Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Тульский государственный университет»

В.П. БАРАНОВ

ДИСКРЕТНАЯ МАТЕМАТИКА

Учебное пособие

Издательство ТулГУ

Тула 2013

УДК 517.5

Баранов В.П. **Дискретная математика:** Учебное пособие. – Тула: Изд-во ТулГУ, 2013. – 215 с.

В пособии изложены основные разделы дискретной математики: понятия теории множеств, комбинаторика, теория графов и сетей, элементы теории кодирования, алгебра логики и ее приложения, конечные автоматы, элементы теории алгоритмов.

Пособие предназначено для студентов специальностей 010400 «Прикладная математика и информатика», 010200 «Математика и компьютерные науки», 010800 «Механика и математическое моделирование», а также может быть полезно студентам, магистрам и аспирантам других направлений и специальностей, использующих методы дискретной математики.

Табл. 21. Ил. 56. Библиогр.: 16 назв.

Печатается по решению библиотечно-издательского совета Тульского государственного университета

Рецензенты: кафедра алгебры, математического анализа и геометрии Тульского государственного педагогического университета им. Л.Н. Толстого (зав. кафедрой д-р физ.-мат. наук, проф. Н.М. Добровольский);

д-р физ.-мат. наук, проф. кафедры математического моделирования Тульского государственного университета В.В. Глаголев

ОГЛАВЛЕНИЕ

Предисловие

Дискретная математика – область математики, занимающаяся изучением дискретных структур, которые возникают как в пределах самой математики, так и в ее приложениях. В качестве синонима иногда употребляется термин «дискретный анализ». Как следует из названия дисциплины, главной ее спецификой является дискретность, которая является антиподом непрерывности. Дискретность и непрерывность вместе составляют единство, которое характерно для математики. В качестве примера можно привести числовые системы: множество натуральных чисел – дискретный объект, а множество действительных чисел – непрерывный. Другой пример: при численном решении какой-либо задачи, связанной с исследованием и расчетом непрерывного объекта (расчет траектории движения материального тела, определение формы поверхности, преобразование непрерывного сигнала, анализ временных рядов и др.) главным этапом является дискретизация задачи, т. е. выбор адекватной дискретной модели, результаты расчета которой с заданной степенью точности позволяют найти искомые величины в непрерывной задаче.

Классическая математика – это математика непрерывных величин. Основное понятие классической математики, понятие предела, связано с представлением о непрерывной действительной прямой – континууме. Основная модель классической математики – система дифференциальных уравнений, описывающая движение по непрерывной траектории в фазовом пространстве.

Элементы дискретной математики зародились в рамках классической, но до середины XX века не занимали в ней заметного места.

В широком смысле дискретная математика включает в себя и такие сложившиеся разделы математики, как теория чисел, алгебра, математическая логика и ряд разделов, которые наиболее интенсивно стали развиваться в середине XX века в связи с научно-техническим прогрессом, прежде всего связанным с широким использованием компьютеров. Последнее привело к

необходимости изучения сложных управляющих систем. В узком смысле слова дискретная математика ограничивается только этими новыми разделами. Именно в узком смысле понимается дискретная математика в данном учебном пособии. К упомянутым новым разделам, составляющим содержание книги, относятся: комбинаторный анализ, теория графов и сетей, теория кодирования, теория функциональных систем (теория булевых функций), теория конечных автоматов, языков и грамматик, теория алгоритмов.

В настоящее время дискретная математика является не только фундаментом математической кибернетики, но и важным звеном математического образования.

1. Введение в теорию множеств

1.1. Основные понятия теории множеств

Понятие *множества* является первичным и поэтому формально не может быть определено. Обычно множество объясняют, следуя основателю теории множеств Г. Кантору, как совокупность объектов произвольной природы, рассматриваемую, как единое целое.

Объекты, составляющие множество, называют его *элементами*. Множества обозначают прописными буквами латинского алфавита (X, Y, ...), элементы множеств – строчными буквами (x, y, ...). Утверждение «элемент x принадлежит множеству X» записывается следующим образом: $x \in X$ ($\in$ – символ принадлежности). В противном случае – $x \notin X$ (или $x \overline{\in} X$).

Если каждый элемент множества X входит в множество Y, то X называется *подмножеством* Y. При этом пишут: $X \subseteq Y$ или $Y \supseteq X$. Если $X \subseteq Y$ и $X \neq Y$, то пишут $X \subset Y$ или $Y \supset X$. Здесь $\subseteq, \subset, \supseteq, \supset$ – символы включения. Множества X и Y равны, если они состоят из одних и тех же элементов, иначе говоря, если $X \subseteq Y$ и $Y \subseteq X$.

Множество, не содержащее ни одного элемента, называется *пустым* (обозначается $\varnothing$). Множество называется *истинным*, если оно не пустое.

Из определения подмножества следует, что каждое множество является подмножеством самого себя: $X \subseteq X$. Кроме того, считают, что пустое множество есть подмножество любого множества X: $\varnothing \subset X$.

