде используется язык алгебры предикатов.

Пример. Предикат $x_1 > x_2$ – двухместный предикат, предметной областью которого могут служить любые множества действительных чисел. Высказывание 6 > 5 истинно, а высказывание 4>8 ложно.

Пример. Великая теорема Ферма, не доказанная до сих пор, утверждает, что для любого целого числа $n > 2$ не существует натуральных чисел x, y, z, удовлетворяющих равенству $x^n + y^n = z^n$. Если этому равенству поставить в соответствие предикат $P_F(x, y, z)$, истинный только тогда, когда равенство $x^n + y^n = z^n$ выполняется, а через $N(x)$ обозначить предикат «x – натуральное число», то теорема Ферма равносильна утверждению

«выражение $N(x) \wedge N(y) \wedge N(z) \wedge N(n) \wedge (n > 2) \to \overline{P}_F(x, y, z)$ истинно для любых чисел x, y, z, n».

Пример. Определим следующие предикаты:

Предикат тождества $E : N^2 \to B$. $E(a_1, a_2) = 1$ тогда и только тогда, когда $a_1 = a_2$.

Предикат делимости $D : N^2 \to B$. $D(a_1, a_2) = 1$ тогда и только тогда, когда a_1 делится на a_2.

Предикат суммы $S : N^3 \to B$. $S(a_1, a_2, a_3) = 1$ тогда и только тогда, когда $a_1 + a_2 = a_3$.

Тогда предикатные формулы

$(D(a,b) \wedge D(b,c)) \to D(a,c)$;

$S(a,b,c) \wedge D(a,d) \wedge D(b,d) \to D(c,d)$;

$\overline{D}(a,b) \wedge \overline{S}(a,b,c)$;

$S(a,b,c) \equiv S(b,a,c)$

имеют следующие словесные формулировки:

- «если a делится на b и b делится на c, то a делится на c»;
- «если каждое слагаемое a, b суммы целых чисел делится на некоторое число d, то и сумма c делится на это число»;
- «число a не делится на число b и неверно, что их сумма равна c»;
- «от перестановки мест слагаемых a и b сумма c не меняется».

2.5.2. Кванторы

Пусть $P(x)$ – предикат, определенный на M. Высказывание «для всех x из M $P(x)$ истинно» обозначается $\forall x P(x)$. Множество M не входит в обозначение и должно быть ясно из контекста. Знак $\forall x$ называется **квантором общности**.

Высказывание «существует такой x из M, что $P(x)$ истинно» обозначается как $\exists x P(x)$. Знак $\exists x$ называется **квантором существования**. Переход от $P(x)$ к $\forall x P(x)$ или $\exists x P(x)$ называется **связыванием** переменной x, а также навешиванием квантора на переменную x (или на предикат $P(x)$). Переменная, на которую навешан квантор, называется **связанной**, в противном случае – **свободной**.

Смысл связанных и свободных переменных в предикатных выражениях различен. Свободная переменная – это обычная переменная, которая может принимать различные значения из M; выражение $P(x)$ – переменное высказывание, зависящее от значения x. Выражение $\forall x P(x)$ не зависит от переменной x и при фиксированных P и M имеет вполне определенное значение. Переменные, являющиеся, по существу, связанными, встречаются не только в логике. Например, в выражениях $\sum_{x=1}^{10} f(x)$ или $\int_a^b f(x)dx$ переменная x связана. При фиксированной f первое выражение равно определенному числу, а второе – функции от a и b.

Навешивать кванторы можно и на многоместные предикаты и вообще на любые выражения, которые при этом заключаются в скобки. Выражение, на которое навешивается квантор $\forall x$ или $\exists x$, называется **областью действия квантора.** Навешивание квантора на многоместный предикат уменьшает в нем число свободных переменных и превращает его в предикат от меньшего числа переменных.

Пример. Пусть $P(x)$ – предикат «x – четное число». Тогда высказывание $\forall x P(x)$ истинно на любом множестве четных чисел и ложно, если M содержит хотя бы одно нечетное число. Высказывание $\exists x P(x)$ истинно на любом множестве, содержащем хотя бы одно четное число, и ложно на любом множестве нечетных чисел.

Пример. Теорема Ферма формулируется с помощью кванторов следующим образом:

$$\forall x \forall y \forall z \forall n (N(x) \wedge N(y) \wedge N(z) \wedge N(n) \wedge (n > 2) \to \overline{P}_F(x, y, z)).$$

Пример. Пусть предикат $P(x,y)$ описывает отношение «x любит y» на множестве людей. Для разных вариантов навешивания кванторов на обе переменные словесные интерпретации полученных формул будут различны:

$\forall x\exists yP(x,y)$ – «для любого x существует y, которого он любит»;

$\exists y\forall xP(x,y)$ – «существует такой y, которого любят все x»;

$\forall y\forall xP(x,y)$ – «все люди любят всех людей»;

$\exists y\exists xP(x,y)$ – «существует человек, который кого-то любит»;

$\exists x\forall yP(x,y)$ – «существует человек, который любит всех»;

$\forall y\exists xP(x,y)$ – «каждого человека кто-то любит».

2.5.3. Выполнимость и истинность

При логической интерпретации формул логики предикатов возможны три основные ситуации:

1. Если в области M для формулы F существует такая подстановка констант вместо всех переменных, что F становится истинным высказыванием, то формула F называется **выполнимой в области М.**

2. Если формула F выполнима в M при любых подстановках констант, то она называется **тождественно истинной в** M. Формула, тождественно истинная в любых M, называется тождественно истинной, или **общезначимой**.

Пример. Формула $\exists x(P(x)\vee\overline{P}(x))$ тождественно истинна.

3. Если формула F невыполнима в M, то она называется **тождественно ложной в** M. Если формула невыполнима ни в каких M, она называется тождественно ложной, или **противоречивой**.

Пример. Формула $\exists x(P(x)\wedge\overline{P}(x))$ тождественно ложна.

Определение. Моделью $\mathfrak{R}$ в логике предикатов называют множество M вместе с заданной на нем совокупностью предикатов $\Sigma=\{P_1,P_2,\ldots,P_k\}$: $\mathfrak{R}=(M;P_1,P_2,\ldots,P_k)$, где M – основное множество модели $\mathfrak{R}$; $\Sigma=\{P_1,P_2,\ldots,P_k\}$ – сигнатура модели $\mathfrak{R}$.

Пример. Определить истинность, ложность или выполнимость на модели $\mathfrak{R}=\{N;S,П,E\}$ следующих формул:

$(П(x,y,z)\wedge П(x,y,u))\to E(z,u)$;

$\exists yП(x,x,y)$;

$\exists xП(x,x,y)$.

Здесь S,Π,E – предикаты суммы, произведения и равенства.

Первая предикатная формула является тождественно истинной на модели $\Re$ ввиду единственности значения произведения чисел из N. При любых подстановках констант формула истинна.

Вторая формула на модели $\Re$ выражает существование натурального квадрата натурального числа x. Она также тождественно истинна на модели $\Re$.

Третья формула выполнима на модели $\Re$. Она читается так: «существует натуральное значение квадратного корня для натурального числа y из N», или «$\sqrt{y}$ – натуральное число». Очевидно, что она истинна при подстановках вместо y чисел 0, 1, 4, 9, 16, … и ложна при подстановке 2, 3, 5, …

2.5.4. Префиксная нормальная форма

Формулы называются **эквивалентными**, если при любых подстановках констант они принимают одинаковые значения. В частности, все тождественно истинные (или тождественно ложные) формулы эквивалентны.

Множество истинных формул логики предикатов входит в любую теорию, и, следовательно, его исследование является важнейшей целью логики предикатов. В этом исследовании прежде всего возникают две проблемы: получение истинных формул и проверка формулы на истинность.

Те же проблемы имеют место и в логике высказываний. Но там есть стандартная разрешающая процедура: вычисление формул на наборах значений переменных (построение таблиц истинности). С ее помощью порождающую процедуру для множества M_m тождественно истинных высказываний можно реализовать следующим образом: строим последовательно все формулы, вычисляем каждую из них на всех наборах и включаем в M_m только те, которые истинны на всех наборах.

Аналогичная процедура в логике предикатов сталкивается с большими трудностями, связанными с тем, что предметные переменные имеют в общем случае бесконечные области определения, поэтому прямой перебор всех значений, как правило, невозможен. Приходится использовать приемы, базирующиеся на эквивалентных соотношениях. Приведем в качестве примера основные из них.

Соотношения

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

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

определяют правила вынесения кванторов из под отрицаний и являются фактически законами де Моргана для кванторов.

Формулы

$$\forall x P_1(x) \wedge \forall x P_2(x) \equiv \forall x (P_1(x) \wedge P_2(x)); \tag{2.3}$$

$$\exists x P_1(x) \vee \exists x P_2(x) \equiv \exists x (P_1(x) \vee P_2(x)) \tag{2.4}$$

выражают дистрибутивные законы для кванторов. Свойство (2.3) можно проиллюстрировать следующим высказыванием: «Все – летчики и все – герои. Каждый и летчик, и герой». А свойство (2.4) – «В данном слове есть буква А, или в данном слове есть буква Б. В данном слове есть буква А или Б».

Если же $\exists$ и $\forall$ в этих выражениях поменять местами, то получатся соотношения, верные лишь в одну сторону:

$$\exists x (P_1(x) \wedge P_2(x)) \to \exists x P_1(x) \wedge \exists x P_2(x); \tag{2.5}$$

$$(\forall x P_1(x) \vee \forall x P_2(x)) \to \forall x (P_1(x) \vee P_2(x)). \tag{2.6}$$

Для справедливости выражение (2.6) необходимо, чтобы в левой части хотя бы один предикат выполнялся для всех x, для правой же достаточно, чтобы один предикат был истинен там, где другой ложен.

В таких случаях говорят, что левая часть более сильное утверждение, чем правая, поскольку она требует для своей истинности выполнения более жестких условий. Так, в (2.5) в левой части требуется, чтобы $P_1(a)$ и $P_2(a)$ были истинны для одного и того же a, тогда как в правой части P_1 и P_2 могут быть истинны при различных a_1 и a_2.

Например, высказывание «Наша Света – красавица, спортсменка отличница» более сильное, чем высказывание «Есть у нас красавицы, спортсменки, отличницы», потому что второе не обязательно означает, что перечисленными качествами обладает одно лицо. Еще один пример, когда выражение (2.5) в обратную сторону неверно: $P_1(x)$ – «x – четное число», $P_2(x)$ – «x – нечетное число».

Законы коммутативности выполняются лишь для одноименных кванторов:

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

$\exists x \exists y P(x, y) \equiv \exists y \exists x P(x, y)$.

Разноименные кванторы в общем случае не коммутативны (см. пример «x любит y» в разд. 2.5.2).

Приведем без доказательства еще несколько соотношений:

$\forall x(P(x) \wedge Y) \equiv \forall x P(x) \wedge Y$;

$\forall x(P(x) \vee Y) \equiv \forall x P(x) \vee Y$;

$\exists x(P(x) \wedge Y) \equiv \exists x P(x) \wedge Y$;

$\exists x(P(x) \vee Y) \equiv \exists x P(x) \vee Y$.

Последние соотношения означают, что формулу, не содержащую x, можно выносить за область действия квантора, связывающего x.

Как и в логике высказываний, в логике предикатов существуют эквивалентные нормальные формы представления любых предикатных формул.

Определение [5]. Префиксной нормальной формой (ПНФ) называется формула, имеющая вид

$$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n F,$$

где $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n$ – кванторы, F – формула, не имеющая кванторов, с операциями $\{\wedge, \vee, -\}$. В логике предикатов для любой формулы существует эквивалентная ей ПНФ.

Процедура получения ПНФ включает следующие этапы [5]:

1. Используя формулы

$P_1 \equiv P_2 = (P_1 \to P_2) \wedge (P_2 \to P_1)$;

$P_2 \to P_1 = \overline{P}_1 \vee P_2$,

заменить операции $\{\to, \equiv\}$ на $\{\wedge, \vee, \neg\}$.

2. Воспользовавшись выражениями замены кванторов, а также правилом двойного отрицания и правилом де Моргана

$\overline{\overline{P}} = P$;

$\overline{(P_1 \vee P_2)} = \overline{P}_1 \wedge \overline{P}_2$;

$\overline{(P_1 \wedge P_2)} = \overline{P}_1 \vee \overline{P}_2$,

представить предикатную формулу таким образом, чтобы символы отрицания были расположены непосредственно перед (над) символами предикатов.

3. Для формул, содержащих подформулы вида

$\forall x P_1(x) \vee \forall x P_2(x), \quad \exists x P_1(x) \wedge \exists x P_2(x)$,

ввести новые переменные.

4. С помощью формул эквивалентных преобразований получить формулы в виде ПНФ.

Пример. Привести к ПНФ следующую предикатную формулу: $\overline{(\exists x \forall y P_1(x, y)) \wedge (\exists x \forall y P_2(x, y))}$.

Применив правило де Моргана, получим

$$\overline{(\exists x \forall y P_1(x,y)) \wedge (\exists x \forall y P_2(x,y))} \equiv \overline{(\exists x \forall y P_1(x,y))} \vee \overline{(\exists x \forall y P_2(x,y))}.$$

Далее, перенесем кванторы через отрицание:

$$\overline{(\exists x \forall y P_1(x,y))} \vee \overline{(\exists x \forall y P_2(x,y))} \equiv \forall x \exists y \overline{P_1(x,y)} \vee \forall x \exists y \overline{P_2(x,y)}.$$

Так как квантор общности $\forall x$ не дистрибутивен относительно дизъюнкции, поменяем в каком-либо предикате, например во втором, переменную x на новую переменную z:

$$\forall x \exists y \overline{P_1(x,y)} \vee \forall x \exists y \overline{P_2(x,y)} \equiv \forall x \exists y \overline{P_1(x,y)} \vee \forall z \exists y \overline{P_2(z,y)}.$$

Воспользовавшись дважды эквивалентным отношением выноса функции, не зависящей от x, из под кванторов $\exists x, \forall x$, получим

$$\forall x \exists y \overline{P_1(x,y)} \vee \forall z \exists y \overline{P_2(z,y)} \equiv \forall x \forall z (\exists y \overline{P_1(x,y)} \vee \exists y \overline{P_2(z,y)}).$$

Поскольку квантор существования $\exists y$ дистрибутивен относительно дизъюнкции, окончательно получим

$$\forall x \forall z (\exists y \overline{P_1(x,y)} \vee \exists y \overline{P_2(z,y)}) \equiv \forall x \forall z \exists y (\overline{P_1(x,y)} \vee \overline{P_2(z,y)}).$$

Глава 3
ГРАФЫ И СЕТИ

3.1. Графы

Графические представления в широком смысле – любые наглядные отображения исследуемой системы, процесса, явления на плоскости. К ним могут быть отнесены рисунки, чертежи, графики зависимостей, блок-схемы процессов, диаграммы и т. д. Такие изображения наглядно представляют различные взаимосвязи и взаимообусловленности: топологическое (пространственное) расположение объектов, хронологические (временные) зависимости процессов и явлений, логические, структурные, причинно–следственные (каузальные) и другие взаимосвязи.

Основное достоинство графических представлений – наглядность и соответственно возможность быстрого анализа ситуации. Количественные характеристики гораздо быстрее воспринимаются в форме двухтрехмерных гистограмм или круговых диаграмм, чем в форме таблиц. Точно также процессы лучше воспринимаются в форме структурных схем, чем в форме вербальных алгоритмов.

Мощным и наиболее исследованным классом объектов, относящимся к графическим представлениям, являются графы. Теория графов имеет обширные приложения, т. к. ее язык, с одной стороны, нагляден и понятен, с другой – удобен в формальном исследовании. Например, структура молекулы является графом, в котором вершинами являются атомы, а ребрами – валентные связи. Блок-схема алгоритма представляет собой ориентированный граф, в котором вершинами являются отдельные операторы, а дуги указывают переходы между ними. Другие примеры графов: элементы и соединения в электрической цепи, схема перекрестков и дорог, множество предприятий с указанием двухсторонних связей между ними, группа людей с указанием их психологической совместимости, структура управления с указанием объектов и их подчиненности друг другу и т. д.

На языке теории графов формулируются и решаются многие задачи управления, в том числе задачи сетевого планирования и управления, анализа и проектирования организационных структур, анализа процессов функционирования динамических систем.

3.1.1. Основные определения теории графов

Графом G называется совокупность двух множеств: вершин V и ребер E, между элементами которых определено отношение **инцидентности** – каждое ребро $e \in E$ инцидентно ровно двум вершинам $v', v'' \in V$, которые оно соединяет. При этом вершина v' (v'') и ребро e называются **инцидентными** друг другу, а вершины v', v'', являющиеся для ребра концевыми точками, называются **смежными**. Именно отношение инцидентности является самым важным элементом графа, т. к. определяет способ объединения множеств V и E в единый графический объект.

Ребро, соединяющее две вершины, может иметь направление от одной вершины к другой; в этом случае оно называется **направленным**, или **ориентированным**, или **дугой,** и изображается стрелкой, направленной от одной вершины (начала) к другой (концу). Граф, содержащий дуги, называется ориентированным, или **орграфом**. Граф, содержащий только ребра, называется неориентированным, или **н-графом**.

Пример.

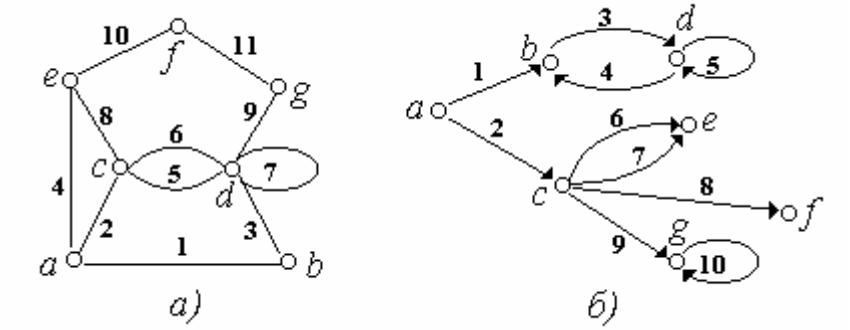


Рис. 3.1. *а* – н-граф; *б* – орграф

Ребра, инцидентные одной и той же паре вершин, называются **кратными**. Граф, содержащий кратные ребра, называется **мультиграфом**. Ребро, концевые вершины которого совпадают, называется **петлей**.

Вершина, не инцидентная ни одному ребру, называется **изолированной**. Граф может состоять только из изолированных вершин. В этом случае он называется **нуль-графом**.

Граф без петель и кратных ребер называется **полным**, если каждая пара вершин соединена ребром.