Различают два вида подмножеств множества X: само X и $\varnothing$ называют *несобственными* подмножествами; все остальные подмножества, если они существуют, – *собственными*.

В теории множеств для удобства и краткости записей используют специальные обозначения:

$\forall$ – квантор общности (означающий «любой», «для всех», «каков бы ни был»);

$\exists$ – квантор существования (означающий «существует», «найдется», «можно найти»);

$\Rightarrow$ – импликация (символ следствия, означающий «влечет за собой»).

С помощью этих символов условие $X \subseteq Y$ запишется так:

$$\forall x : x \in X \Rightarrow x \in Y .$$

Множество X называется *эквивалентным* множеству Y (обозначается $X \sim Y$), если между элементами X и Y можно установить взаимно однозначное соответствие. Различают *конечные* и *бесконечные* множества. Число элементов множества X называется его *мощностью* или *кардинальным числом* и обозначается $|X|$ или $\operatorname{card} X$ (англ. *cardinality* – мощность). Таким образом, конечное множество, содержащее n элементом, имеет мощность $|X| = n$. Если X эквивалентно множеству натуральных чисел $\mathbb{N}$, то его называют *счетным* и его мощность обозначается через $\aleph_0$. Множество X, эквивалентное множеству действительных чисел $\mathbb{R}$, называется *континуальным*, а его кардинальное число *c – мощностью континуума*.

1.2. Способы задания множеств

Для задания множества существуют различные способы. Множество считают заданным, если о каждом элементе можно сказать, принадлежит он данному множеству или нет.

1) *Задание множества с использованием общепринятых обозначений*. Для числовых множеств имеем: $\mathbb{N}$ – множество натуральных чисел; $\mathbb{Z}$ – множество целых чисел; $\mathbb{Q}$ – множество рациональных чисел; $\mathbb{R}$ – множество действительных чисел; $\mathbb{C}$ – множество комплексных чисел.

2) *Задание множества перечислением его элементов*. Конечное множество можно задать перечислением его элементов и записать в виде

$$X = \{x_1, x_2, ..., x_n\}.$$

Например, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} – множество десятичных цифр.

3) *Задание множества с помощью характеристического свойства его элементов*. Характеристическое свойство – это свойство, которым обладают все элементы данного множества и только они. Этот способ применим для конечных и бесконечных множеств.

Пусть $P(x)$ – утверждение, заключающееся в том, что элемент x обладает свойством P. Тогда запись

$$X = \{x \mid P(x)\}$$

означает, что рассматривается множество всех элементов x, обладающих свойством P.

Например, отрезок [0, 1] действительной прямой можно определить следующим образом

$$[0,\ 1] = \{x \mid x \in \mathbb{R},\ 0 \le x \le 1\}.$$

4) *Рекурсивное задание множества*. Этот способ заключается в следующем:

1°. Указываются некоторые исходные элементы, входящие в множество.

2°. Описывается механизм, позволяющий получить новые элементы из имеющихся.

3°. Объявляется, что в множестве нет никаких других объектов кроме тех, которые можно получить из исходных, применяя описанный в п. 2° механизм.

Например, множество $X = \{1,\ 2,\ 2^2,\ 2^3,\ ...\} \subset \mathbb{R}$ можно задать рекурсивно:

1°. $x = 1 \in X$.

2°. $\forall x \in X \Rightarrow 2 \cdot x \in X$.

3°. X – наименьшее подмножество натурального ряда, удовлетворяющее условиям 1° и 2°.

Условие 3° определяет X как пересечение всех множеств, удовлетворяющих условиям 1° и 2°.

В дальнейшем при рекурсивном задании множеств последний пункт, как правило, не указывается, но всякий раз это требование подразумевается.

В качестве еще одного примера рекурсивного задания множества определим совокупность всех слов в данном алфавите.

Пусть $X = \{x_1, x_2, ..., x_m\}$ – произвольное конечное множество, элементы которого будем называть буквами, а само множество алфавитом. Элементы определяемого множества X^* будем называть словами.

1°. Каждая буква является словом: $x_i \in X^*$, $i = 1, 2, \ldots, m$.

2°. Результат приписывания к слову любой буквы является словом:

$$\forall x \in X^* \Rightarrow xx_1 \in X^*,\ xx_2 \in X^*,\ \ldots,\ xx_m \in X^*.$$

Например, если $X = \{0, 1\}$, то X^* содержит следующие элементы:

0, 1 (согласно 1°);

00, 01, 10, 11 (согласно 2°);

000, 001, 010, 011, 100, 101, 110, 111 (согласно 2°) и т.д.

Теория множеств, рассматриваемая без ограничений на способы задания множеств, называется *наивной теорией множеств*. В этой теории еще при жизни ее создателя Г. Кантора были обнаружены многочисленные парадоксы. Приведем один из известных парадоксов Б. Рассела, открытый им в 1903 г.