Пример

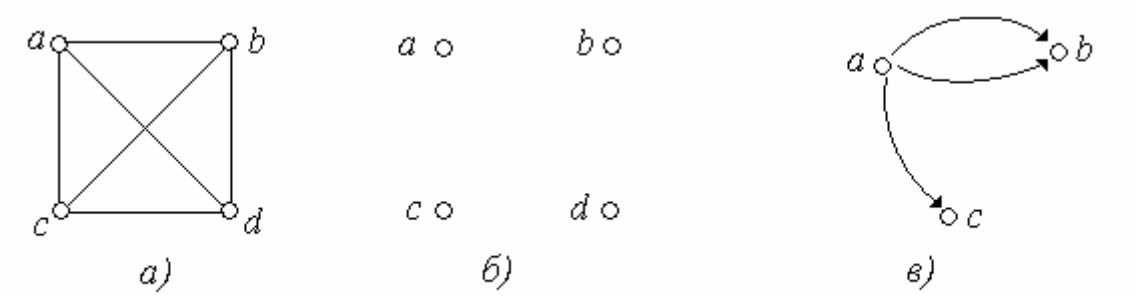


Рис. 3.2. *а* – полный граф; *б* – нуль-граф; *в* – мультиграф

Дополнением графа G называется граф $\overline{G}$, имеющий те же вершины, что и граф G, и содержащий только те ребра, которые нужно добавить к графу G, чтобы получить полный граф.

Пример

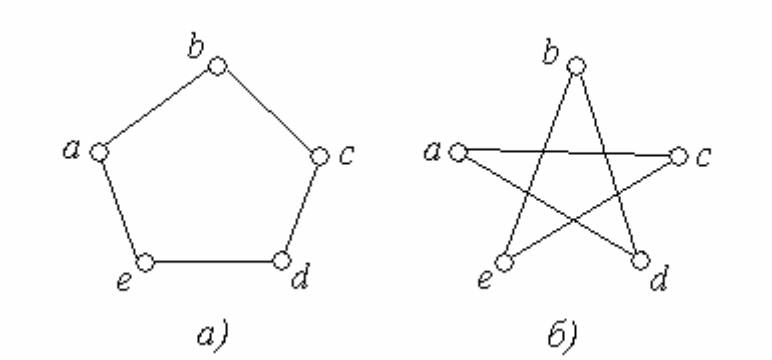


Рис. 3.3. *а* – граф *G*; *б* – дополнение графа *G*

Каждому неориентированному графу соответствует ориентированный граф с тем же множеством вершин, в котором каждое ребро заменено двумя дугами, инцидентными тем же вершинам и имеющим противоположные направления.

Локальной степенью (или просто степенью) вершины $v \in V$ н-графа G называется количество ребер $\rho(v)$, инцидентных вершине v. В н-графе сумма степеней всех вершин равна удвоенному числу ребер m графа, т. е. четна, если считать, что петля дает вклад 2 в степень вершины:

$$\sum_{v \in V} \rho(v) = 2m .$$

Сумма степеней вершин любого графа равна удвоенному числу его ребер. Отсюда следует, что в н-графе число вершин нечетной степени четно.

Пример. Н-граф на рис. 3.1, *а* имеет две вершины нечетной степени (вершины a и e). Степени остальных вершин четны.

Для вершин орграфа определяются две локальные степени:

$\rho_1(v)$ – количество дуг, исходящих из вершины v;

$\rho_2(v)$ – количество дуг, входящих в вершину v.

Петля дает вклад 1 в обе эти степени.

В орграфе суммы степеней всех вершин $\rho_1(v)$ и $\rho_2(v)$ равны количеству дуг m этого графа и равны между собой:

$$\sum_{v\in V} \rho_1(v) = \sum_{v\in V} \rho_2(v) = m.$$

Пример. Для орграфа на рис. 3.1, *б* локальные степени вершин равны: $\rho_1(a)=3$, $\rho_1(b)=0$, $\rho_1(c)=0$, $\rho_2(a)=0$, $\rho_2(b)=2$, $\rho_2(c)=1$.

3.1.2. Способы задания графов

Наиболее простым и естественным способом задания графа является графический. Однако таким образом можно задать только небольшие графы, к тому же он неудобен для автоматизированной обработки и передачи графической информации. Рассмотрим другие способы, используемые в теории графов.

В общем виде задать граф – значит описать множества его вершин и ребер, а также отношение инцидентности. Для описания вершин и ребер их достаточно занумеровать. Пусть $v_1, v_2, \ldots, v_n$ – вершины графа G; $e_1, e_2, \ldots, e_m$ – ребра. Отношение инцидентности может быть задано следующими способами:

1. Матрицей инцидентности $\|\varepsilon_{ij}\|$ размера $n \times m$. По вертикали и горизонтали указываются вершины и ребра соответственно, а на пересечении i-й вершины и j-го ребра, в случае неориентированного графа, проставляется 1, если они инцидентны, и 0 – в противоположном случае, т. е.

$$\varepsilon_{ij} = \begin{cases} 1, \text{ если ребро } e_i \text{ инцидентно вершине } v_j; \\ 0 - \text{ в противном случае,} \end{cases}$$

а в случае орграфа: –1, если вершина является началом дуги; 1 – если вершина является концом дуги и 0 – если вершины не инцидентны. Ес-

ли некоторая вершина является для ребра и началом и концом (т. е. ребро – петля), проставляется любое другое число, например 2.

2. Списком ребер графа, представленным двумя столбцами: в левом перечисляются все ребра $e \in E$, в правом – инцидентные им вершины $(v',\ v'')$. Для н-графа порядок вершин произволен, для орграфа первым стоит номер начала дуги. При наличии в графе изолированных вершин они помещаются в конец списка.

3. Матрицей смежности $\|\delta_{kl}\|$ – квадратной матрицей размера $n \times n$. По вертикали и горизонтали перечисляются все вершины, а на пересечении k-й и l-й вершин, в случае н-графа, проставляется число δ_{kl}, равное числу ребер, соединяющих эти вершины. Для орграфа δ_{kl} равно числу ребер с началом в k-й вершине и концом в l-й вершине.

Пример. Задать различными способами графы, представленные на рис. 3.4.

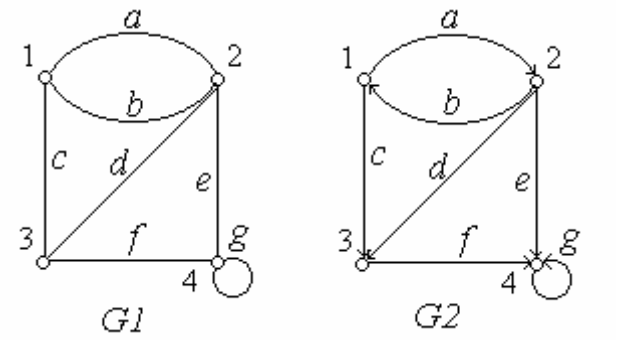


Рис. 3.4

Матрицы инцидентности графов имеют вид

*G*1	*a*	*b*	*c*	*d*	*e*	*f*	*g*
1	1	1	1				
2	1	1		1	1		
3			1	1		1	
4					1	1	1

*G*2	*a*	*b*	*c*	*d*	*e*	*f*	*g*
1	–1	1	–1				
2	1	–1		–1	–1		
3			1	1		–1	
4					1	1	2

Список ребер является более компактным описанием графа:

Ребро	Вершины
a	1 2
b	2 1
c	1 3
d	2 3

e	2 4
f	3 4
g	4 4

Следующие таблицы представляют матрицы смежности графов $G1$ и $G2$:

$G1$	1	2	3	4
1		2	1	
2	2		1	1
3	1	1		1
4		1	1	1

$G2$	1	2	3	4
1		1	1	
2	1		1	1
3				1
4				1

3.1.3. Операции над частями графа

Определение. Граф H называется **частью графа** G (или частичным графом), $H \subset G$, если множества его вершин $V(H)$ и ребер $E(H)$ содержатся в множествах $V(G)$ и $E(G)$ соответственно.

Пример. Пусть задан н-граф – схема железных дорог РФ. Здесь роль вершин играют железнодорожные станции, роль ребер – перегоны. Частичным графом можно считать схему электрифицированных железных дорог РФ.

Рассмотрим некоторые частные случаи частичных графов.

Если $V(H) = V(G)$, то граф H называется **суграфом** графа G. Суграф H содержит все те же вершины, что и граф G, но отличается от него количеством ребер. Например, нулевой суграф схемы железных дорог РФ содержит только железнодорожные станции.

Подграфом $G(V')$ графа $G(V)$ с множеством вершин V' называется часть графа, которой принадлежат все ребра с обоими концами из V'.

Пример. Схема железных дорог Томской области является суграфом схемы железных дорог РФ.

Над частями графа G могут производиться следующие операции:

Дополнение $\bar{H}$ к части H графа G определяется множеством всех ребер графа G, не принадлежащих H: $E(H) \cap E(\bar{H}) = \varnothing$, $E(H) \cup E(\bar{H}) = E(G)$.

Сумма $H_1 \cup H_2$ частей H_1 и H_2 графа G:

$V(H_1 \cup H_2) = V(H_1) \cup V(H_2)$ и $E(H_1 \cup H_2) = E(H_1) \cup E(H_2)$.

Произведение $H_1 \cap H_2$ частей H_1 и H_2 графа G:

$V(H_1 \cap H_2) = V(H_1) \cap V(H_2)$ и $E(H_1 \cap H_2) = E(H_1) \cap E(H_2)$.

3.1.4. Маршруты, пути, цепи, циклы

Пусть G – неориентированный граф. **Маршрутом** в G называется такая последовательность ребер $(e_1, e_2, \ldots, e_n)$, в которой каждые два соседних ребра – e_{i-1} и e_i – имеют общую вершину. В маршруте одно и то же ребро может встречаться несколько раз. Вершина v_0 – начало маршрута. Она инцидентна e_1 и не инцидентна e_2.

Маршрут, у которого начало и конец совпадают, называется **циклическим**. Маршрут, в котором все ребра разные, называется **цепью**. Цепь, не пересекающая себя, т. е. не имеющая повторяющихся вершин, называется **простой цепью**.

Циклический маршрут называется **циклом**, если он является цепью, и простым циклом, когда это **простая цепь**.

Вершины v', v'' называются связанными, когда существует маршрут M с началом v' и концом v''. Связанные маршрутом вершины связаны также и простой цепью. Отношение связности вершин обладает свойствами эквивалентности (см. разд. 1.2.3) и определяет разбиение множества вершин графа на непересекающиеся подмножества V_i, где $i = 1,2,\ldots,k$. Граф G называется **связным**, если все его вершины связаны между собой. Поэтому все подграфы $G(V_i)$ связны и называются связными компонентами графа. Каждый н-граф распадается единственным образом в прямую сумму своих связных компонент $G = \bigcup_i G(V_i)$.

Пример. Для вершин 1 и 6 графа G, приведенного на рис. 3.5, привести примеры маршрута, цепи, простой цепи; определить на графе циклический маршрут, цикл, простой цикл, приняв вершину 1 за их начало и конец.

Маршрутом является последовательность ребер – $e1, e2, e3, e4, e5, e6, e7, e8, e1, e6$. Цепью – $e1, e2, e3, e4, e5, e6, e7$.

Простой цепью – $e1, e2, e3, e4$.

Циклическим маршрутом (и одновременно циклом) является последовательность ребер $e1, e2, e3, e4, e5, e6, e7, e8$.

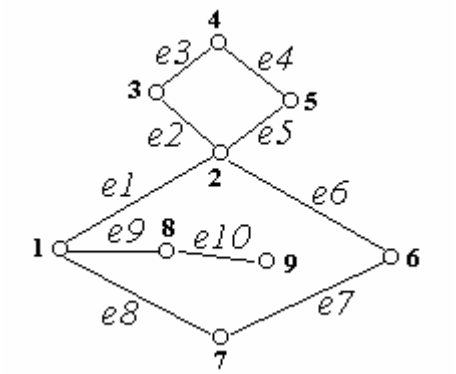


Рис. 3.5

Простым циклом – $e1, e6, e7, e8$.

Пусть G – ориентированный граф. Последовательность дуг, в которой конец каждой предыдущей дуги e_{i-1} совпадает с началом следующей – e_i, называется **путем**. (Дуги проходят по направлениям

их ориентации). В пути одна дуга может встречаться несколько раз.

Путь называется **ориентированной цепью** (или просто цепью), если каждая дуга встречается не более одного раза, и **простой цепью**, если не имеет повторяющихся вершин.

Замкнутый путь называется **контуром**. Контур называется **циклом**, если он является цепью, и простым циклом, когда это **простая цепь**.

Вершина v' называется **достижимой** из вершины v'', если существует путь $L(v',\ldots,v'')$.

Орграф G называют **связным**, если он связен без учета ориентации дуг, и **сильно связным**, если из любой его вершины в любую другую существует путь.

Число ребер (дуг) маршрута (пути) называется его **длиной**.

Расстоянием $d(v',v'')$ между вершинами v' и v'' н-графа G называется минимальная длина простой цепи между v' и v''. **Центром** называется вершина н-графа, от которой максимальное из расстояний до других вершин минимально. Максимальное расстояние от центра графа до его вершин называется **радиусом** графа G.

Пример. Определить, какой из приведенных на рис. 3.6 орграфов является связным? Какой из них является сильно связным?

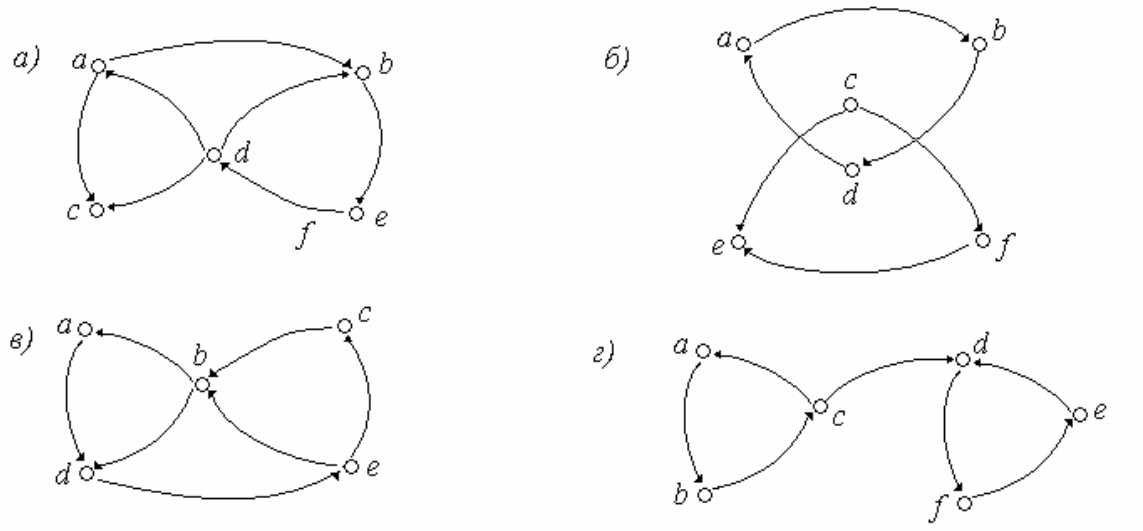


Рис. 3.6

Ответ. Связными являются графы *а, в, г*. Граф *б* несвязен и включает два компонента.

Граф *а* не является сильно связным, т. к., например, из вершины *с* нельзя выйти, двигаясь по направлениям дуг. Граф *г* также не является

сильно связным, так как из его правой части (вершины d, f, e) нельзя попасть в левую. Граф *в* сильно связен, т. к. не содержит недостижимых вершин.

Пример. Для связного н-графа на рис. 3.7 определить расстояния между вершинами. Какая вершина является центром графа? Чему равен радиус графа?

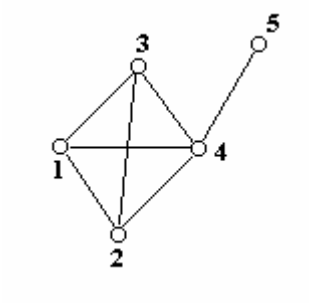


Рис. 3.7

Расстояния между вершинами графа:
(1,3) – 1; (1,2) – 1; (1,4) – 1; (1,5) – 2; (2,3) – 1; (2,4) – 1; (2,5) – 2; (3,4) – 1, (3,5) – 2, (4,5) – 1.

Центром графа является вершина 4, т. к. для нее максимальное расстояние от всех других вершин минимально и равно 1. Радиус графа равен 1.

3.1.5. Эйлеровы циклы и цепи

Эйлеров цикл – цикл, содержащий все ребра графа. Эйлеров граф – граф, имеющий эйлеров цикл.

Понятие эйлерова цикла связано с известной задачей Л. Эйлера о Кенигсбергских мостах на реке Прегал. Схема этих мостов, соединяющих берега реки 2,3 с двумя ее островами 1,4 и острова между собой, приведена на рис. 3.8, *а*. Эйлер сформулировал эту задачу следующим образом: можно ли, начав с некоторой точки, пройти все мосты по одному разу и вернуться в исходный пункт. Для решения задачи Эйлер преобразовал рисунок в граф, обозначив берега реки и острова как вершины графа, а мосты как ребра графа (рис. 3.8, *б*).

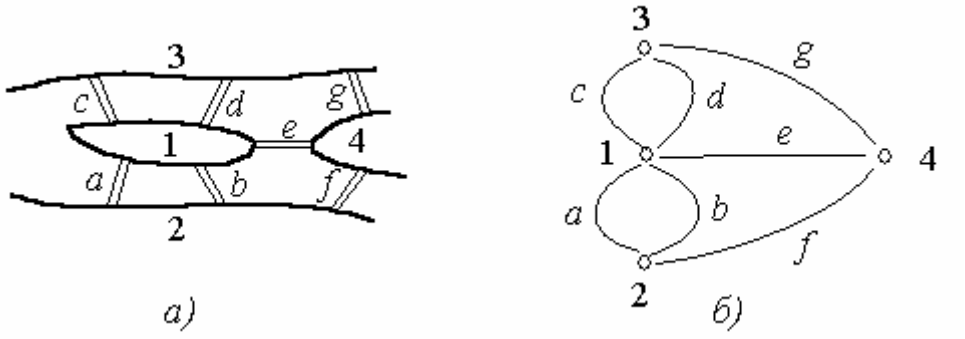


Рис. 3.8. Схема и граф для задачи о Кенигсбергских мостах

Легко видеть, что в такой постановке задача о Кенигсбергских мостах эквивалентна задаче определения на графе такого цикла (цикличе-

ского маршрута, не содержащего повторяющихся ребер), который проходит через все ребра графа. Ее решение дается следующей теоремой.

Теорема Эйлера: конечный неориентированный граф G эйлеров тогда и только тогда, когда он связен и степени всех его вершин четны.

Необходимость этих условий достаточно очевидна. В несвязном графе каждый цикл принадлежит какой-либо его связной части, т. е. не проходит через все его части. С другой стороны, каждый раз, когда эйлеров цикл приходит в какую-либо вершину, он должен выйти из нее по другому ребру, т. е. инцидентные вершинам ребра можно разбить на пары соседних в эйлеровом цикле. Это относится и к начальной вершине, которая одновременно является и конечной. Для нее инцидентные ребра также распадаются на пары, но, кроме того, есть ребро, относящееся к началу цикла, и ребро, относящееся к концу цикла.