Пусть M – множество всех множеств, которые не содержат себя в качестве своего элемента. Содержит ли множество M само себя в качестве элемента? Если да, то, по определению M, оно не должно быть элементом M – противоречие. Если нет – то, по определению M, оно должно быть элементом M – вновь противоречие.

Этот парадокс имеет много популярных формулировок. Приведем некоторые из них.

1) Одному полковому брадобрею приказали «брить всякого, кто сам не бреется, и не брить того, кто сам бреется». Как он должен поступить с собой?

2) В одной стране вышел указ: «Мэры всех городов должны жить не в своем городе, а в специальном Городе мэров». Где должен жить мэр Города мэров?

3) Некая библиотека решила составить библиографический каталог, в который входили бы все те и только те библиографические каталоги, которые не содержат ссылок на самих себя. Должен ли такой каталог включать ссылку на себя?

Для преодоления противоречий в наивной теории множеств было предложено несколько возможных ее аксиоматизаций, в рамках которых утверждение о существовании *множества всех множеств* было бы невыводимым.

1.3. Операции над множествами

Во многих случаях удается избежать противоречий наивной теории множеств, если выбрать некоторое так называемое *универсальное множество* U и ограничиться рассмотрением только его подмножеств.

Если некоторые множества взять в качестве исходных, то из них можно получить новые с помощью следующих операций.

Объединением множеств X и Y (обозначение $X \bigcup Y$) называется множество, состоящее из всех тех элементов, которые принадлежат хотя бы одному из множеств X или Y:

$$X \bigcup Y = \{x \mid x \in X \vee x \in Y\}.$$

Вместо символа объединения $\bigcup$ используется также +.

Пересечением множеств X и Y (обозначение $X \bigcap Y$) называется множество, состоящее из элементов, принадлежащих каждому из множеств X и Y:

$$X \bigcap Y = \{x \mid x \in X \wedge x \in Y\}.$$

Для операции пересечения используются также другие обозначения:

$$X \bigcap Y = X \cdot Y = XY.$$

Аналогично определяются объединение и пересечение произвольной совокупности множеств X_α, $\alpha \in A$. Здесь A – множество индексов. Если A – множество n первых натураль-

ных чисел, то употребляются обозначения $\bigcup_{i=1}^{n} X_i$ и $\bigcap_{i=1}^{n} X_i$, а в случае если $A = N$, то будем писать $\bigcup_{i=1}^{\infty} X_i$ и $\bigcap_{i=1}^{\infty} X_i$.

Разностью множеств X и Y (обозначение $X \setminus Y$) называется множество, состоящее из всех элементов X, не принадлежащих Y:

$$X \setminus Y = \{x \mid x \in X \wedge x \notin Y\}.$$

В отличие от двух предыдущих операций разность двухместна и некоммутативна: $X \setminus Y \neq Y \setminus X$. Если $X \setminus Y = \varnothing$, то $X \subseteq Y$.

Симметрической разностью множеств X и Y (обозначение $X \Delta Y$) называется множество элементов X и Y, которые содержатся только в одном из этих множеств:

$$X \Delta Y = \{x \mid x \in (X \setminus Y) \bigcup (Y \setminus X)\}.$$

Дополнением к множеству X (обозначение $\bar{X}$) относительно универсального множества U называется множество:

$$\bar{X} = U \setminus X \text{ или } \bar{X} = \{x \mid x \notin X\}.$$

Операции объединения, пересечения и дополнения часто называют *булевыми операциями* над множествами. Так как операция разности не обладает свойством ассоциативности, то ее выражают через другие операции, например, операции дополнения и пересечения:

$$A \setminus B = A\bar{B}.$$

Универсальное множество позволяет геометрически изображать множества и операции над ними с помощью *диаграмм Венна* (рис. 1.1).

1.4. Теоретико-множественные тождества

Пусть U – универсальное множество, а X, Y, Z – его подмножества. Тогда имеют место следующие тождественные равенства.

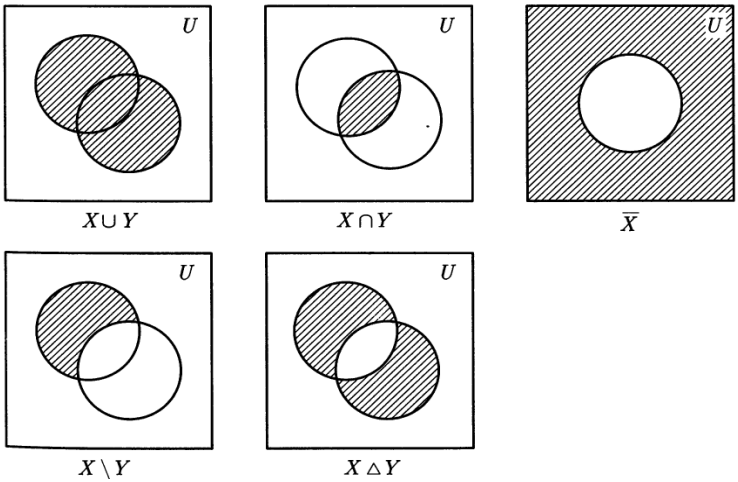


Рис. 1.1. Диаграммы Венна

1. $(X \cup Y) \cup Z = X \cup (Y \cup Z)$
2. $(X \cap Y) \cap Z = (X \cup Y) \cap Z$

– ассоциативность объединения и пересечения.

3. $X \cup Y = Y \cup X$
4. $X \cup Y = Y \cup X$

– коммутативность объединения и пересечения.

5. $(X \cup Y) \cap Z = (X \cap Z) \cup (Y \cap Z)$
6. $(X \cap Y) \cup Z = (X \cup Z) \cap (Y \cup Z)$

– дистрибутивность.

7. $X \cup X = X$
8. $X \cap X = X$

– идемпотентность.

9. $\overline{X \cup Y} = \overline{X} \cap \overline{Y}$
10. $\overline{X \cap Y} = \overline{X} \cup \overline{Y}$

– законы де Моргана.

11. $X \cup \overline{X} = U$
12. $X \cap \overline{X} = \varnothing$

– дополнимость.

13. $\overline{\overline{X}} = X$ – закон двойного дополнения.

$$\left.\begin{array}{l} 14.\ X \bigcup \varnothing = X \\ 15.\ X \bigcup U = U \\ 16.\ X \bigcap \varnothing = \varnothing \\ 17.\ X \bigcap U = X \end{array}\right\} - \text{существование универсальных границ.}$$

$$\left.\begin{array}{l} 18.\ \bar{U} = \varnothing \\ 19.\ \bar{\varnothing} = U \end{array}\right\} - \text{законы дополнения.}$$

Приведенная система тождеств является полной в том смысле, что любое соотношение между множествами является следствием этих тождеств: Справедливость равенств 1 – 19 можно установить, используя *принцип равнообъёмности*, согласно которому нужно доказать, что множества, стоящие в левой и правой частях равенства состоят из одних и тех же элементов. В качестве примера приведем доказательство равенства 9. Остальные тождества доказываются аналогично. Имеем:

$x \in \overline{X \bigcup Y} \Rightarrow x \notin X \bigcup Y \Rightarrow x \notin X$ и $x \notin Y \Rightarrow x \in \bar{X}$ и $x \in \bar{Y} \Rightarrow x \in \bar{X} \bigcap \bar{Y}$;

$x \in \bar{X} \bigcap \bar{Y} \Rightarrow x \in \bar{X}$ и $x \in \bar{Y} \Rightarrow x \notin X$ и $x \notin Y \Rightarrow x \notin X \bigcup Y \Rightarrow x \in \overline{X \bigcup Y}$.

1.5. Прямое произведение множеств

Прямым произведением множеств X и Y называют множество $X \times Y$, элементами которого являются всевозможные упорядоченные пары (x, y), такие, что $x \in X$, $y \in Y$:

$$X \times Y = \{(x, y) \mid x \in X,\ y \in Y\}.$$

Эта операция над множествами, в отличие от рассмотренных ранее, изменяет природу элементов: в новом множестве элементами являются пары.

Прямое произведение в общем случае не обладает свойствами коммутативности и ассоциативности: $X \times Y \neq Y \times X$, $(X \times Y) \times Z \neq X \times (Y \times Z)$.

Пусть теперь даны n множеств: $X_1, X_1, ..., X_n$. Упорядоченный набор из n элементов, таких, что $x_1 \in X_1$, $x_2 \in X_2, \ldots,$ $x_n \in X_n$, называется *вектором* или *кортежем*. Множество таких

векторов представляет собой прямое произведение множеств $X_1, X_1, ..., X_n$:

$$X_1 \times X_2 ... \times X_n = \{(x_1, x_2, ..., x_n) \mid x_1 \in A_1, x_2 \in X_2, ..., x_n \in X_n\}.$$

Если $X_1 = X_2 = ... = X_n = X$, то множество $X_1 \times X_2 \times ... \times X_n$ называется *степенью* (прямой) множества X и обозначается через X^n.

Проекцией вектора $v = (x_1, x_2, ..., x_n)$ на i-ю ось (обозначение $\text{пр}_i v$) называется его i-я компонента. Проекцией вектора v на оси с номерами $i_1, i_2, ..., i_k$ называется вектор $(x_{i_1}, x_{i_2}, ..., x_{i_k})$ длины k (обозначение $\text{пр}_{i_1, i_2, ..., i_k} v$).

Пусть V – множество векторов одинаковой длины. Тогда проекцией множества V на i-ю ось называется множество проекций всех векторов на i-ю ось: $\text{пр}_i V = \{\text{пр}_i v \mid v \in V\}$. Аналогично определяется проекция множества V на несколько осей: $\text{пр}_{i_1, ..., i_k} V = \{\text{пр}_{i_1, ..., i_k} v \mid v \in V\}$.