Близкой по смыслу к задаче поиска эйлерова графа является так называемая **задача графопостроителя**. Она может быть сформулирована следующим образом: заданный плоский граф необходимо вычертить, не отрывая пера от бумаги и не обводя дважды одно и то же ребро.

Задача графопостроителя не эквивалентна задаче о Кенигсбергских мостах и допускает решение не только в классе эйлеровых циклов, т. к. не требует, чтобы головка графопостроителя возвращалась в исходную точку. Решение может лежать и в классе **эйлеровых цепей**.

Эйлерова цепь – цепь, включающая все ребра данного н-графа G, и имеющая различные начальную и конечную вершины.

Теорема. Чтобы в конечном н-графе существовала эйлерова цепь, необходимы и достаточны его связность и четность локальных степеней всех вершин, кроме начальной и конечной, которые должны иметь нечетные степени.

Рассмотрим некоторые примеры.

Пример. Схема и граф для задачи о Кенигсбергских мостах приведены на рис. 3.8.

В графе все вершины имеют нечетные степени. Следовательно, граф не имеет эйлерова цикла и решение поставленной задачи невозможно.

Пример. Имеют ли пятиугольник и пятигранник – пирамида, приведенные на рис. 3.9, эйлеров цикл (цепь)?

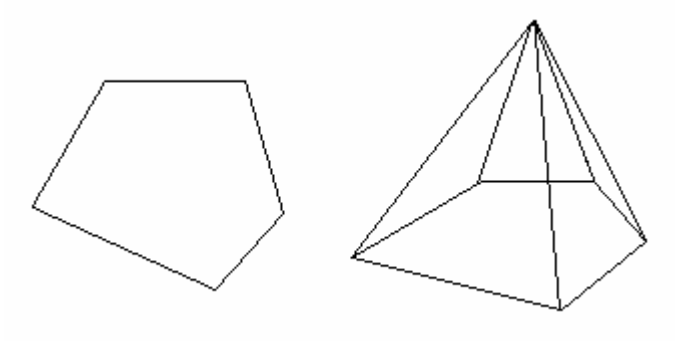


Рис. 3.9

Локальная степень каждой вершины пятиугольника равна двум, т. е. четна, соответственно пятиугольник – эйлеров граф.

Пятигранник–пирамида имеет нечетные степени всех вершин и не является эйлеровым графом.

Пример. Найти эйлеров цикл для графа, представленного на рис. 3.10, *а*.

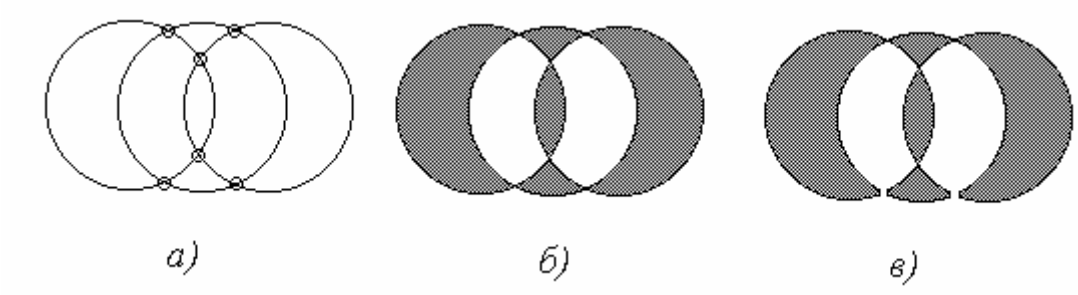


Рис. 3.10. Нахождение эйлерова цикла

Граф связный и имеет 6 вершин, все четные. Следовательно, данный граф – эйлеров и может быть нарисован одним росчерком пера. Откуда же начинать вычерчивание?

Существует следующий способ определения порядка вычерчивания: в графе следует выбрать одну область и заштриховать ее; область, граничащую с заштрихованной, пропустить, а имеющую лишь общую вершину – заштриховать, и так действовать до тех пор, пока все возможные области не будут заштрихованы (рис. 3.10, *б*).

Далее заштрихованный граф следует разъединить в одной или нескольких вершинах так, чтобы образовалась односвязная (без «дыр») заштрихованная область (рис. 3.10, *в*). Таким образом, теория графов дала не только условия разрешимости задачи, но и конструктивный метод ее решения.

3.1.6. Обобщенная теорема об эйлеровых цепях

Сформулируем еще одну теорему, являющуюся обобщением теоремы Эйлера.

Теорема. Если граф G является связным и имеет $2k$ вершин нечетной степени, то в нем можно выделить k различных цепей, содержащих все его ребра в совокупности ровно по одному разу.

Доказательство. Обозначим нечетные вершины графа $A_1, A_2, \ldots, A_k, B_1, \ldots, B_k$. Соединив A_i и B_i для всех i, добавим к графу k ребер $A_1, B_1; A_2, B_2; \ldots; A_k, B_k$. В новом графе все вершины четные, следовательно, должен существовать эйлеров цикл. Если теперь выбросить из найденного эйлерова цикла новые ребра, то цикл распадется на k отдельных цепей, содержащих все ребра графа.

На методике доказательства может быть построен конструктивный алгоритм определения эйлеровых цепей на графе. Рассмотрим пример. Пусть задана схема трамвайных путей, которая может быть представлена графом на рис. 3.11. Требуется выбрать маршруты движения так, чтобы можно было добраться до любой точки схемы, делая пересадки лишь в точках, являющихся вершинами графа. Методика решения последовательно показана на рис. 3.11, *а* – 3.11, *г*.

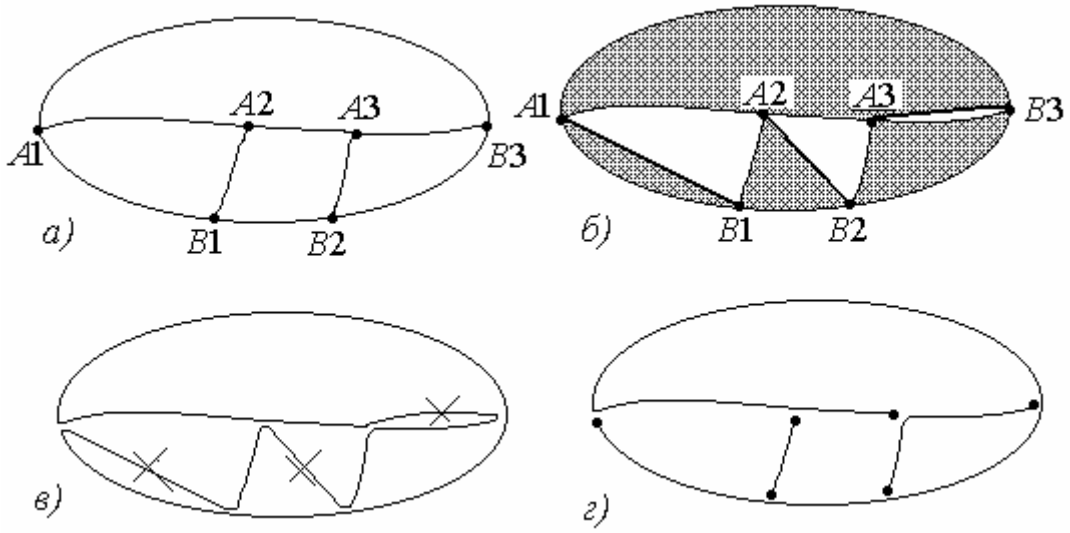


Рис. 3.11

Пример. По шахматной доске ходит конь. Каждый его возможный ход может рассматриваться как связь между двумя клетками доски или как ребро графа, вершинами которого являются клетки доски. Можно ли проложить такой маршрут, чтобы конь сделал все ходы, не повторив ни одного из них дважды? Можно ли покрыть все клетки набором цепей, сколько их понадобится? Решить тот же пример для короля.

Для решения данной задачи нужно представить шахматную доску в виде графа, приписав каждой клетке локальную степень, равную числу различных ходов, которые может сделать из этой клетки конь. На рис. 3.12, *а* представлена четверть шахматной доски. Число в клетке – количество различных ходов коня из этой клетки.

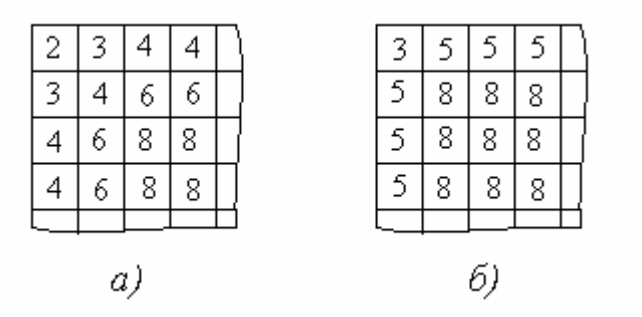


Рис. 3.12

Из рис. 3.12, *а* следует, что для коня граф имеет 8 вершин нечетной степени и, соответственно, не является эйлеровым. Используя обобщенную теорему Эйлера, можно сделать вывод, что данный граф можно покрыть четырьмя эйлеровыми цепями.

Рис. 3.12, *б* соответствует ходам шахматного короля. Построенный по этим данным граф также не является эйлеровым и может быть покрыт четырнадцатью эйлеровыми цепями.

Интересная и практически важная задача, имеющая смысл поиска эйлерова цикла на ориентированном графе, может быть сформулирована следующим образом: проложить маршрут осмотра экспонатов выставки, размещенных по обеим сторонам коридоров некоторого помещения. Очевидно, что данная задача имеет смысл поиска такого эйлерова пути на ориентированном графе, чтобы каждое ребро было пройдено дважды в разных направлениях. Разрешимость данной задачи определяется следующей теоремой.

Теорема. Чтобы в конечном ориентированном графе G существовал эйлеров цикл, необходимо и достаточно равенство степеней всех вершин этого графа по входящим и выходящим дугам $\forall v \in G \quad (\rho_1(v) \quad \rho_2(v))$.

Учитывая, что любому н-графу канонически соответствует орграф, можно сформулировать следующее утверждение: в конечном связном н–графе всегда можно построить ориентированный цикл, проходящий через каждую дугу (проходящий через каждое ребро по одному разу в каждом из двух направлений).

3.1.7. Гамильтонов цикл. Взвешенные графы

Еще одной практически важной группой задач, решаемых с использованием аппарата теории графов, являются варианты так называемой задачи коммивояжера.

Пусть имеется k городов, расстояния между которыми известны. Коммивояжер отправляется в путь из одного из них с тем, чтобы посетить остальные $k-1$ городов ровно по одному разу и вернуться в исходный город. Совершить свое путешествие он должен по наилучшему, в некотором смысле, маршруту.

Решение данной задачи может быть разделено на два этапа. На первом этапе нужно найти маршруты, которые в принципе решают задачу, т. е. позволяют посетить все города по одному разу и вернуться в исходную точку. На втором этапе из допустимых решений нужно выбрать наилучшее.

Решение первой подзадачи связано с построением **гамильтонового цикла**.

Определение. Гамильтоновым циклом называется простой цикл, проходящий через все вершины рассматриваемого графа. Если данный маршрут не замкнут, он называется гамильтоновой цепью.

Легко видеть, что сформулированная ранее задача коммивояжера есть задача отыскания на графе гамильтонова цикла. Внешне она похожа на задачу отыскания эйлерова цикла и кажется легко разрешимой. Однако это не так. В настоящее время не удалось сформулировать необходимых условий, которым должен удовлетворять граф, чтобы на нем можно было проложить гамильтонов цикл.

Заметим, что проблема существования гамильтонова пути принадлежит к классу так называемых NP-полных задач. Это широкий класс задач, включающий фундаментальные задачи из теории графов, логики, теории чисел, дискретной оптимизации и других дисциплин, ни для одной из которых неизвестен полиномиальный алгоритм (т. е. с числом шагов, ограниченным полиномом от размерности задачи), причем существование полиномиального алгоритма хотя бы для одной из них автоматически влекло бы за собой существование полиномиальных алгоритмов для всех этих задач. Именно факт фундаментальности многих NP-полных задач в различных областях и то, что, несмотря на независимые друг от друга усилия специалистов в этих областях, не удалось найти полиномиального алгоритма ни для одной из этих задач, склоняет к предположению, что такого алгоритма не существует.

Известно лишь **достаточное условие** существования гамильтоновой цепи на графе. Оно звучит следующим образом: если степень каж-

дой вершины графа, который имеет n вершин $(n \geq 3)$, не меньше $n/2$, то на этом графе можно построить гамильтонову цепь.

Практическую ценность такого условия можно показать на примере задачи о прохождении коня через все клетки шахматной доски, не заходя ни на одну дважды. Известно, что гамильтонова цепь для данной задачи существует, и не единственная. При этом сформулированное ранее достаточное условие требует для каждой вершины степень не менее 32, хотя реальные степени находятся в диапазоне от двух до восьми.

Что касается второй подзадачи, то решить ее в принципе просто, поскольку число всех маршрутов конечно. Для этого нужно организовать полный перебор всех маршрутов, вычислить их длины и выбрать самый короткий. Если число городов невелико, то полный перебор всех маршрутов, например начинающихся и заканчивающихся в городе А, можно хорошо организовать, воспользовавшись вспомогательным графом, называемым **граф – дерево**, а также введя понятие **взвешенного графа**.

Определим понятие взвешенного графа. Сопоставим каждой вершине $v_i \in V$ вес w_i из множества весов W. В результате получим множество взвешенных вершин $\{v_i, w_i\}$, при этом не обязательно, чтобы все веса были различными.

Аналогично сопоставим каждому ребру $e_j \in E$ вес p_j из множества весов P. В результате получим множество взвешенных ребер $\{e_j, p_j\}$. Определенные выше множества взвешенных вершин и ребер определяют в совокупности граф, взвешенный по вершинам и ребрам.

3.1.8. Граф-дерево и граф-лес

Определение. Н-граф называется неориентированным деревом (или просто деревом), если он связен и не содержит циклов, а значит, петель и кратных ребер. Дерево – это минимальный связный граф в том смысле, что при удалении хотя бы одного ребра он теряет связность. Наличие этих двух свойств (связность и отсутствие циклов) позволяет жестко связать число вершин и число ребер: в дереве с n вершинами всегда $n-1$ ребер. Пример графа–дерева приведен на рис. 3.13. В этом графе 8 вершин и 7 ребер. Ни одно ребро нельзя удалить из графа без того, чтобы он не потерял связность.

Вершина v графа G называется концевой, или висячей, если $\rho(v)=1$. В графе на рис. 3.13 концевыми являются вершины v_3, v_4, v_5, v_6, v_8 .

Неориентированный граф-дерево может быть превращен в ориентированный. Ориентация неориентированного дерева проводится следующим образом. В дереве G выбирается вершина v_0 – так называемый корень дерева G, и все ребра такого дерева с корнем ориентируются от этой вершины – корня. Для каждой выбранной вершины ориентация дерева единственна, все ребра ориентированы от корня. Если изменить направления всех ребер ориентированного дерева (к корню), получим ориентированный граф, который иногда называют **сетью сборки**.

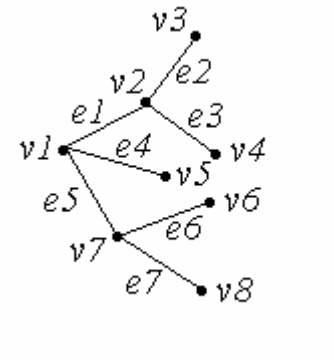


Рис. 3.13. Граф-дерево

В каждую вершину ориентированного дерева (за исключением корня) входит только одно ребро. Любое дерево можно ориентировать, выбрав в качестве корня любую его вершину.

Пусть v – вершина дерева G с корнем v_0; $B(v)$ – множество всех вершин, связанных с корнем цепями, проходящими через вершину v. Это множество порождает подграф $G(v)$, который называется **ветвью** вершины v в дереве с корнем v_0. Например, ветвью вершины v_2 на рис. 3.13 является подграф $G(v_2)$, содержащий вершины v_2, v_3, v_4 и ребра e_2, e_3.

Определение. Лес – несвязный н-граф без циклов. Связные компоненты леса являются деревьями. Любая часть леса также является лесом или деревом. В неориентированном дереве между любыми двумя вершинами существует цепь, и притом только одна.

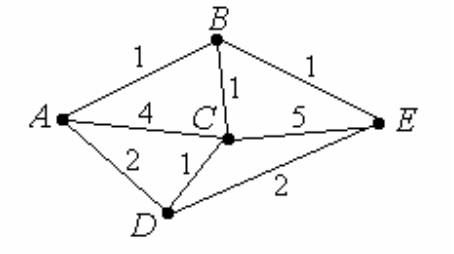


Рис. 3.14. Граф, взвешенный по ребрам

Рассмотрим пример использования графа-дерева для решения задачи поиска гамильтоновых путей на взвешенном по ребрам графе, приведенном на рис. 3.14. Веса можно рассматривать как некоторый эквивалент затрат, связанных с переходом по ребру из одной вершины в другую. Будем считать, что коммивояжер отправляется из вершины A с тем, чтобы посетить вершины B, C, D, E и вновь вернуться в A.

Для решения задачи удобно воспользоваться вспомогательным графом-деревом, который позволяет не только получить все гамильтоновы пути, но и отслеживать вес каждого пути. Методика построения

следующая: выделяется исходная вершина A и ей присваивается нулевой вес. На вспомогательном графе такая вершина помечается как $A0$. Из исходной вершины проводятся ребра во все смежные вершины, по которым маршрут еще не проходил. Новым вершинам присваиваются веса, равные затратам, которые необходимо понести для их достижения из исходной вершины. Для рассматриваемого примера результатом первого шага станут вершины $B1, C4, D2$. Далее точно таким же образом производятся шаги из вновь полученных вершин, пока эти пути не приведут в исходную вершину. Часть путей могут быть тупиковыми, т. к. не позволяют завершить маршрут, не заходя дважды в одну и ту же вершину. В данном примере, в частности, последовательность вершин A, B, C, D является тупиковой.

Приведенный на рис. 3.15 граф-дерево построен для решения задачи поиска кратчайшего гамильтонового пути с использованием метода полного перебора вариантов.

Проблема, однако, в том, что при большом числе вершин полный перебор вариантов – это работа, требующая огромных вычислений. В частности, для полного графа с k вершинами число маршрутов равно $(k-1)!$. Если 5!=120, то 10!=3 628 800. И все-таки, перебор маршрутов иногда бывает полезен. Известен метод решения задачи, в соответствии с которым шаг за шагом строится не полное, а «усеченное» дерево – часть ветвей в процессе его «выращивания» отсекаются, а оставшиеся ветви ведут к решению. Этот метод называется методом ветвей и границ.