Пример 1.1. Пусть $X = Y = \square$.Тогда прямым произведением $X \times Y = \square^2$ является множество точек плоскости, то есть пар вида (x, y), где $x, y \in R$ и являются координатами точек плоскости.

Координатное представление точек плоскости, предложенное французским математиком и философом Р. Декартом, является исторически первым примером прямого произведения. Потому прямое произведение называют также *декартовым*.

Пример 1.2. Пусть $X = \{a, b, c, d, e, f, g, h\}$, $Y = \{1, 2, ..., 8\}$. Тогда $X \times Y$ – множество, содержащее обозначения всех 64 клеток шахматной доски.

1.6. Соответствия, отображения, отношения

Соответствия. *Соответствием* между множествами X и Y называется подмножество $G \subseteq X \times Y$. Если $(x, y) \in G$, то говорят, что y соответствует x при соответствии G. Мно-

жество $\text{пр}_1 G \subseteq X$ называется *областью определения*, а множество $\text{пр}_2 G \subseteq Y$ – *областью значений* соответствия. Если $\text{пр}_1 G = X$, то соответствие называется *всюду определенным* или *полностью определенным* (в противном случае – *частичным*); если $\text{пр}_2 G = Y$, то соответствие называется *сюръективным* или *сюръекцией*.

Множество всех $y \in Y$, соответствующих элементу $x \in X$, называется *образом* x в Y при соответствии G. Множество всех $x \in X$, которым соответствует элемент $y \in Y$, называется *прообразом* y в X при соответствии G.

Соответствие G называется *инъективным* или *инъекцией*, если прообразом любого элемента из $\text{пр}_2 G$ является единственный элемент из $\text{пр}_1 G = X$. Соответствие G называется *функциональным* или *однозначным*, если образом любого элемента из $\text{пр}_1 G$ является единственный элемент из $\text{пр}_2 G$. Соответствие G называется *взаимно однозначным* или *биекцией*, если оно всюду определено, сюръективно, функционально и инъективно.

О т о б р а ж е н и я. *Отображением* f множества X во множество Y называется функциональное соответствие (обозначение $f : X \to Y$). Множество X называется *областью определения* отображения, элемент $x \in X$ – *аргументом* отображения, элемент $f(x) \in Y$ – образом x при отображении f. При этом пишут $x \to f(x)$. Часто, когда множества X, Y – числовые, отображение называют *функцией*. Если числовое только множество Y, то отображение называют *функционалом*.

Образом подмножества $A \subseteq X$ при отображении f называется множество

$$f(A) = \{ f(x) \mid x \in A \}$$

Прообразом подмножества $B \subseteq Y$ при отображении f называется множество

$$f^{-1}(B) = \{ x \in A \mid f(x) \in B \}.$$

По аналогии с соответствиями различают сюръективные, инъективные и биективные отображения.

Пример 1.3. Обозначим через $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x \geq 0\}$. Рассмотрим следующие три отображения

$$f: \mathbb{R} \to \mathbb{R}^+;\ g: \mathbb{R}^+ \to \mathbb{R};\ h: \mathbb{R}^+ \to \mathbb{R}^+,$$

которые зададим одной формулой: $f(x) = x^2;\ g(x) = x^2;\ h(x) = x^2$. Они различны, так как различны исходные множества. При этом f является сюръективным, но не инъективным; g – инъективно, но не сюръективно; h – биективно.

Отображения вида $X \to X$ называются *преобразованиями* множества X. Преобразование e_X называется *тождественным*, если $\forall x \in X:\ e_X(x) = x$.

Пусть $f: X \to Y$ и $g: Y \to Z$ – некоторые отображения. *Суперпозицией* этих отображений называется отображение $gf: X \to Z$, определяемое следующим образом:

$$(gf)(x) = g(f(x)),\ x \in X.$$

Отметим, что суперпозиция определена не для любых пар отображений. Однако суперпозиция двух преобразований одного и того же множества определена всегда.

Операция суперпозиции ассоциативна: $(hg)f = h(gf)$, где $f: X \to Y$, $g: Y \to Z$, $h: Z \to V$ – отображения.

Пусть $f: X \to Y$ и $g: Y \to X$. Отображение g называется *обратным* к отображению f (а отображение f *обратным* к g), если

$$fg = e_Y;\ gf = e_X.$$

Обратное отображение обозначается f^{-1}. Если обратное отображение существует, то оно единственно. Необходимое и достаточное условие существования обратного отображения дает следующая теорема.

Теорема 1.1. *Отображение f имеет обратное тогда и только тогда, когда оно биективно.*

Доказательство. Пусть $f: X \to Y$.

Необходимость. Пусть существует обратное отображение $f^{-1} = g: Y \to X$. Рассмотрим $\forall y \in Y$ и $x = g(y)$. Тогда $f(x) = f(g(y)) = y$, где x – прообраз y при отображении f. Таким образом $\forall y \in Y$ имеет прообраз x, т. е. f сюръективно.

Далее, если $x, x_1 \in X$, причем $f(x) = f(x_1)$, то $g(f(x)) = g(f(x_1))$. Следовательно, $e_X(x) = e_X(x_1)$, т. е. $x = x_1$ и f инъективно. Отсюда f биективно, и необходимость доказана.

Достаточность. Пусть f биективно. Определим отображение $g: Y \to X$ следующим образом. Положим $g(y) = x$, если $f(x) = y$. В силу биективности f отображение g определено на всем Y, и $g = f^{-1}$.▪

О т н о ш е н и я. *Бинарным отношением* α на множестве X называется подмножество $\alpha \subseteq X^2$. Тот факт, что $x \in X$ находится в отношении α с $y \in X$, обозначается следующим образом: $x \alpha y$

Областью определения бинарного отношения α на множестве X называется множество

$$\delta_\alpha = \{x \mid \exists y : x \alpha y\},$$

а областью значений – множество

$$\gamma_\alpha = \{y \mid \exists x : y \alpha x\}.$$

П р и м е р 1.4. Примеры отношений:

– отношение равенства «=» на множестве X состоит из всех пар вида (x, x), $x \in X$. Если элемент x находится в отношении равенства к элементу y, то пишут $x = y$;

– отношение неравенства «<» на множестве $\mathbb{R}$: $\{(x, y) \in \mathbb{R}^2 \mid x < y\}$;

– отношение делимости «|»на множестве $\mathbb{Z}$: $\{(x, y) \in \mathbb{Z}^2 \mid \exists c \in Z : x = yc\}$.

Так как отношения определяются как подмножества, то над ними можно производить теоретико-множественные операции. Например, если α – отношение «=», то $\bar{\alpha} =$«$\neq$», а $\alpha + \bar{\alpha} =$«$\leq$».

Дополнением бинарного отношения α на множестве X считается множество

$$\bar{\alpha} = X^2 \setminus \alpha .$$

Обратным отношением (*обращением*) для бинарного отношения α называется множество

$$\alpha^{-1} = \{(x, y) \mid (y, x) \in \alpha\} .$$

Произведением отношений α и β называется отношение

$$\alpha\beta = \{(x, y) \mid \exists c : (x, c) \in \alpha \wedge (c, y) \in \beta\} .$$

Всякое подмножество $\rho \subseteq X^n$ называют *n-местным отношением* на множестве X.

Совокупность всех отношений на множестве X, для которых заданы операции суммы, произведения, разности, дополнения и обращения, образуют *алгебру отношений* (*исчисление отношений*) множества X. В частности, последняя находит применение при разработке реляционных баз данных.

Свойства отношений. Отношения делятся на различные виды в зависимости от того, обладают или не обладают некоторыми свойствами.

1. *Рефлексивность*: $\forall x \in X : x \alpha y$. Например, рефлексивно на множестве прямых отношение «прямая x пересекает прямую x».

2. *Симметричность*: $\forall x, y \in X : x \alpha y \Rightarrow y \alpha x$. Например, симметрично отношение параллельности на множестве прямых плоскости.

3. *Транзитивность*: $\forall x, \text{y}, \text{z} \in X : x \alpha y \wedge y \alpha z \Rightarrow x \alpha z$. Например, транзитивно на множестве отрезков отношение «отрезок x длиннее отрезка y».

Среди различных бинарных отношений выделяются два специальных типа, играющих важную роль в разнообразных математических конструкциях и доказательствах.

Отношение эквивалентности. Отношение α на множестве X называется *отношением эквивалентности*, если оно обладает свойствами рефлексивности, симметричности и транзитивности. Отношение эквивалентности $x\alpha y$ часто обозначают: $x \sim y$.

Пример 1.5. Примеры отношения эквивалентности:

– отношение «одного роста» на множестве X людей;

– отношение подобия на множестве треугольников;

– отношение принадлежности двух студентов к одной студенческой группе.

Смежным классом (*классом эквивалентности*) элемента x по эквивалентности α называется множество

$$[x]_\alpha = x/\alpha = \{y \mid x\alpha y\}.$$

Любой элемент $y \in [x]_\alpha$ называется *представителем* этого класса.

Множество классов эквивалентности элементов множества X по эквивалентности α называется *фактор-множеством* X *по* α и обозначается X/α.

С каждым отношением эквивалентности связано разбиение множества на непересекающиеся подмножества, которое лежит в основе всевозможных классификаций.

Разбиением множества X называется всякое представление этого множества в виде суммы непересекающихся подмножеств:

$$X = \bigcup_{i \in I} X_i,\ i \neq j \Rightarrow X_i \cap X_j = \varnothing.$$

Здесь I – множество индексов, которое может быть конечным, счетным или несчетным. Множества X_i называют *слоями* разбиения.