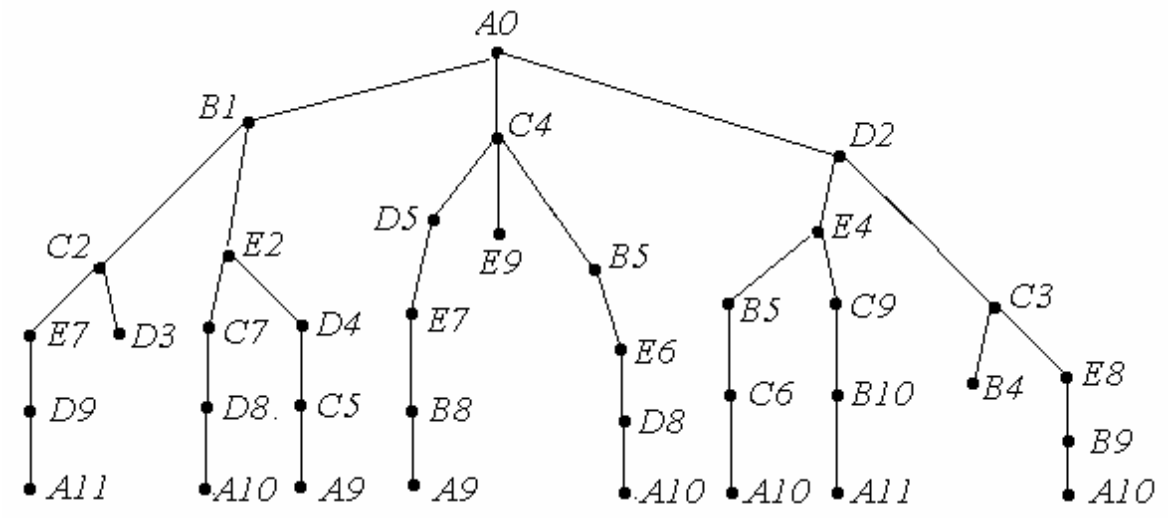


Рис. 3.15. Граф-дерево, соответствующий полному перебору вариантов построения гамильтонового цикла в исходном графе рис. 3.14

Идея метода достаточно очевидна – каким-либо образом выбрать на графе некоторый путь, желательно покороче, а затем отбрасывать все варианты, которые явно длиннее. Простейшим алгоритмом может быть продолжение движения из вершины, имеющей минимальный вес. В рассмотренном примере (после первого шага) движение должно быть продолжено из вершины с наименьшим весом $B1$, что приводит в вершины $C2$ и $E2$.

Выигрыш состоит в том, что неподходящие варианты просматриваются не до конца, а лишь до того момента, когда становится ясно, что рассматриваемый вариант хуже имеющегося. Общая полезность такого подхода зависит от степени дифференциации различных маршрутов и от удачности построения первого маршрута. Ясно также, что в самом плохом случае мы получим полный перебор вариантов.

3.1.9. Связность. Цикломатическое число графа

Во многих задачах интерес представляют характеристики связности графа. Ранее отмечалось, что граф может быть связным или несвязным. Но это бинарная характеристика. Ясно, что связный граф может быть более связен и менее связен.

Как обычно, для иллюстрации тех или иных свойств графа рассмотрим простую задачу. Назовем ее «задача о рисовых полях». Известно, что при выращивании риса небольшие участки земли – чеки – заливают водой, а перед сбором урожая эту воду спускают. Для заполнения чеков нужно в некоторых земляных плотинах, отгораживающих чеки друг от друга, открыть проходы. Вопрос: сколько стенок (земляных плотин) следует разрушить, чтобы заполнить все поле водой?

Решение этой задачи очень простое, если использовать теорию графов. На рисовое поле можно смотреть как на граф, имеющий циклы (каждый чек – цикл). Для заполнения поля водой достаточно в каждом чеке разрушить одну стенку (в каждом цикле удалить одно ребро). Оставшиеся ребра образуют граф-дерево. Число вершин n в графе останется без изменения, а число ребер будет равно $n-1$. Если ранее в графе было N ребер, то удалить пришлось $k = N - n + 1$ ребро. Величина $k = N - n + 1$ называется **цикломатическим числом** связного графа.

Цикломатическое число несвязного графа равно сумме цикломатических чисел связных компонент. Очевидно, что для любого графа–дерева это число равно нулю. Для леса – равно сумме цикломатических чисел связных компонент графа, т. е. тоже нулю. Для остальных графов цикломатические числа положительны.

Пример. Сколько и какие ребра следует удалить в графах на рис. 3.16, чтобы превратить их в деревья?

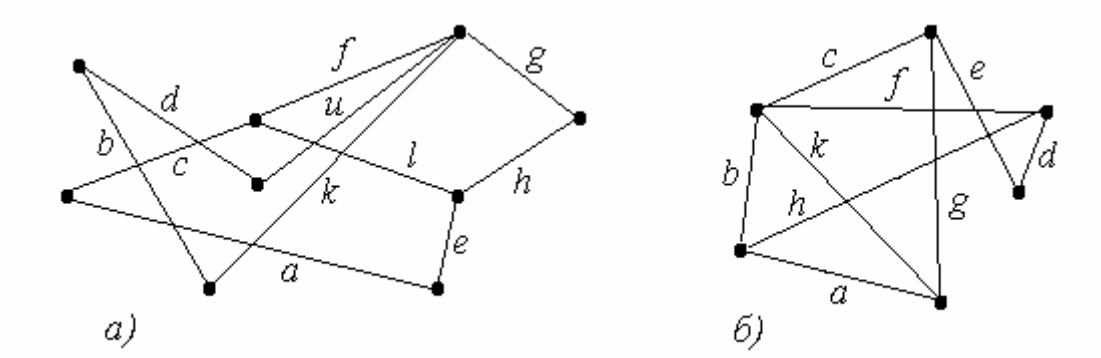


Рис. 3.16

Ответ. Граф *а* включает 9 вершин и 11 ребер, соответственно его цикломатическое число равно 11 – 9 + 1 = 3. Для графа *б* эти расчеты дают 9 – 6 + 1 = 4. Вариантов удаления ребер из графа может быть довольно много. На рис. 3.17 показаны графы, полученные удалением ребер t, g, k из графа a и ребер a, c, d, f из графа *б*.

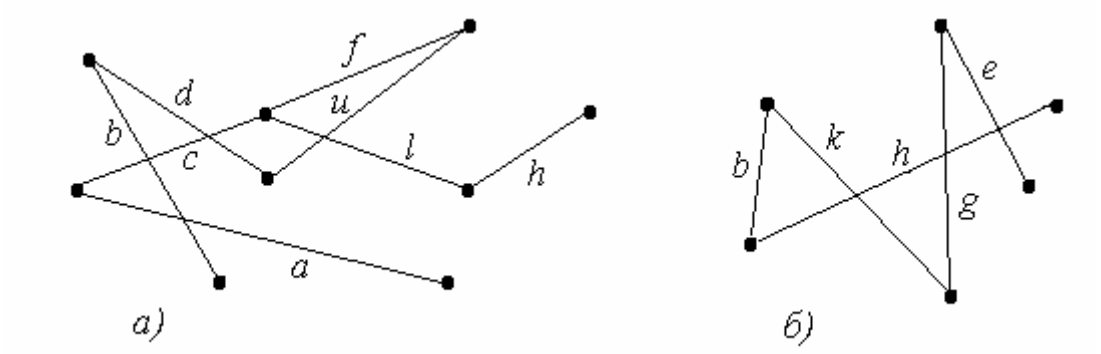


Рис. 3.17

3.1.10. Двудольные (четные) графы

Определение. Граф G называется двудольным (или четным), если множество его вершин V распадается на два непересекающихся подмножества V' и V'', таких, что каждое ребро графа G имеет один конец из V', а другой из V''. Двудольные графы рассматриваются во многих задачах дискретной математики, например в так называемых задачах о назначениях, когда одна группа вершин соответствует некоторым должностям, а другая – претендентам на эти должности. Из самой постановки очевидно, что связи в графе могут быть только между вершинами, относящимися к разным группам.

Особая роль двудольных графов применительно к задачам управления состоит в том, что они используются для представления объектов в системах событийного моделирования. Основная идея такого представления состоит в выделении двух типов элементов, которые в разных системах моделирования могут называться по-разному, но всегда один тип связан со статическим состоянием объекта, а другой – с переходом из одного статического состояния в другое. Дуги показывают направления переходов. Важно, что однородные элементы не могут быть соединены напрямую, например *событие – событие*, а только через элементы другой группы, через переходы.

Одна из таких систем – сети Петри, широко используемые для моделирования технических, экономических, юридических и других систем. Структуру сети Петри можно рассматривать как двудольный граф, в котором одно множество вершин состоит из позиций, а другое множество – из переходов. Ориентированные дуги соединяют позиции и переходы. Дуга, направленная от позиции p_i к переходу t_i определяет позицию, которая является входом перехода. Кратные входы в переход указываются кратными дугами на входных позициях в переход. Выходная позиция указывается дугой от перехода к позиции.

На рис. 3.18 представлена сеть Петри, изображенная так, как это принято в теории сетей Петри и включающая 5 позиций и 4 перехода.

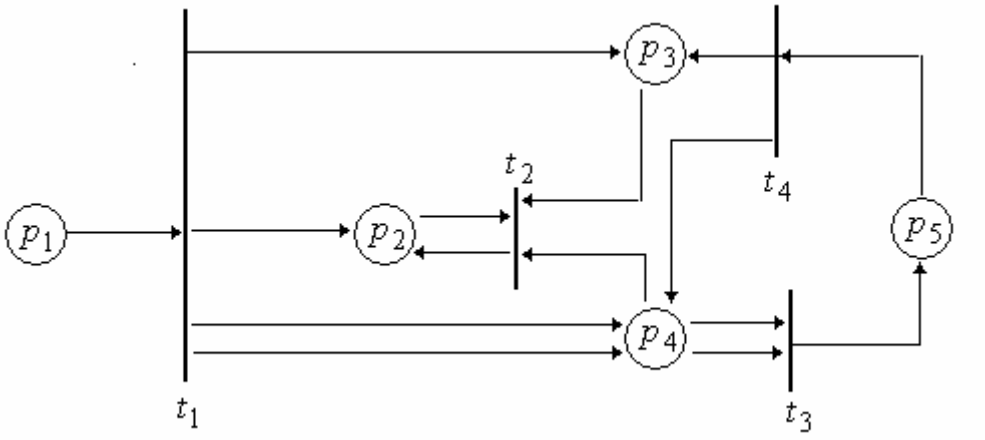


Рис. 3.18. Двудольный граф – сеть Петри

С точки зрения теории графов приведенная сеть Петри – это ориентированный двудольный мультиграф, т. к. допускает существование кратных дуг, в котором все множество вершин $V = \{p_1, p_2, p_3, p_4, p_5, t_1, t_2, t_3, t_4\}$ можно поделить на два подмножества: подмножество позиций $P = \{p_1, p_2, p_3, p_4, p_5\}$ и подмножество переходов $T = \{t_1, t_2, t_3, t_4\}$. Каждая дуга направлена от элемента одного под-

множества к элементу другого подмножества. На этих подмножествах определено отношение инцидентности $F \subseteq (P * T) \cup (T * P)$.

Для двудольных графов справедлива следующая теорема: граф G является двудольным, если все его циклы имеют четную длину.

Данная теорема является удобным конструктивным инструментом, позволяющим быстро проверить двудольность графа.

Пример

Проверить двудольность графов на рис. 3.19.

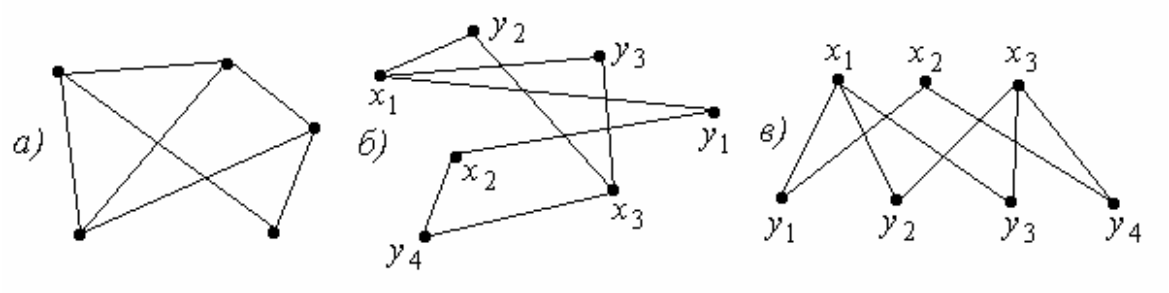


Рис. 3.19

Для решения задачи проще всего использовать вышеприведенную теорему о четной длине циклов двудольного графа. Граф 3.19, *а* не является двудольным, так как имеет циклы нечетной длины. Граф 3.19, *б* двудольный, т. к. все его циклы имеют четную длину. На рис. 3.19, *в* тот же граф представлен в традиционном для двудольных графов виде – с разделенными множествами вершин.

3.1.11. Планарность графов

Говорят, что граф G укладывается на плоскости, т. е. является планарным, или плоским, если его можно нарисовать так, что его ребра будут пересекаться лишь в концевых точках – вершинах. Изображение планарного графа на плоскости называется планарной укладкой. Определение планарности графов имеет большое практическое значение. Достаточно привести задачу разводки печатных плат, когда необходимо избежать пересечения проводников в местах, не предназначенных для соединений. Если изобразить места указанных соединений как вершины графа, то возникнет задача построения графа с непересекающимися ребрами. Важно отметить, что интерес представляет именно возможность построения графа с непересекающимися ребрами. Например, граф на рис. 3.20, *а* изображен так, что его ребра пересекаются. Однако этот граф планарен, т. к. его легко перерисовать нужным образом (рис. 3.20, *в*).

На рис. 3.20 приведены примеры планарных графов, а на рис. 3.21 – два основных непланарных графа – K_5 и $K_{3,3}$, которые обычно называют графами Куратовского.

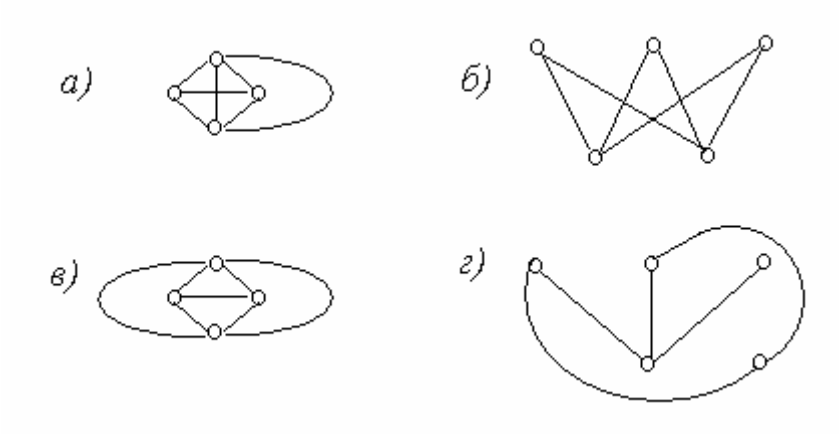


Рис. 3.20. Планарные графы

Графы Куратовского считаются основными непланарными графами потому, что играют решающую роль в исследовании планарности графов.

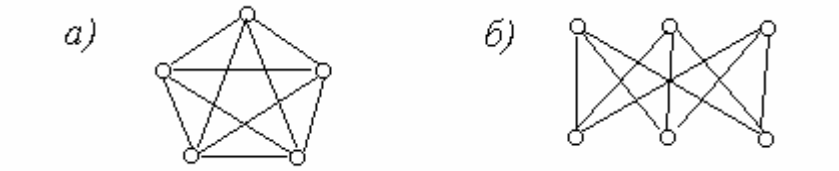


Рис. 3.21. Непланарные графы (графы Куратовского): *а* – граф K_5; *б* – граф $K_{3,3}$

Теорема. Граф планарен тогда и только тогда, когда он не содержит в качестве подграфа графа K_5 или $K_{3,3}$.

До появления этой теоремы определение планарности графа считалось одной из труднейших задач теории графов.

3.2. Сети

Одним из частных случаев графовых представлений является сеть. Сеть можно представить как систему, транспортирующую некий продукт из одной точки в другую. Примером может служить система нефтепроводов, где нефть течет из одних точек к другим. Используя такую концепцию, сеть можно представить как ориентированный граф, дуги которого передают продукт от одной вершины к другой. Обычно на этот граф наложены естественные ограничения. В частности:

• граф не содержит петель, т. к. это противоречит задаче транспортировки продукта;

• если есть дуга из вершины a в вершину b, то обратной дуги из b в a нет, т. е. поток продукта рассматривается только в одну сторону;

• граф должен быть связным.

Необходимыми элементами сети являются по крайней мере две вершины – α и β, называемые обычно **источником** и **стоком**. Локальная степень вершины α по входным дугам $\rho_1(\alpha)$ равна нулю, так что в источник ничто не втекает. Локальная степень вершины β по выходным дугам $\rho_2(\beta)$ равна нулю, так что из стока ничего не вытекает. Источники и стоки часто называют входными и выходными полюсами сети.

Определение. Сеть – это ориентированный граф S, имеющий выделенные вершины – входные и выходные полюсы сети, такие, что локальные степени входных полюсов по входящим дугам и локальные степени выходных полюсов по выходящим дугам равны нулю.

Вершины, отличные от полюсов, называются внутренними вершинами сети. Ребро, инцидентное хотя бы одному полюсу, называется полюсным ребром.

Определение. (k,l)-полюсником называется сеть с $k+l$ полюсами, разбитыми на два класса: k входных и l выходных полюсов.

Наибольший интерес представляют сети (1,1), называемые обычно двухполюсными сетями.

Будем называть цепью простую цепь между полюсами сети. Обозначим входной и выходной полюсы цепи S символами α_s и β_s. Полюсные ребра образуют входную $Z\alpha_s$ и выходную $Z\beta_s$ звезды, пересечение которых состоит из **сквозных ребер** (оно может быть пустым), инцидентных обоим полюсам.

Пример. На рис. 3.22 приведена двухполюсная сеть, в которой входным полюсом (источником) является вершина α_s, а выходным полюсом (стоком) – вершина β_s. Направленные ребра a,d,f,h образуют входную звезду $Z\alpha_s$, ребра h,g,c образуют выходную звезду $Z\beta_s$. Ребро h – сквозное.