Имеет место следующая теорема.

Теорема 1.2. *Для того чтобы отношение* α *позволяло разбить множество* X *на классы, необходимо и достаточно, чтобы* α *было отношением эквивалентности.*

Пример 1.6. Плоскость $\mathbb{R}^2$ разбита на прямые

$$\{(x, y) \mid x + y = c\},\ c \in \mathbb{R}.$$

Этому разбиению соответствует отношение α такое, что $(x_1, y_1)\alpha(x_2, y_2)$ если $x_1 + y_1 = x_2 + y_2$.

Покажем, что каждая эквивалентность $\alpha \subseteq X^2$ отвечает некоторому разбиению множества X.

Для каждого $x \in X$ обозначим через $[x]$ класс всех элементов, эквивалентных x:

$$[x] = \{y \mid y\alpha x\}.$$

Из рефлексивности α следует, что $x \in [x]$. Далее, если $y \in [x]$, то есть $y\alpha x$, то $\forall z \in [y] \Rightarrow z\alpha y$. Из транзитивности имеем, что $z\alpha x$, то есть $z \in [x]$. Таким образом, $[y] \subseteq [x]$. В силу симметричности отношения $y\alpha x \Rightarrow x\alpha y$, то есть $x \in [y]$. Повторяя рассуждения, получим, что $[x] \subseteq [y]$. Следовательно, $[x] = [y]$. Таким образом, каждый элемент $x \in X$ входит в некоторый класс $[x]$ и различные классы не пересекаются, то есть классы образуют разбиение множества X, отвечающее отношению эквивалентности α.

Приведем примеры использования отношения эквивалентности для образования математических понятий.

1. *Понятие вектора*. Сначала вводится понятие направленного отрезка, как пары точек (A, B). Два отрезка (A, B) и (C, D) объявляются эквивалентными, если середины отрезков (A, D) и $(C,$ B$)$ совпадают. Далее проверяется, что это отношение между направленными отрезками рефлексивно, симметрично и транзитивно. Класс эквивалентных отрезков и есть вектор.

2. *Построение рациональных чисел из целых*. Рассмотрим всевозможные пары (a, b) из целых чисел такие, что $b \neq 0$. Пары (a, b) и (c, d) объявляются эквивалентными, если $ad = bc$. Далее проверяется рефлексивность, симметричность и транзитивность. Класс эквивалентных пар – рациональное число.

Отношение порядка. Бинарное отношение β на множестве X называется *отношением порядка*, если оно рефлексивно, транзитивно и антисимметрично. Последнее свойство означает: $x\beta y \wedge y\beta x \Rightarrow x = y$.

Пример 1.7. Примеры отношений порядка:

– отношение «$\le$» на множестве действительных чисел. Отношение «$<$» порядком не является, так как оно не рефлексивно;

– отношение «$\subseteq$» на множестве подмножеств некоторого множества;

– на множестве двоичных слов длины n можно ввести отношение порядка следующим образом. Пусть $\tilde{\alpha} = \alpha_1\alpha_2...\alpha_n$ и $\tilde{\beta} = \beta_1\beta_2...\beta_n$ – двоичные слова. Положим $\tilde{\alpha} \le \tilde{\beta}$, если для $\forall i\ \tilde{\alpha}_i \le \tilde{\beta}_i$, $i = 1, 2, \ldots, n$.

Пусть β – отношение порядка на множестве X. Элементы $x, y \in X$ называются *сравнимыми*, если $x\beta y$ или $y\beta x$, в противном случае – *несравнимыми*.

Порядок называется *линейным*, если любые два элемента сравнимы. В противном случае говорят о *частичном* порядке. Множество A с заданным на нем порядком (частичным или линейным) называется *упорядоченным* (частично или линейно). Первое отношение в примере 1.7 задает линейный порядок, два других отношения порядка – частичные. Например, двоичные слова 011 и 110 несравнимы.

Элемент a частично упорядоченного множества X называется *максимальным* (*минимальным*), если из того, что $a \le x$ следует $a = x$. Элемент $a \in X$ называется *наибольшим* (*наименьшим*), если $x \le a$ ($a \le x$) для всех $x \in X$.

Верхней (нижней) гранью подмножества Y частично упорядоченного множества X называется такой элемент $x \in X$, что

$$\forall y \in Y \Rightarrow y \le x \ (\forall y \in Y \Rightarrow x \le y).$$

Точной в*ерхней (нижней) гранью подмножества* $Y \subseteq X$ называется наименьшая верхняя (наибольшая нижняя) грань для Y. Точная верхняя и точная нижняя грани обозначаются соответственно $\sup Y$ и $\inf Y$.

Линейный порядок на множестве X называется *полным*, если каждое непустое подмножество множества X имеет

наименьший элемент. В этом случае множество X называется *вполне упорядоченным*.