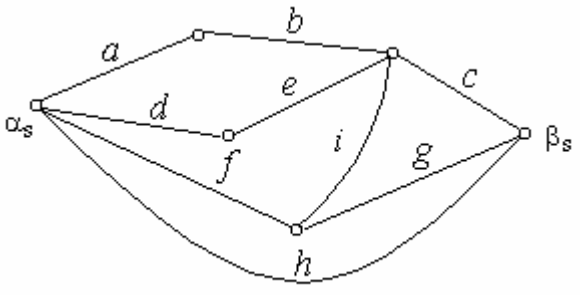


Рис. 3.22. Двухполюсная сеть

3.2.1. Потоки в сетях

Пусть S – произвольная ориентированная сеть, каждому ребру u которой приписано неотрицательное число $c(u)$ – пропускная способность ребра. Функция $f(u)$, заданная на множестве всех ребер и принимающая неотрицательные значениями, называется **потоком** в сети S, если:

- для любой дуги u величина $f(u)$, называемая потоком по дуге u, не превосходит пропускной способности дуги;
- в каждой внутренней вершине сети v выполняется закон Кирхгофа, согласно которому сумма значений потока по ребрам, входящим в вершину, равна сумме потока по ребрам, выходящим из вершины.

Поскольку сумма значений $f(v)$ по всем внутренним вершинам сети равна нулю, то $f(\alpha_s) = -f(\beta_s)$.

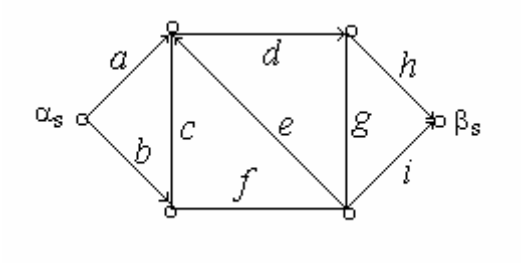


Рис. 3.23

Сечением сети называется множество ребер, при удалении которых сеть становится несвязной, причем полюсы попадают в разные компоненты связности. Ясно, что каждая цепь (а тем более каждый путь из α_s в β_s) проходит хотя бы через одно ребро сечения. В сети на рис. 3.23 примерами сечений являются $\{d,e,f\}$, $\{b,c,e,g,h\}$, $\{d,g,h,i\}$. Сечение будем называть **простым**, если при удалении любого его ребра оно перестает быть сечением. Так, $\{d,e,f\}$, $\{b,c,e,g,h\}$ являются простыми сечениями, в то время как $\{d,g,h,i\}$ таковым не является.

Если в связной цепи удаляется простое сечение, то сеть распадается ровно на две части: левую часть, содержащую α_s, и правую часть, содержащую β_s. Каждое ребро простого сечения связывает вершины из разных частей. Будем называть ребро прямым, если в сети оно ориентировано слева направо, и обратным – в противном случае. Будет ли ориентированное ребро прямым или обратным, вообще говоря, зависит от выбора сечения. Так, в нашем примере (рис. 3.23) ребро e в сечениях $\{d,e,f\}$ и $\{b,c,e,g,h\}$ – обратное, а в сечении $\{a,c,e,g,i\}$ – прямое.

Каждому простому сечению w припишем пропускную способность $c(w)$, равную сумме пропускных способностей всех его прямых ребер. В примере на рис. 3.23 сечение $\{d,e,f\}$ имеет пропускную способность 5 + 1 = 6, а сечение $\{b,c,e,g,h\}$ – 3 + 2 + 3 + 2 = 10. Если сеть несвязна и

ее полюсы находятся в разных компонентах связности, то естественно считать единственным простым сечением сети пустое множество, а его пропускную способность нулевой.

3.2.2. Расчет максимального потока в сети

Одной из наиболее важных для теории сетей является задача о максимальном потоке, который может пропустить сеть.

Величиной потока f в сети S называется величина, равная сумме потоков по всем дугам, заходящим в β_s, или, что то же самое, величина, равная сумме потоков по всем дугам, исходящим из α_s.

Пусть f – поток в сети S. Дуга $u \in U$, где U – множество дуг данной сети, называется насыщенной, если поток по ней равен ее пропускной способности, т. е. если $f(u) = c(u)$.

Поток f называется **полным**, если любой путь в S из α_s в β_s содержит по крайней мере одну насыщенную дугу.

Поток называется **максимальным**, если его величина $f_{\max}$ принимает максимальное значение по сравнению с другими допустимыми потоками в сети *S*.

Очевидно, максимальный поток $f_{\max}$ обязательно является полным, т. к. в противном случае в S существует некоторая простая цепь из α_s в β_s, не содержащая насыщенных дуг, а следовательно, можно увеличить потоки по всем дугам и тем самым увеличить $f_{\max}$, что противоречит условию максимальности потока. Обратное же, вообще говоря, неверно. Существуют полные потоки, не являющиеся максимальными.

Решение задачи о максимальном потоке базируется на теореме Форда и Фалкерсона.

Теорема (Форд и Фалкерсон). Для любой сети с одним источником и одним стоком максимальная величина потока $f_{\max}$ через сеть S равна минимальной из пропускных способностей $c_{\min}$ ее простых сечений.

Существует алгоритм нахождения максимального потока в сети, разработанный теми же авторами, который состоит из двух этапов. На первом этапе находится полный поток в сети, на втором – реализуется переход от полного потока к максимальному.

Этап 1. Расчет полного потока..

Сначала рассмотрим алгоритм построения полного потока в сети S. Предположим, что в сети имеется некоторый поток и путь из α_s в β_s, состоящий из ненасыщенных дуг. Тогда очевидно, что поток в сети можно увеличить на величину Δ, равную минимальной из остаточных пропускных способностей дуг, входящих в этот путь. Перебирая все возможные пути из α_s в β_s и проводя такую процедуру увеличения потока, пока это возможно, получим в результате полный поток, т. е. такой поток, для которого каждый путь из α_s в β_s содержит по крайней мере одну насыщенную дугу. Более конструктивно данный алгоритм можно представить в виде следующей последовательности шагов.

Алгоритм

Шаг 1. Полагаем $\forall u \in U$, $f(u) = 0$ (т. е. начинаем с нулевого потока). Кроме того, полагаем $S' = S$, где S' – некоторая рабочая сеть.

Шаг 2. Удаляем из орграфа S' все дуги, являющиеся насыщенными при данном потоке f в сети S.

Шаг 3. Ищем в S' простую цепь η из α_s в β_s. Если такой цепи нет, то f – искомый полный поток в сети S. В противном случае переходим к шагу 4.

Шаг 4. Увеличиваем поток $f(u)$ по каждой дуге u из η на одинаковую величину $a > 0$ такую, что по крайней мере одна дуга из η оказывается насыщенной, а потоки по остальным дугам из η не превышают их пропускных способностей. При этом величина потока f также увеличивается на a, а сам поток остается допустимым. После этого переходим к шагу 2.

Пример. Требуется получить полный поток в сети S, приведенной на рис. 3.24, *а*. Начальное состояние этой сети S' со всеми нулевыми потоками представлено на рис. 3.24, *б*.

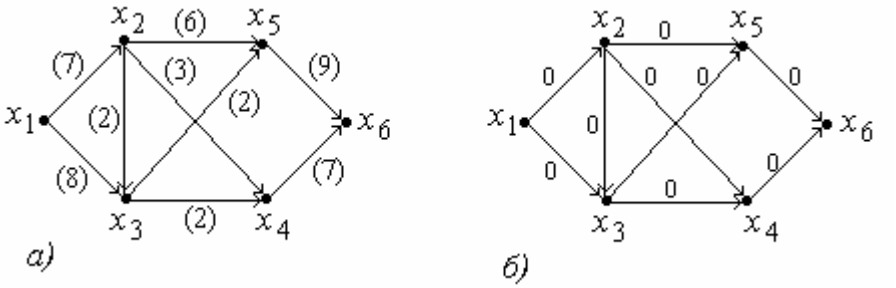


Рис. 3.24

Первая итерация. Выделим в S' простую цепь $\eta_1 = (x_1 \;\; x_2 \;\; x_4 \;\; x_6)$ и увеличим потоки по дугам из η_1 на 3 единицы (до насыщения дуги (x_2, x_4)). В результате получим поток $f = f_1$, со-

держащий одну насыщенную дугу (рис. 3.25, *а*). Пометим ее знаком «+» и удалим из орграфа S'. Оставшийся орграф снова обозначаем S'.

Вторая итерация. Выделим в S' простую цепь $\eta_2 = (x_1 \;\; x_2 \;\; x_3 \;\; x_4 \;\; x_6)$ и увеличим потоки по дугам из η_2 на две единицы до насыщения дуг (x_2, x_3) и (x_3, x_4). В результате получим поток $f = f_2$, содержащий три насыщенные дуги (рис. 3.25, *б*). Уберем насыщенные дуги из графа. Оставшийся орграф снова обозначаем S'. Он представлен на рис. 3.26, *а*.

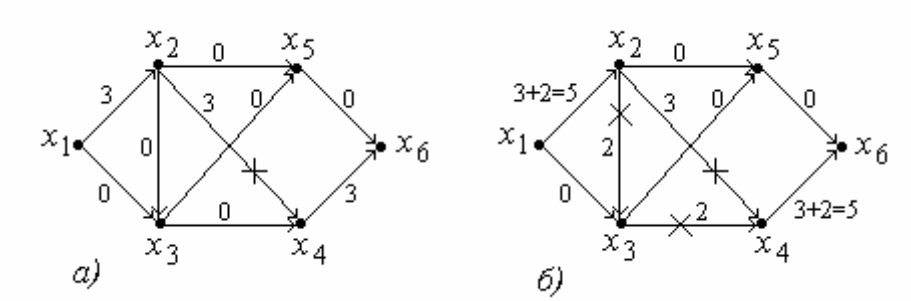


Рис. 3.25

Третья итерация. Выделим в S' простую цепь $\eta_3 = (x_1 \;\; x_3 \;\; x_5 \;\; x_6)$ и увеличим потоки по дугам из η_3 на две единицы до насыщения дуги (x_3, x_5). В результате получим поток $f = f_3$, содержащий четыре насыщенные дуги (рис. 3.26, *b*).

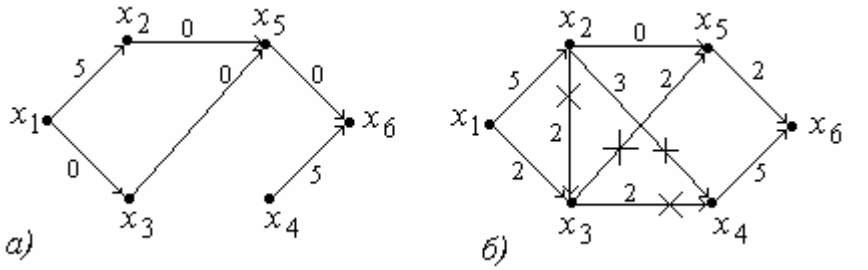


Рис. 3.26

Удалим вновь полученную насыщенную дугу из S'. Оставшийся орграф снова обозначаем S' (рис. 27, *а*).

Четвертая итерация. Выделим в S' простую цепь $\eta_4 = (x_1 \;\; x_2 \;\; x_5 \;\; x_6)$ и увеличим потоки по дугам из η_4 на две единицы до насыщения дуги $(x_1 \;\; x_2)$. В результате получим поток $f = f_4$, содержащий пять насыщенных дуг (рис. 3.27, *б*).

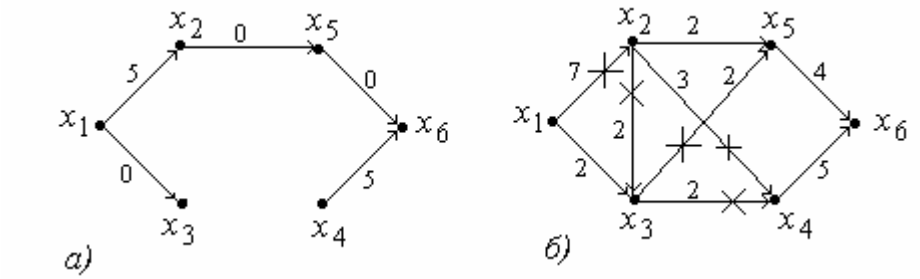


Рис. 3.27

В оставшемся орграфе нет прямой цепи от x_1 к x_6, соответственно, поток f_4 является полным. Величина полученного полного потока равна 9.

Правильность полученного результата можно проверить, выполнив несколько различных сечений итоговой сети. Уточнить обозначение сечений. На рис. 3.28, *б* представлена полученная сеть на которую нанесены три сечения. В сечении *а–а* поток равен $f_{a-a} = 7 - 2 + 2 + 2 = 9$. Здесь поток по ребру (x_2, x_3) имеет отрицательный знак. В сечении *б–б* поток $f_{б\text{-}б} = 2 + 2 + 5 = 9$. Наконец, поток в сечении *с–с* также равен $f_{c-c} = 2 + 3 + 4 = 9$, что говорит о том, что задача решена верно.

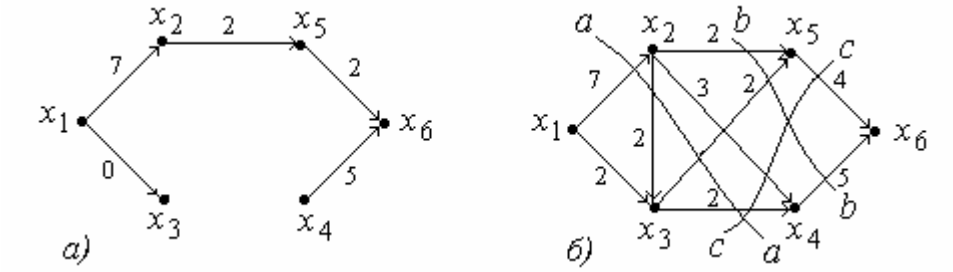


Рис. 3.28

Этап 2. Расчет максимального потока.

Рассмотрим в сети, для которой уже построен полный поток, произвольный маршрут из α_s в β_s. Дуги, образующие этот маршрут, естественным образом делятся на два типа: прямые (ориентированные от α_s в β_s) и обратные (ориентированные от β_s к α_s). Пусть существует путь, в котором прямые дуги не насыщены, а потоки на обратных дугах положительны. Пусть Δ_1 – минимальная из остаточных пропускных способностей прямых дуг, а Δ_2 – минимальная из величин потоков обратных дуг. Тогда поток в сети можно увеличить на величину $\Delta = \min\{\Delta_1, \Delta_2\}$, прибавляя Δ к потокам на прямых дугах и вычитая Δ из потоков на обратных дугах.

На рис. 3.29 приведена сеть, для которой был рассчитан полный поток. В скобках указаны пропускные способности ребер.

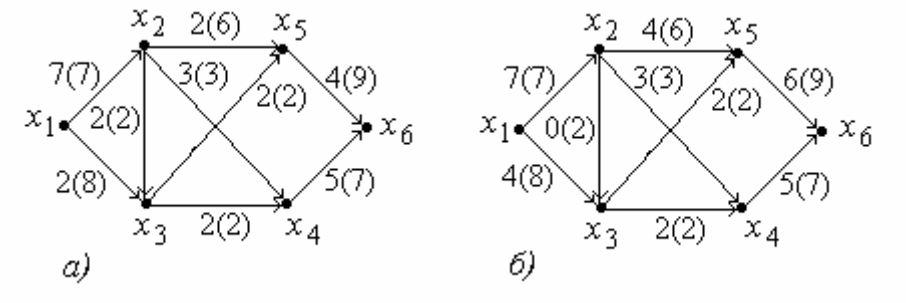


Рис. 3.29

В этой сети (орграфе) можно найти цепь $\mu = (x_1 \;\; x_3 \;\; x_2 \;\; x_5 \;\; x_6)$, отвечающую указанным требованиям: прямые дуги не насыщены, а обратная дуга (x_3, x_2) насыщена.

Поток по этой цепи можно увеличить на две единицы. При этом поток по дуге (x_3, x_2) станет нулевым. Общий поток в сети увеличится до 11 единиц, как это показано на рис. 3.29. При этом, как и ранее, выполняется равенство потоков по всем сечениям. Других цепей, отвечающих условию ненасыщенности прямых ребер, в орграфе нет, – следовательно, задача вычисления максимального потока решена.

Глава 4
АВТОМАТЫ, ЯЗЫКИ, ЭЛЕМЕНТЫ КОДИРОВАНИЯ

4.1. Элементы теории автоматов

Рассмотренные ранее, например в разделе «Синтез логических схем», преобразователи являются функциональными (логическими схемами), т. е. не имеют памяти. Выход преобразователя полностью определяется сигналами на его входах. Часто, однако, результат преобразования $вход \to выход$ зависит не только от того, какая информация в данный момент появилась на входах, но и от того, что происходило раньше, от предыстории преобразования. Например, реакция человека на чье-либо замечание может быть существенно различной, в зависимости от его настроения, которое определяется событиями, имевшими место ранее.

Такие нефункциональные преобразователи дискретной информации, реакция которых зависит не только от состояния входа на данный момент, но и от того, что было на входе раньше, от входной истории, называются **автоматами** и изучаются в разделе теории управляющих систем, который называется «Теория автоматов». При этом функциональные преобразователи иногда рассматриваются как частный случай автоматов – автоматы без памяти.

С определенной точки зрения автоматами являются как реальные устройства (вычислительные машины, автоматы, живые организмы и т. д.), так и абстрактные системы (например, формальная система, аксиоматические теории и т. д.). Понятие автомата может служить модельным объектом в самых разнообразных задачах, благодаря чему возможно применение теории автоматов в различных научных и прикладных исследованиях. Например, при разработке и реализации сложного поведения в управляемых событиями программах, таких как сетевые адаптеры и компиляторы.

Большинство задач теории автоматов – общие для основных видов управляющих систем. К ним относятся задачи анализа и синтеза автоматов, задачи полноты, минимизации, эквивалентных преобразований автоматов и др.

4.1.1. Общее определение конечного автомата

Автомат представляет собой кибернетическую систему, перерабатывающую дискретную информацию и меняющую свое внутреннее состояние лишь в допустимые моменты времени. В каждый момент времени автомат находится в некотором **состоянии** и может изменить это состояние под действием входного сигнала. Определить понятие состояния для общего случая весьма трудно, слишком оно фундаментально. Неформально **состояние системы – это ее характеристика, однозначно определяющая ее дальнейшее поведение, все последующие реакции системы на внешние воздействия.** На один и тот же сигнал автомат может реагировать по-разному, в зависимости от того, в каком состоянии находится в данный момент. Таким образом, в своих состояниях автомат запоминает свою историю, свое «концентрированное» прошлое.

Число возможных историй бесконечно велико, даже если вариантов входных воздействий не много. Однако на множестве предысторий можно ввести отношение эквивалентности (см. разд. 1.2.3). При этом две предыстории попадут в один класс эквивалентности, если они приводят автомат в одно и то же состояние. Очевидно, автомату не нужно запоминать конкретные входные истории. Достаточно, чтобы он запоминал классы эквивалентности, к которым данные истории принадлежат. Наиболее интересен случай, когда число классов эквивалентности и соответственно состояний конечно. Такой преобразователь называется **конечным автоматом**. Функциональный преобразователь (логическая схема, автомат без памяти) может рассматриваться как конечный автомат с одним состоянием.

Общее теоретико-множественное определение конечного автомата может быть сформулировано следующим образом.

Определение. Абстрактным конечным автоматом называется шестерка объектов

$$A = \{S, s_0, X, Y, \varphi, \psi\},$$

где S – конечное непустое множество состояний;

$s_0 \in S$ – начальное состояние;

X – конечное непустое множество входных сигналов;

Y – множество выходных сигналов;

$\varphi : S \times X \to S$ – функция переходов;

$\psi : S \times X \to S$ – функция выходов.

По характеру отсчета дискретного времени конечные автоматы делятся на **синхронные и асинхронные**. В синхронных конечных автома-

тах моменты времени (такты), в которые автомат считывает входные сигналы, определяются принудительно синхронизирующими сигналами. После очередного синхронизирующего сигнала, с учётом считанной информации и в соответствии с функциями φ, ψ, происходит переход автомата в новое состояние и выдача сигнала на выходе, после чего автомат может воспринимать следующее значение входного сигнала.

Асинхронный конечный автомат считывает входной сигнал непрерывно, и поэтому, реагируя на достаточно длинный входной сигнал постоянной величины, он может, как следует из соотношений для функционирования автомата, несколько раз изменять состояние, выдавая соответствующее число выходных сигналов, пока не перейдёт в устойчивое состояние, которое уже не может быть изменено данным входным сигналом.

4.1.2. Автоматы Мили и Мура

По способу формирования функций выходов среди синхронных автоматов выделяют автоматы **Мили** и автоматы **Мура**. В автомате Мили функция выходов ψ определяет значение выходного символа по классической схеме абстрактного автомата. Она является двухаргументной и символ $y(t)$ в выходном канале обнаруживается только при наличии символа во входном канале $x(t)$. Функции перехода и выхода для автомата Мили можно записать в виде

$$s(t+h) \quad \varphi(s(t), x(t));$$

$$y(t+h) \quad \psi(s(t), x(t)),$$

где h – длительность такта.

Зависимость выходного сигнала только от состояния представлена в автоматах типа Мура. В автоматах этого типа функция выходов определяет значение выходного символа только по одному аргументу – состоянию автомата. При этом символ $y(t)$ в выходном канале существует все время, пока автомат находится в состоянии $s(t)$. Для автомата Мура функции перехода и выхода можно записать как

$$s(t+h) \quad \varphi(s(t), x(t));$$

$$y(t+h) \quad \psi(s(t)).$$

Считается, что реализация автоматов Мили, как правило, более проста, но в них возникают проблемы с синхронностью формирования выходных сигналов. Между моделями Мили и Мура существует соот-

ветствие, позволяющее преобразовать закон функционирования одного из них в другой или обратно.

4.1.3. Способы задания конечных автоматов

Существует несколько способов представления конечных автоматов, каждый из которых имеет свои достоинства и недостатки. Функционирование несложного автомата удобно представлять графически, с помощью диаграммы состояний, являющейся ориентированным графом, у которого состояния представлены вершинами, а дуги соответствуют переходам из одного состояния в другое. Для сложных автоматов графическое представление становится слишком громоздким. Более целесообразным в этом случае представляются различные табличные способы задания в форме таблиц переходов и выходов либо в форме матриц смежности направленного графа.

В [1] приведен пример автомата, описывающего поведение отца, отправившего сына в школу. Сын приносит двойки и пятерки. Но отец не наказывает сына за каждую двойку. Он понимает, что она может быть и случайной, поэтому выбрана более гибкая тактика, которая описывается автоматом, граф которого представлен на рис. 4.1.

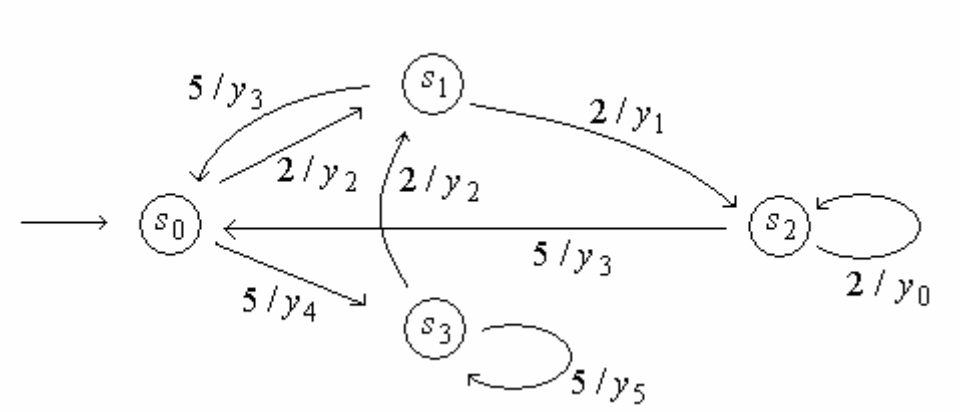


Рис. 4.1. Автомат, описывающий поведение отца, отправившего сына в школу

Автомат имеет четыре состояния $\{s_0, s_1, s_2, s_3\}$ и два входных сигнала x_i – оценки, получаемые сыном в школе: $\{2,5\}$. Начиная с начального состояния s_0 (оно помечено входной стрелкой), автомат под действием входных сигналов переходит из одного состояния в другое и формирует выходные сигналы – реакции на входы. Выходы автомата $\{y_0, y_1, \ldots, y_5\}$ будем интерпретировать как действия отца следующим

образом: y_0 – брать ремень; y_1 – ругать сына; y_2 – успокаивать сына; y_3 – надеяться; y_4 – радоваться; y_5 – ликовать.

Сына, получившего двойку, ожидает дома совершенно разная реакция отца, в зависимости от предыстории его учебы. Отец помнит предысторию учебы сына и строит свое поведение с учетом этой предыстории. Например, после третьей двойки во входной истории 2,2,2 сына встретят ремнем, а в истории 2,2,5,2 – будут успокаивать. Каждая предыстория определяет текущее состояние автомата (отца), при этом некоторые предыстории эквивалентны (те, что приводят автомат в одно и то же состояние). История 2,2,5 эквивалентна пустой истории. Текущее состояние автомата представляет все то, что автомат знает о прошлом с точки зрения его будущего поведения.

В приведенном примере автомат может рассматриваться как синхронный, если сын приносит по одной оценке, либо как асинхронный, если сын получает сразу несколько оценок.

Кроме того, рассмотренный автомат является, очевидно, автоматом Мили, т. к. реакция зависит как от состояния отца, так и от полученной оценки.

Эквивалентом структурного представления автомата может являться, например, матрица смежности орграфа. В отличие от обычной матрицы смежности (см. разд. 3.1.2) в качестве элемента δ_{kl} записываются входной и выходной сигналы, соответствующие переходу автомата из k-го состояния в l-е. Если переход $s_k \to s_l$ происходит под воздействием нескольких сигналов, элемент δ_{kl} представляет собой множество пар «вход/выход» для этого перехода, соединенных знаком дизъюнкции.

Для рассмотренного примера матрица смежности задана табл. 4.1.

Таблица 4.1

	s_0	s_1	s_2	s_3
s_0		$2 / y_2$		$5 / y_4$
s_1	$5 / y_3$		$2 / y_1$	
s_2	$5 / y_5$		$2 / y_0$	
s_3	$5 / y_5$	$2 / y_2$		

Другим вариантом табличного представления является построение таблиц переходов и выходов, которые несколько отличаются для автоматов Мили и Мура.

Для автомата Мили таблица переходов в общем виде приведена ниже.

Таблица 4.2

	s_0	s_1	…	s_k
x_1	$\varphi(s_0,x_1)$	$\varphi(s_1,x_1)$	…	$\varphi(s_k,x_1)$
x_2	$\varphi(s_0,x_2)$	$\varphi(s_1,x_2)$	…	$\varphi(s_k,x_2)$
…	…	…	…	…
x_k	$\varphi(s_0,x_k)$	$\varphi(s_1,x_k)$	…	$\varphi(s_k,x_k)$

Таблица выходов получается из таблицы переходов заменой функций $\varphi(s_i,x_k)$ на $\psi(s_i,x_k)$.

Табличное описание автомата Мура можно записать как табл. 4.3.

Таблица 4.3

	s_0	s_1	…	s_k
	$\psi(s_0)$	$\psi(s_1)$	…	$\psi(s_k)$
x_1	$\varphi(s_0,x_1)$	$\varphi(s_1,x_1)$	…	$\varphi(s_k,x_1)$
x_2	$\varphi(s_0,x_2)$	$\varphi(s_1,x_2)$	…	$\varphi(s_k,x_2)$
…	…	…	…	…
x_k	$\varphi(s_0,x_k)$	$\varphi(s_1,x_k)$	…	$\varphi(s_k,x_k)$

Для рассмотренного примера, который представляет собой автомат Мили, таблицы переходов и выходов имеют вид табл. 4.4 и табл. 4.5.

Таблица 4.4

$\varphi(s,x)$	s_0	s_1	s_2	s_3
x_1 (5)	s_3	s_0	s_0	s_3
x_2 (2)	s_1	s_2	s_2	s_1

Таблица 4.5

$\psi(s,x)$	s_0	s_1	s_2	s_3

x_1 (5)	y_4	y_3	y_3	y_5
x_2 (2)	y_2	y_1	y_0	y_2

4.1.4. Реализация конечных автоматов

Рассмотрим два вида реализации конечного автомата: программную и аппаратную.

Программную реализацию конечного автомата можно выполнить на любом языке высокого уровня, причем топология блок-схемы программы будет повторять топологию графа переходов автомата. Построив программу, соответствующую графу на рис. 4.1, и добавив исполнительные устройства, реализующие отдельные выходные операции (бить ремнем, ругать, успокаивать и т. д.), можно воспитание сына поручить компьютеру.

Аппаратная реализация требует применения устройств памяти для запоминания текущего состояния автомата. Обычно на практике, используют двоичные элементы памяти. Функциональный блок автомата реализуется как конечный функциональный преобразователь. Таким образом, общий подход к аппаратной реализации конечного автомата совпадает с общей процедурой синтеза логических схем, описанной в разд. 2.3, и включает следующие шаги:

- входные и выходные сигналы и внутренние состояния автомата кодируются двоичными кодами;
- по таблицам переходов и выходов составляются кодированные таблицы переходов и выходов – фактически табличное задание отображений φ и ψ;
- по кодированным таблицам переходов и выходов формируются аналитические выражения логических функций и проводится их минимизация;
- полученные минимальные формы реализуются в заданном элементном базисе;
- решаются схемотехнические вопросы синхронизации – привязки моментов выдачи выходного сигнала и изменения состояния внутренней памяти к моментам поступления входных сигналов на вход автомата.

Рассмотрим реализацию автомата из рассмотренного примера. Входных сигналов два; мы их закодируем так: $2 \to 0$, $5 \to 1$. Выходных сигналов шесть. Закодируем их: $y_0 \to 000$, $y_1 \to 001$, …, $y_5 \to 101$. Внутренних состояний четыре. Закодируем их: $s_0 \to 00$, $s_1 \to 01$,

$s_2 \to 10$, $s_3 \to 11$. Таким образом, имеем: $X = \{0,1\}$, $Y = \{000,001,010,011,100\}$, $S = \{00,01,10,11\}$. Структурная схема этого автомата после двоичного кодирования имеет вид, показанный на рис. 4.2, где F – функциональный преобразователь без памяти, реализующий отображения φ и ψ, БП – блок памяти.

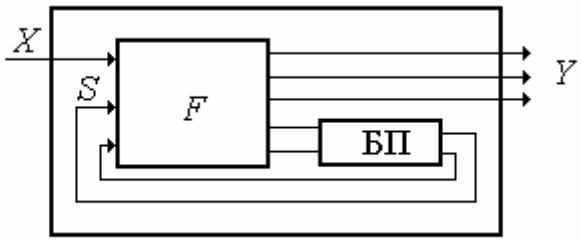


Рис. 4.2. Структурная схема автомата

В кодированной таблице переходов и выходов автомата (табл. 4.6) один двоичный разряд x кодирует входной сигнал x_i, пары двоичных разрядов q_1, q_2, p_1, p_2 кодируют соответственно текущее s_i и следующее s_{i+1} состояния автомата, разряды z_1, z_2, z_3 кодируют выходной сигнал y_i.

Таблица 4.6

x_i	s_i		s_{i+1}		y_i		
x	q_1	q_2	p_1	p_2	z_1	z_2	z_3
0	0	0	0	1	0	1	0
0	0	1	1	0	0	0	1
0	1	0	1	0	0	0	0
0	1	1	0	1	0	1	0
1	0	0	1	1	1	0	0
1	0	1	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	1	1	1	1	0	1

После записи логической формулы и минимизации в классе ДНФ, как это рассмотрено в разд. 2.4, получим аналитические выражения для всех двоичных функций, реализация которых показана на рис. 4.3.

Блоки Т1 и Т2 – триггеры, которые запоминают двоичный сигнал до прихода следующего. Вход t в триггере – синхронизационный вход, разрешающий переключение триггера. Сигнал на этом входе должен появляться в момент получения автоматом очередного входного сигнала. Этот же синхросигнал обеспечивает появление на выходе импульсного значения выходного сигнала.

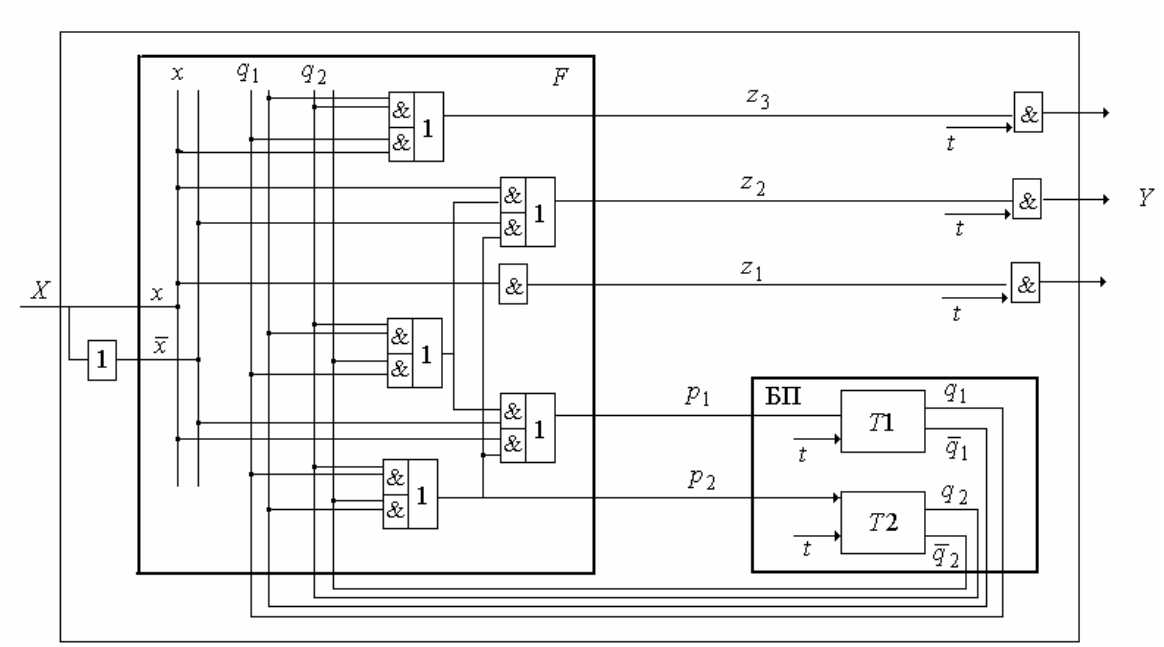


Рис. 4.3. Функциональная схема автомата

Электронные часы

В качестве технического устройства, построенного на конечноавтоматной модели, рассмотрим электронные часы [3]. Электронные часы обычно показывают время, дату, дают возможность установки времени и даты, а также выполняют множество других функций. Управление всеми этими возможностями производится встроенным преобразователем, входами которого являются события нажатия внешних управляющих кнопок.

Рассмотрим простейший вариант, когда показываются и корректируются только дата (год, месяц, день) или время (минуты и секунды). Числа, соответствующие этим данным, хранятся в регистрах памяти и могут быть переданы на регистры отображения путем управления комбинационными схемами. Управление осуществляется двумя кнопками – «*a*» и «*b*».

Граф переходов устройства управления, организующего работу всех элементов часов, изображен на рис. 4.4.

В начальном состоянии отображается время. Конечный автомат реагирует на нажатие кнопки «*a*» переходом в состояние «Установка минут», в котором событие нажатия кнопки «*b*» вызовет увеличение на единицу числа, хранящегося в регистрах, отведенных для минут. Событие нажатия кнопки «*b*» в состоянии «Установка месяца» вызовет увеличение числа, хранящегося в регистрах, отведенных для месяца и т. д.

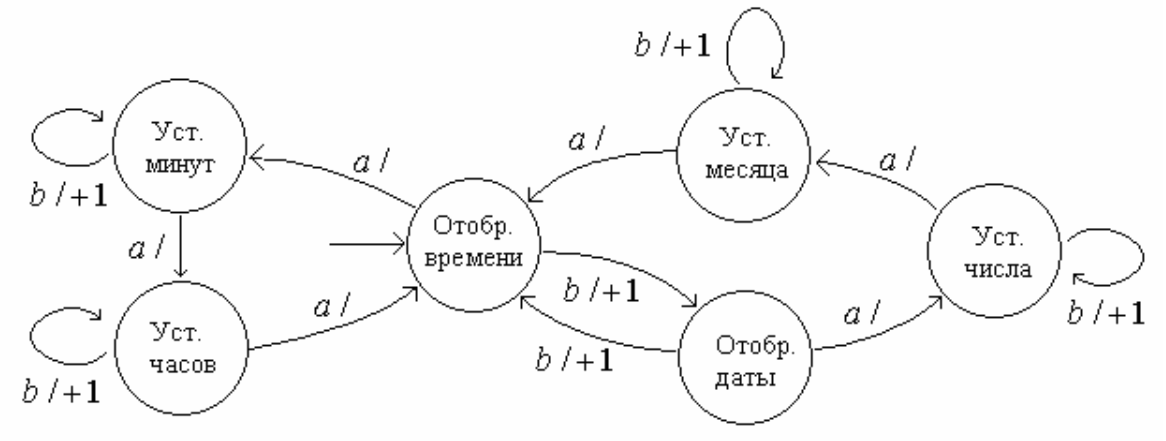


Рис. 4.4. Автомат устройства управления электронными часами

4.1.5. Автоматы-распознаватели

Еще одна практически важная точка зрения на автоматы состоит в том, что они рассматриваются не как преобразователи информации, реагирующие на отдельные входные сигналы, а как распознаватели последовательностей входных сигналов. Многолетний опыт использования алгоритмов в математике показал, что, какова бы ни была проблема и алгоритм ее решения, всегда можно описать эту проблему и алгоритм в виде процедуры переработки слов в некотором специально выбранном алфавите.

Введем некоторые базовые понятия.

Пусть задано конечное непустое множество символов $A = \{a_1, a_2, \ldots, a_n\}$. Назовем его **алфавитом**. Природа букв может быть различной (цифры, буквы, иероглифы и т. д.), главное, чтобы из символов можно было образовывать слова (цепочки). Например, A может быть подмножеством русского алфавита или просто $A = \{0,1\}$.

Словом, в алфавите называется любая конечная упорядоченная последовательность букв этого алфавита. Число букв, входящих в слово α, называется длиной слова α и обозначается $|\alpha|$. Будем обозначать A^n и A^* множество всех слов длины n и множество всех слов произвольной длины соответственно, построенных из букв алфавита A.

Слово может быть пустым. Пустое слово обозначается Λ: $\Lambda \in A^*, |\Lambda| = 0, \Lambda \notin A$.

Слово γ называется подсловом α, если $\alpha = \beta\gamma\delta$, где β, δ – любые слова в данном алфавите, в том числе пустые.

Над словами можно выполнять некоторые операции, важнейшей из которых является конкатенация.

Конкатенацией слов α и β называется слово $w = \alpha \circ \beta$, или просто $w = \alpha\beta$. Например, если $A = \{0,1\}$, то 11011∘100010 = 11011100010. Операция конкатенации ассоциативна, но не коммутативна, т. е. $\alpha(\beta\gamma) = (\alpha\beta)\gamma$, но $\alpha\beta \neq \beta\alpha$. Слово вида $\alpha\alpha\ldots\alpha$, в которое α входит n раз, обычно обозначают α^n. Обозначение $\alpha*$ соответствует слову, включающему произвольное число букв α.

Языком в алфавите A называется произвольное множество L слов в этом алфавите, т. е. $L \subseteq A^*$. Если A – множество букв русского алфавита, то L может быть множеством слов русского языка, если A – множество символов языка программирования, то L – множество слов этого языка.

Пусть S – подмножество множества A^*. Тогда S^* – множество всех слов, образованных конкатенацией слов из множества S, т. е. $S^* = \{w_1 w_2 \ldots w_n : w_i \in S\}$.

К языкам применимы обычные операции над множествами: объединение, пересечение, разность, взятие дополнения.

Автомат-распознаватель представляет собой устройство, распознающее или допускающее определенные элементы множества $A*$, где A – конечный алфавит. Различные автоматы распознают или допускают различные элементы множества $A*$. Подмножество элементов множества $A*$, допускаемое автоматом M, – это язык, который называется языком L над алфавитом A, допускаемым автоматом M, и обозначается $M(L)$.

Роль распознавателя может выполнить автомат без выхода. Свяжем с некоторыми состояниями автомата символ «Да» и объединим их в множество $T \subseteq S$. С остальными состояниями свяжем символ «Нет». Тогда множество входных цепочек автомата разобьется на два класса: одни – приводящие автомат в одно из состояний, помеченных «Да», все другие – приводящие автомат в одно из состояний, помеченных «Нет».

Определение. Конечным автоматом-распознавателем без выхода называется пятерка объектов:

$A = \{S, X, s_0, \varphi, T\}$,

где S – конечное непустое множество (состояний);

X – конечное непустое множество входных сигналов (входной алфавит);

$s_0 \in S$ – начальное состояние;

$\varphi : S \times X \to S$ – функция переходов;

$T \subseteq S$ – множество финальных состояний.

Будем считать, что конечный автомат–распознаватель $A=\{S,X,s_0,\varphi,T\}$ **допускает** входную цепочку $a\in X^*$, если a переводит его из начального состояния в одно из финальных состояний, т. е. $\varphi(s_0,a)\in T$. Множество всех цепочек, допускаемых автоматом A, образует язык L_A, допускаемый A.

Таким образом, если автомат находится в состоянии s и «читает» букву a, то (a,s) является входом для φ и $\varphi(a,s)$ – следующее состояние автомата. Выходом является состояние из S (возможно то же самое). Для автомата в качестве «начала» можно использовать состояние s_0. Если автомат «читает» букву a из X, то он переходит в состояние $s_1=\varphi(a,s_0)$. Если автомат теперь «читает» следующую букву из X, например b, то он переходит в состояние $s_2=\varphi(b,s_1)$ $\varphi(b,\varphi(a,s_0))$. Следовательно, по мере «прочтения» букв алфавита автомат переходит из одного состояния в другое.

Пример. Пусть M – автомат с алфавитом $X=\{a,b\}$, множеством состояний $S=\{s_0,s_1,s_2\}$, а функция переходов φ задана табл. 4.7.

Таблица 4.7

	s_0	s_1	s_2
a	s_1	s_1	s_2
b	s_2	s_2	s_1

Предположим, что M «читает» букву a, за которой следуют буквы b и a. Поскольку автомат начинает функционировать в состоянии s_0 и буква, которую он «читает», – это a, то $\varphi(a,s_0)=s_1$, поэтому теперь автомат находится в состоянии s_1. Следующей буквой для прочтения является b и $\varphi(b,s_1)=s_2$. Наконец, «читается» последняя буква a, и, так как $\varphi(a,s_2)=s_2$, автомат остается в состоянии s_2.

Приведенный выше автомат можно представить графически, как показано на рис. 4.5. Здесь дуга из вершины s в вершину s' помечена буквой из алфавита A, например буквой a, если $\varphi(a,s)=s'$.

Если из одного состояния в другое может вести не одна дуга, то такая диаграмма называется мультиграфом.

Слово $a_0a_1a_2\ldots a_n$ автомат «читает» слева направо, т. е. начинает с a_0 и заканчивает a_n. Автомат допускает или распознает слово

$a_0a_1a_2\ldots a_n$, если после «прочтения» он останавливается в финальном состоянии, которое, обычно обозначается двойной окружностью.

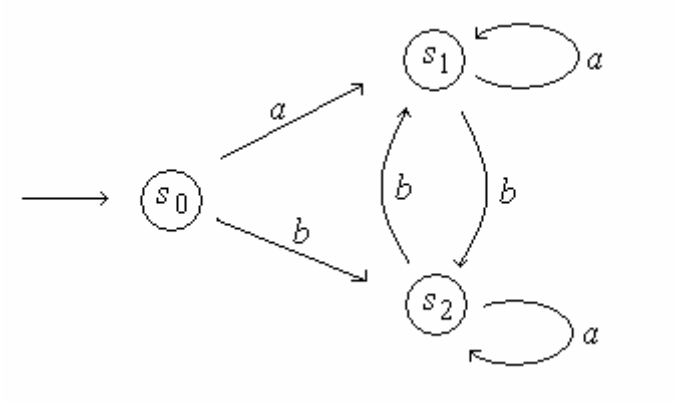


Рис. 4.5

Например, для автомата, диаграмма состояния которого изображена на рис. 4.6, состояние s_0 – начальное, а состояние s_3 – финальное.

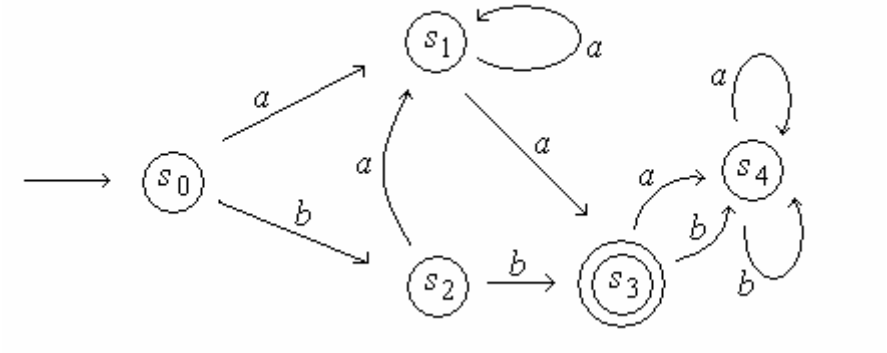


Рис. 4.6

Данный автомат допускает слово baa, поскольку после прочтения b он переходит в состояние s_2; после прочтения a – в состояние s_1; после прочтения второго a он переходит в состояние s_3, которое является финальным состоянием. Можно убедиться, что автомат допускает также слова $abbba$ и bb, но не распознает слова bbb, $abab$ или abb.

Пример. Рассмотрим автомат с диаграммой состояния, изображенной на рис. 4.7 [7].

Очевидно, что автомат допускает слово bb. Для буквы a в каждом состоянии имеется петля, поэтому при чтении a состояние не меняется, что дает возможность до прочтения следующей буквы b читать любое необходимое количество букв a. Таким образом, автомат читает $aababaaa$, $baaba$, $baaab$, $aabaabaa$ и может фактически прочесть любое слово языка, заданного регулярным выражением $a*ba*ba$. На-

помним, что $a*$ обозначает слово, содержащее произвольное число букв a. Поскольку $A=\{a,b\}$, то такой язык можно описать также как множество всех слов, содержащих точно два b. Состояние s_4 является примером состояния зацикливания. Попав в состояние зацикливания, автомат никогда не сможет из него выйти.

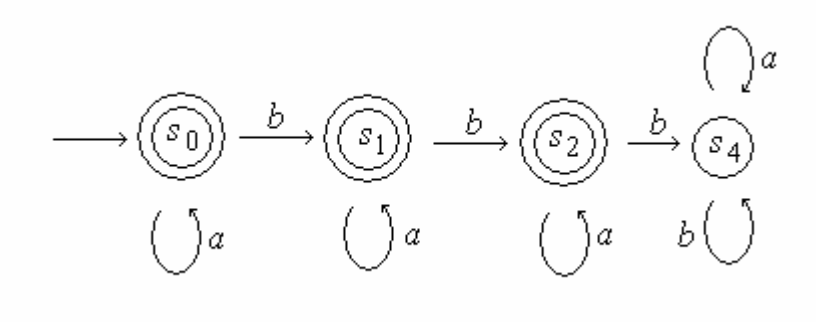


Рис. 4.7

Пример. Рассмотрим автомат с диаграммой состояний, изображенной на рис. 4.8.

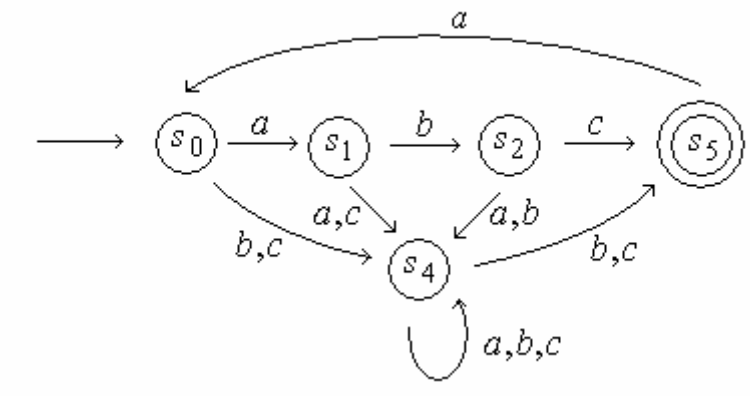


Рис. 4.8

Каждое слово следует начинать с ab и заканчивать на c. Однако петлю $aabc$ можно повторять требуемое количество раз, поскольку она начинается и заканчивается в s_3. Поэтому регулярным выражением для этого автомата будет $abc(aabc)*$.

Рассмотренные автоматы часто называются **детерминированными** автоматами, т. к. для любого исходного состояния и для любой буквы, которая подается в автомат для чтения, существует одно и только одно конечное состояние. Иными словами, $\varphi: A\times S\to S$ является функцией. Автоматы, для которых φ не обязательно является функцией, называются недетерминированными автоматами.

4.2. Элементы кодирования

Вопросы кодирования издавна играли заметную роль в математике [1]. В частности:

- десятичная позиционная система счисления – это способ кодирования натуральных чисел. Римские цифры – другой способ кодирования натуральных чисел;
- декартовы координаты – способ кодирования геометрических объектов числами.

Однако ранее средства кодирования играли вспомогательную роль и не рассматривались как отдельный предмет математического изучения, но с появлением компьютеров ситуация кардинально изменилась. Кодирование буквально пронизывает информационные технологии и является центральным вопросом при решении самых разных задач программирования:

- представление данных произвольной природы (чисел, текста, графиков) в памяти компьютеров;
- защита информации от несанкционированного доступа;
- обеспечение помехоустойчивости при передаче данных по каналам связи;
- сжатие информации в базах данных и т. д.

4.2.1. Формулировка задачи кодирования

Пусть заданы алфавиты A, B и функция $F : S \to B^*$, где S – некоторое множество слов в алфавите A, $S \subset A^*$. Функция F называется **кодированием**, элементы множества S – **сообщениями**, а элементы $\beta \quad F(\alpha)$, где $\alpha \in S$, $\beta \in B^*$, – **кодами** соответствующих сообщений. Обратная функция F^{-1} (если она существует) называется **декодированием**.

Если $|B| = m$, то F называется m–ичным кодированием. Наиболее распространен случай $B = \{0,1\}$ – двоичное кодирование. Именно этот случай рассматривается в дальнейшем.

Типичная задача теории кодирования формулируется следующим образом [1]: при заданных алфавитах A, B и множестве сообщений S найти такое кодирование F, которое обладает определенными свойствами и оптимально в некотором смысле.

Свойства, которые требуются от кодирования, бывают самой разной природы [1]:

• существование декодирования – естественное свойство, однако даже оно требуется не всегда. Например, трансляция программы на языке высокого уровня в машинные коды – это кодирование, для которого не требуется декодирования;

• помехоустойчивость, или исправление ошибок: функция декодирования F^{-1} обладает таким свойством, что $F^{-1}(\beta) \quad F^{-1}(\beta')$, если β в определенном смысле близко к β';

• заданная сложность (или простота) кодирования и декодирования. Например, в криптографии изучаются такие способы кодирования, при которых имеется просто вычисляемая функция F, но определение функции F^{-1} требует очень сложных вычислений.

4.2.1. Алфавитное (побуквенное) кодирование

Кодирование F может сопоставлять код всему сообщению из множества S как единому целому или же строить код сообщения из кодов его частей. Элементарной частью сообщения является одна буква алфавита A. Этот простейший случай, когда кодируется каждая буква сообщения S, называется **алфавитным кодированием**.

Если $\alpha = \alpha_1\alpha_2$, то α_1 называется префиксом (или началом) слова, а α_2 – постфиксом (концом) слова.

Алфавитное кодирование задается схемой (или таблицей кодов):

$$\sigma = (a_1 \to \beta_1,\ a_2 \to \beta_2, \ldots, a_n \to \beta_n), \quad a_i \in A, \quad \beta_i \in B^*.$$

Множество слов $V = \{\beta_i \,/\, \beta_i \in B^*\}$ называется множеством элементарных кодов. Таким образом, буква a_i **кодируется словом** в алфавите B^*.

Алфавитное кодирование пригодно для любого множества сообщений S:

$$F:\ A^* \to B^*,\ = a_{i_1} \ldots a_{i_k} \quad \alpha \in A^*, \quad F(\alpha) \quad \beta_{i_1} \ldots \beta_{i_k}.$$

Пример. Рассмотрим алфавиты $A = \{0,1,2,3,4,5,6,7,8,9\}$ и схему

$\sigma \quad (0 \to 0,\ 1 \to 1,\ 2 \to 10,\ 3 \to 11,\ 4 \to 100,\ 5 \to 101,$

$6 \to 110,\ 7 \to 111,\ 8 \to 1000,\ 9 \to 1001)$.

Эта схема однозначна, но кодирование не является взаимно однозначным. Например, $F_\sigma(33) = 111111 = F_\sigma(77)$, а значит, невозможно декодирование. С другой стороны, схема

$\sigma \quad (0 \to 0000,\ 1 \to 0001,\ 2 \to 0010,\ 3 \to 0011,\ 4 \to 0100,$

$5 \to 0101,\ 6 \to 0110,\ 7 \to 0111,\ 8 \to 1000,\ 9 \to 1001),$

известная под названием «двоично-десятичное кодирование», допускает однозначное декодирование.

Схема алфавитного кодирования σ называется **разделимой**, если любое слово, составленное из элементарных кодов, единственным образом разлагается на элементарные коды. Алфавитное кодирование с разделимой схемой допускает декодирование.

Схема σ называется **префиксной**, если элементарный код одной буквы **не является** префиксом элементарного кода другой буквы.

Теорема. Префиксная схема является разделимой.

Чтобы схема алфавитного кодирования была разделимой, необходимо, чтобы длины элементарных кодов удовлетворяли определенному соотношению, известному как **неравенство Макмиллана** [1].

Теорема. Если схема $\sigma = (\alpha_i \to \beta_i)_{i=1}^{n}$ разделима, то $\sum_{i=1}^{n} \frac{1}{2^{k_i}} \le 1$, где $k_i = |\beta_i|$.

Неравенство Макмиллана является не только необходимым, но и достаточным условием разделимости схемы алфавитного кодирования. Поэтому теорему можно переформулировать следующим образом.

Теорема. Если числа $k_1,\ k_2,\ldots,k_n$ удовлетворяют неравенству $\sum_{i=1}^{n} \frac{1}{2^{k_i}} \le 1$, то существует разделимая схема алфавитного кодирования $\sigma = (\alpha_i \to \beta_i)_{i=1}^{n}$, где $\forall i = k_i \quad |\beta_i|$.

Пример. Для схемы кодирования σ $(0 \Rightarrow 0,\ 1 \to 1,\ 2 \to 10,$
$3 \to 11,\ 4 \to 100,\ 5 \to 101,\ 6 \to 110,\ 7 \to 111,\ 8 \to 1000,\ 9 \to 1001)$
левая часть неравенства Макмиллана может быть записана в виде

$$2\frac{1}{2^1} + 2\frac{1}{2^2} + 4\frac{1}{2^3} + 2\frac{1}{2^4} \quad 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{8} = 2\frac{1}{2}.$$

Отсюда следует, что данная схема кодирования не разделима.

Для случая двоично-десятичного кодирования –
σ $(0 \to 0000,\ 1 \to 0001,\ 2 \to 0010,$
$3 \to 0011,\ 4 \to 0100,\ 5 \to 0101,$
$6 \to 0110,\ 7 \to 0111,\ 8 \to 1000,\ 9 \to 1001)$ – имеем

$$\frac{10}{2^4} = \frac{10}{16} < 1,$$

что говорит о разделимости схемы кодирования.

Пример. Схему кодирования, лежащую в основе азбуки Морзе, можно записать как

σ ($A \Rightarrow 01,\ B \to 1000,\ C \to 1010,\ D \to 100,\ E \to 0,\ F \to 0010,$
$G \to 110,\ H \to 0000,\ I \to 00, J \to 0111,\ K \to 101,\ L \to 0100,\ M \to 11,$
$N \to 10,\ O \to 111,\ P \to 0110,\ Q \to 1101,\ R \to 010,\ S \to 000,\ T \to 1,$
$U \to 001,\ V \to 0001,\ W \to 011,\ X \to 1001,\ Y \to 1011,\ Z \to 1100$),
где по историческим и техническим причинам 0 называется точкой, а 1 – тире. Проведя проверку на разделимость, получим

$$2\frac{1}{2^1} + 4\frac{1}{2^2} + 8\frac{1}{2^3} + 12\frac{1}{2^4} \quad 3\frac{5}{8} > 1.$$

Таким образом, схема азбуки Морзе не является разделимой. На самом деле, в азбуке Морзе используются дополнительные элементы – паузы между буквами и словами, что позволяет декодировать сообщение.

4.2.3. Кодирование с минимальной избыточностью

Для практики важно, чтобы коды сообщений имели по возможности наименьшую длину. Алфавитное кодирование пригодно для любых типов сообщений S. Если про S более ничего не известно, то сформулировать задачу оптимизации сложно. Однако на практике часто имеется дополнительная информация. Например, для текстов на естественных языках известно распределение вероятности появления букв в сообщении. Использование такой информации позволяет корректно поставить и решить задачу оптимизации алфавитного кодирования.

Идея оптимизации алфавитного кодирования может состоять в том, чтобы наиболее часто применяемым буквам входного алфавита A сопоставить наиболее короткие элементарные коды. Если длины элементарных кодов равны, как в случае двоично–десятичного кодирования, то данный подход не имеет смысла. Но если длины элементарных кодов различны, то длина кода сообщения зависит от состава букв в сообщении и от того, какие элементарные коды каким буквам назначены.

Алгоритм назначения элементарных кодов может быть следующий: нужно отсортировать буквы, входящие в сообщение S, в порядке убывания количества вхождений; элементарные коды отсортировать в порядке возрастания длины и назначить коды буквам в этом порядке. Естественно, что этот простой метод позволяет решить задачу оптимизации лишь для конкретного сообщения S и конкретной схемы кодирования σ.

Рассмотрим количественную оценку, позволяющую сравнивать между собой различные схемы алфавитного кодирования. Пусть задан алфавит $A = \{a_1, a_2, \ldots, a_n\}$ и вероятность появления букв в сообщении $P = \{p_1, p_2, \ldots, p_n\}$, где p_i – вероятность появления буквы a_i. Будем считать, что $p_1 + p_2 + \ldots + p_n = 1, p_1 \geq p_2 \geq \ldots \geq p_n$.

Для каждой разделимой схемы $\sigma = (a_i \to \beta_i)_{i=1}^{n}$ алфавитного кодирования математическое ожидание длины сообщения при кодировании h_σ определяется следующим образом:

$$h_\sigma(P) = \sum_{i=1}^{n} p_i h_i , \qquad (4.1)$$

где $h_i = |\beta_i|$, и называется **средней ценой кодирования** σ при распределении вероятностей P [1].

Пример. Для разделимой схемы $A = \{a, b\}$, $B = \{0,1\}$, σ $\{a \Rightarrow 0,\ b \to 01\}$, при распределении вероятностей $P = \{0{,}5;\ 0{,}5\}$, цена кодирования h равна 0,5*1+0,5*2=1,5; а при распределении вероятностей {0.9,0.1} она равна 0.9*1+0.1*2=1,1.

Алфавитное кодирование σ^*, для которого средняя цена кодирования минимальна, называется кодированием с минимальной избыточностью, или оптимальным кодированием, для распределения вероятности P.

4.2.4. Алгоритм квазиоптимального кодирования Фано

Рассмотрим два метода построения схем кодирования, принадлежащих Фано и Шеннону [1]. Рекурсивный алгоритм Фано отличается чрезвычайной простотой конструкции и строит разделимую префиксную схему алфавитного кодирования, близкого к оптимальному. Метод Фано заключается в следующем: упорядоченный в порядке убывания вероятностей список букв делится на две последовательные части так, чтобы суммы вероятностей входящих в них букв как можно меньше отличались друг от друга. Буквам из первой части приписывается символ, а буквам из второй части – символ 1. Далее точно так же поступают с каждой из вновь полученных частей, если она содержит по крайней мере две буквы. Процесс продолжается до тех пор, пока весь список не разобьется на части, содержащие по одной букве. Каждой букве ставится в соответствие последовательность символов, приписанных в результате этого процесса данной букве. Легко видеть, что полученный в результате код является префиксным.

В табл. 4.8 приведен пример построения схемы кодирования с использованием алгоритма Фано для алфавита, включающего 7 букв. Буквы встречаются в сообщениях с вероятностями $P = \{0{,}20;0{,}20;0{,}19;0{,}12;0{,}11;0{,}09;0{,}09\}$. **Список букв упорядочен по убыванию вероятностей!**

На первом этапе список букв был разделен на две части. В первую часть вошли буквы с вероятностями $P_1 = \{0,20;0,20;0,19\}$. Во вторую – с вероятностями $P_2 = \{0,12;0,11;0,09;0,09\}$. Общая сумма вероятностей первой часть – 0,59, второй – 0,41. Принятое разбиение обеспечивает минимальную разницу суммарных вероятностей двух частей, равную 0,18. Если принять иное разбиение, например $P_1 = \{0,20;0,20\}$ и $P_2 = \{0,19;0,12;0,11;0,09;0,09\}$, то разница будет равной 0,20. В результате первым символом в кодах трех первых букв принимается 0, а оставшихся четырех – 1.

Далее, первая группа букв снова делится на две части – $P_{11} = \{0,20\}$ и $P_{12} = \{0,20;0,19\}$. В кодах букв первой части вторым символом принимается 0, в кодах букв второй части – 1. Так как первая часть содержит единственную букву, для нее построение кода закончилось. Вторая часть содержит две буквы, и им просто присваиваются разные последние кодирующие символы. Точно такая же процедура производится в отношении второй половины списка букв, полученной в результате первого деления. В нижней строке таблицы, во втором столбце, стоит значение средней цены кодирования, расчитанной для принятой схемы кодирования по формуле (4.1).

Таблица 4.8

p_i	Метод Фано		Метод Хаффмена	
	Коды	h_i	Коды	h_i
0,20	00	2	10	2
0,20	010	3	11	2
0,19	011	3	000	3
0,12	100	4	010	3
0,11	101	4	011	3
0,09	110	4	0010	4
0,09	111	4	0011	4
Цена кодирования	2,80		2,78	

4.2.5. Алгоритм оптимального кодирования Хаффмена

Метод Фано обеспечивает построение префиксных кодов, стоимость которых весьма близка к оптимуму. Хаффмен предложил несколько более сложный, индуктивный, метод, в результате применения которого получается префиксный код, стоимость которого оптимальна [1]. Метод Хаффмена опирается на несколько лемм и теорем, важнейшими из которых являются следующие.

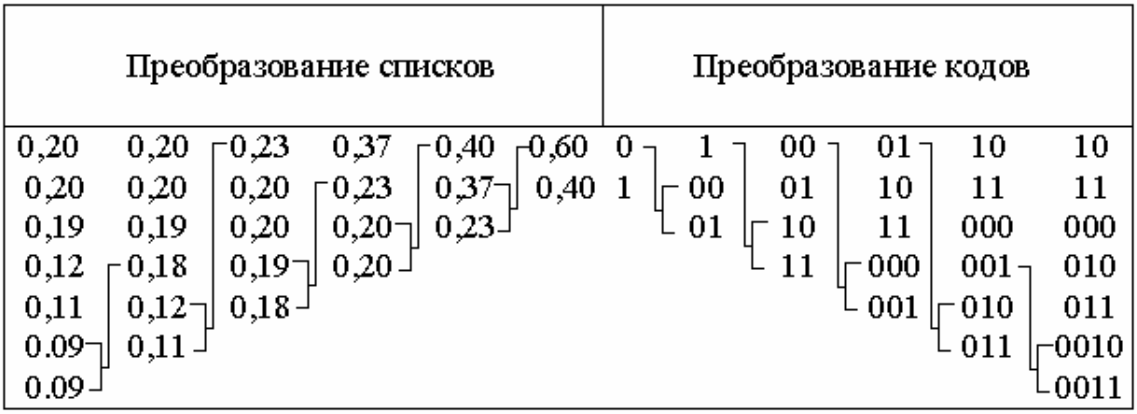
Лемма. Пусть $\sigma = (a_i \to \beta_i)_{i=1}^{n}$ – схема оптимального кодирования для распределения вероятностей $P = \ p_1 \geq p_2 \geq \ldots \geq p_n > 0$. Тогда если $p_i > p_j$, то $h_i \leq h_j$.

Теорема. Если $\sigma_{\overline{n}-1} \quad (a_i \to \beta_i)_{i=1}^{n-1}$ – схема оптимального префиксного кодирования для распределения вероятностей $P = \ p_1 \geq p_2 \geq \ldots \geq p_{n-1} > 0$ и $p_j = q' + q''$, причем $p_1 \geq \ldots \geq p_{j-1} \geq p_{j+1} \geq \ldots \geq p_{n-1} \geq q' \geq q'' > 0$, то кодирование со схемой $\sigma_{\overline{n}} \ \ (a_1 \to \beta_1, \ \ldots \ , a_{j-1} \to \beta_{j-1}, \ a_{j+1} \to \beta_{j+1}, \ldots, a_{n-1} \to$ $\to \beta_{n-1}, \ a_j \to \beta_j 0, \ a_n \to \beta_j 1)$ является оптимальным префиксным кодированием для распределения вероятностей $P_n = \ p_1 \ldots p_{j-1}, p_{j+1}, \ldots, p_{n-1}, q', q''$.

Данная теорема определяет следующий способ построения оптимального кода. В исходном, упорядоченном по убыванию, списке вероятностей отбрасываются две последние (наименьшие) вероятности, а их сумма вставляется в список таким образом, чтобы получающийся список из $n-1$ вероятностей снова был упорядочен по убыванию. Затем эта же процедура повторяется со списком из $n-1$ вероятностей и т. д. до тех пор, пока не получится список из двух вероятностей. Первой из получившихся вероятностей приписывается символ 0, а второй – символ 1.

Затем, согласно вышеприведенной теореме, из оптимального кода для двух букв строится оптимальный код для трех букв при соответствующем списке вероятностей и т. д. до тех пор, пока не получится оптимальный код при исходном списке вероятностей. Описанный метод Хаффмена иллюстрируется в табл. 4.9 последовательностью преобразований вероятностей для того же примера, что и в случае с алгоритмом Фано. Схема кодирования и средняя цена кодирования приведены в табл. 4.8. Цена оптимального кодирования по Хаффмену составляет 0,20 * 2 + 0,20 * 2 + 0,19 * 3 + 0,12 * 3 + 0,11 * 3 + 0,09 * 4 +0,09*4=2,78, что несколько лучше, чем для схемы, полученной с помощью алгоритма Фано.

Таблица 4.9



4.2.6. Помехоустойчивое кодирование

Хотя надежность электронных устройств все время возрастает, в их работе возможны ошибки. Сигнал в канале связи может быть искажен помехой, поверхность магнитного носителя может быть повреждена, в разъеме может быть потерян контакт. Ошибки аппаратуры ведут к искажению или потере данных. При определенных условиях можно применять методы кодирования, позволяющие правильно декодировать исходное сообщение, несмотря на ошибки в данных кода. В качестве исследуемой модели достаточно рассмотреть канал связи с помехами, потому что к этому случаю легко сводятся остальные.

Пусть имеется канал связи C, содержащий источник помех:

$$S \overset{C}{\to} S' \ , \quad S \in A^* , \quad S' \in D^* ,$$

где S – множество переданных, а S' – соответствующее множество принятых по каналу сообщений. Кодирование F, обладающее таким свойством, что

$$S \overset{F}{\to} K \overset{C}{\to} K' \overset{F^{-1}}{\to} S \ , \quad \forall s \in S, \quad F^{-1}(C(F(s))) = s ,$$

называется помехоустойчивым, или самокорректирующимся, или кодированием с исправлением ошибок.

Далее будем считать, без потери общности, что $A = D = \{0,1\}$ и что содержательное кодирование выполняется на устройстве, свободном от помех.

Ошибки в канале могут быть следующих типов:

$0 \to 1$, $1 \to 0$ – ошибка типа замещение разряда;

$0 \to \Lambda$, $1 \to \Lambda$ – ошибка типа выпадения разряда;

$\Lambda \to 0$, $1 \to \Lambda$ – ошибка типа вставки разряда.

Канал характеризуется верхними оценками количества ошибок каждого типа. Общая характеристика ошибок канала (т. е. их количество и типы) обозначается как δ.

Пример. Характеристика $\delta = (1,0,0)$ означает, что в канале возможна одна ошибка типа замещения разряда при передаче сообщения длины n.

Пусть E_δ^s – множество слов, которые могут быть получены из слова s в результате всех возможных комбинаций, допустимых в канале ошибок, т.е. $s \in S \subset A^*$, $E_\delta^s \subset D^*$. Если $s' \in E_\delta^s$, то та конкретная последовательность ошибок, которая позволяет получить из слова s слово s', обозначается $E_\delta(s,s')$.

Теорема. Чтобы существовало помехоустойчивое кодирование с исправлением всех ошибок, необходимо и достаточно, чтобы $\forall s_1, s_2 \in S$, $E^{s_1} \cap E^{s_2} = 0$, т. е. необходимо и достаточно, чтобы существовало разбиение множества D^* на подмножества D_s, такое, что $\forall s \in S$, $E_s \subset D_s$, и при этом выполнялись условия $\bigcup D_s = D^*$, $\bigcap D_s = \varnothing$.

Говоря проще, у каждого слова s должна существовать своя собственная «область», никакая ошибка не должна выводить слово за пределы этой области и разные области не должны пересекаться. Понятно, что чем больше размеры этой области, тем надежнее может быть распознавание слова.

4.2.7. Сжатие данных

При алфавитном кодировании наблюдается некоторый баланс между временем и памятью. Затрачивая дополнительные усилия при кодировании и декодировании, можно экономить память, и, наоборот, пренебрегая оптимальным использованием памяти, можно существенно выиграть во времени кодирования и декодирования. Конечно, этот баланс имеет место только в определенных пределах, и нельзя сократить расход памяти до нуля или построить мгновенно работающие алгоритмы. Для алфавитного кодирования пределы возможного установлены оптимальным алгоритмом, рассмотренным выше. Для достижения дальнейшего прогресса нужно рассмотреть неалфавитное кодирование.

Определение. Методы кодирования, которые позволяют построить (без потери информации) коды сообщений, имеющие меньшую длину по сравнению с исходным сообщением, называются **методами сжатия**

(или упаковки) информации. Качество сжатия обычно определяется **коэффициентом сжатия**, измеряется в процентах и показывает, насколько сжатое сообщение короче исходного.

Допустим, имеется некоторое сообщение, которое закодировано каким-то общепринятым способом и хранится в памяти ЭВМ. Например, текст в кодах ASCII. Заметим, что равномерное кодирование, используемое в кодах ASCII, не является оптимальным для текстов, т. к. в текстах обычно используется существенно меньше, чем 256 символов. Обычно это 60–70 символов, в зависимости от языка.

Если вероятности появления различных букв различны и известны, то можно, воспользовавшись алгоритмом Хаффмена, построить для того же самого сообщения схему оптимального алфавитного кодирования (для заданного алфавита и языка). Расчеты показывают [1], что такое кодирование будет иметь цену несколько меньше 6, т. е. даст выигрыш по сравнению с кодом ASCII примерно на 25 %. Известно, однако, что практические архиваторы (программы сжатия) имеют гораздо лучшие показатели (до 70 % и более). Это означает, что в них используется не алфавитное кодирование.

Рассмотрим следующий способ кодирования.

1. Исходное сообщение по некоторому алгоритму разбивается на последовательности символов, называемых словами (слово может иметь одно или несколько вхождений в текст сообщения).

2. Полученное множество считается буквами нового алфавита. Для этого алфавита строится разделимая схема алфавитного кодирования (равномерного или оптимального). Полученная схема обычно называется словарем, т. к. сопоставляет слову код.

3. Далее код сообщения строится как пара – код словаря и последовательность кодов слов из данного словаря.

4. При декодировании исходное сообщение восстанавливается путем замены кодов слов на слова из словаря.

Пример. Требуется сжать текст на русском языке. В качестве алгоритма деления на слова примем обычные правила языка: слова отделяются друг от друга пробелами или знаками препинания. Можно принять допущение, что в каждом конкретном тексте имеется не более 2^{16} различных слов (обычно гораздо меньше). Таким образом, каждому слову можно сопоставить код – целое число из двух байт (равномерное кодирование). Учитывая, что каждый символ в ASCII кодируется одним байтом, полученный код слова по объму эквивалентен кодам двух букв русского алфавита. Поскольку в среднем слова русского языка состоят более чем из двух букв, такой способ позволяет сжать текст на 75 % и

более. При больших текстах расходы на хранение словаря относительно невелики.

Данный метод попутно позволяет решить задачу полнотекстового поиска, причем для этого не нужно просматривать весь текст, достаточно просмотреть словарь.

Указанный способ можно усовершенствовать по крайней мере в двух отношениях. На шаге 2 можно использовать алгоритм оптимального кодирования, а на шаге 1 – решить экстремальную задачу такого разбиения сообщения на слова, чтобы цена кодирования на шаге 2 была минимальной. Однако на практике такая экстремальная задача весьма трудоемка и временные затраты оказываются слишком большими.

СПИСОК ЛИТЕРАТУРЫ

1. Дискретная математика и математические вопросы кибернетики / под ред. С.В. Яблонского и О.Б. Лупанова. – М.: «Наука», 1974. – 311 с.

2. Новиков Ф.А. Дискретная математика для программистов. – СПб.: Питер, 2000. – 304 с.

3. Карпов Ю.Г. Теория автоматов – СПб.: Питер, 2002. – 224 с.

4. Горбатов В.А. Основы дискретной математики. – М., Высш. шк., 1986. – 310 с.

5. Москинова Г.И. Дискретная математика. – М., «Логос», 2000. – 236 с.

6. Триханов А.В. Теория автоматов: учебное пособие по курсовому проектированию. – Томск: Изд-во ТПУ, 2000. – 135 с.

7. Андерсон Джеймс А. Дискретная математика и комбинаторика: пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 960 с.

Учебное издание

ВОРОНИН Александр Васильевич

ДИСКРЕТНАЯ МАТЕМАТИКА

Учебное пособие

Научный редактор
доктор технических наук,
профессор *А.М. Малышенко*

Редактор *Н.Т.Синельникова*

Верстка *Л.А. Егорова*

Подписано к печати Формат 60×84/16.
Бумага «Снегурочка». Печать Xerox.
Усл. печ.л. 6,74. Уч.-изд.л. 6,11.
Заказ . Тираж экз.

Томский политехнический университет
Система менеджмента качества
Томского политехнического университета
сертифицирована
NATIONAL QUALITY ASSURANCE
по стандарту ISO 9001:2000

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30.