Частично упорядоченное множество X называется *решёткой*, или *структурой*, если для любых двух элементов $x, y \in X$ существует точная нижняя и точная верхняя грани.

Пример 1.8. Примеры решёток:

– множество всех подмножеств данного множества, упорядоченное по включению;

– всякое линейно упорядоченное множество; причем, если $x \le y$, то $\sup(x, y) = y$, $\inf(x, y) = x$.

2. ЭЛЕМЕНТЫ КОМБИНАТОРИКИ

Комбинаторным анализом, или короче *комбинаторикой*, называют часть дискретной математики, в которой изучаются перечислительные задачи теории конечных множеств. К типичным комбинаторным задачам относятся:

1) установление существования и подсчет числа подмножеств конечных множеств, обладающих заданными свойствами;

2) определение числа способов, которыми можно расположить элементы конечных множеств при заданных условиях на эти расположения и др.

Комбинаторика возникла в XVI веке в связи с проблемами азартных игр (игра в кости, карточные игры и лотереи). Одним из первых занялся подсчетом числа возможных комбинаций при игре в кости итальянский математик Тарталья. Первые теоретические исследования проблем комбинаторики были проведены в XVII веке французскими учеными Паскалем и Ферма. Дальнейшее развитие комбинаторики связано с именами Якова Бернулли, Лейбница и Эйлера. Тогда же сложилась и принятая в комбинаторике терминология (сочетания, размещения, перестановки и др.). К началу XX века комбинаторика считалась в основном завершенным разделом математики, лежащим вне основного русла развития математики и ее приложений. Но с середины XX века роль комбинаторики возросла в связи с развитием теории вычислительных систем и теории информации. В настоящее время комбинаторика является одним из наиболее интенсивно развивающихся разделов математики.

2.1. Общие правила комбинаторики

Правило суммы. Это правило позволяет найти число элементов в объединении двух конечных множеств X и Y:

$$|X \cup Y| = |X| + |Y| - |X \cap Y|. \qquad (2.1)$$

Если множества X и Y не пересекаются ($X \cap Y = \varnothing$), то

$$|X \cup Y| = |X| + |Y|. \qquad (2.2)$$

Несмотря на кажущуюся тривиальность этого правила, оно применяется в большинстве комбинаторных задач. Идея его применения состоит в том, что, если в исходной задаче прямой подсчет затруднителен, то нужно все изучаемые комбинации разбить на несколько классов, причем каждая комбинация должна входить только в один класс. В этом случае общее число комбинаций равно сумме чисел комбинаций во всех классах. Иногда правило суммы формулируют следующим образом:

если объект X можно выбрать m способами, а объект Y – n способами, то выбор «либо X, либо Y» можно осуществить $m+n$ способами.

Пример 2.1. Сколько имеется путей из вершины a в вершину b на решетке, показанной на рис. 2.1?

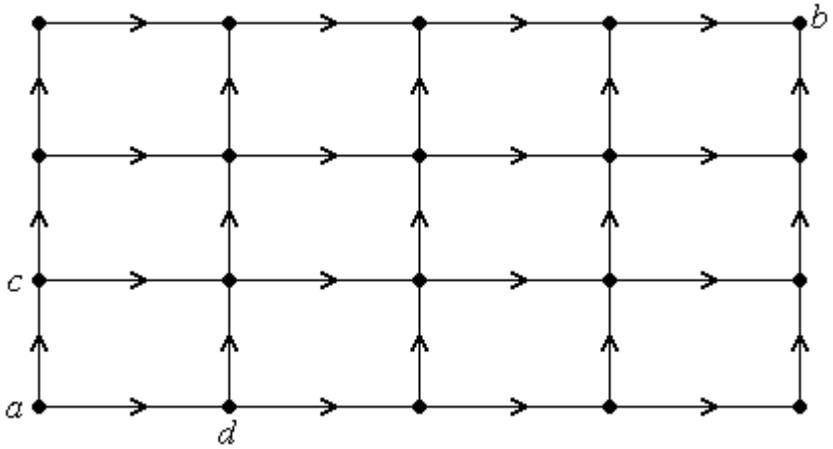


Рис. 2.1. Решетка размерности $[4\times5]$

Обозначим множество всех путей из a в b через L_{ab} и разобьем его на два непересекающиеся подмножества: L_{acb} – множество путей, проходящих через вершину c, L_{adb} – множество путей, проходящих через вершину d. Тогда $|L_{ab}|=|L_{acb}|+|L_{adb}|$.

Очевидно, что искомое количество путей зависит только от размеров решетки. Обозначим через $l_{m,\,n}$ количество путей в решетке размерности $[m \times n]$, где m, n – соответственно количество горизонтальных и вертикальных путей. Тогда $l_{m,\,n} = l_{m-1,\,n} + l_{m,\,n-1}$, причем $l_{m,\,n} = l_{n,\,m}$. Пользуясь этим соотношением для данного примера $(m = 4,\ n = 5)$, получим:

$$l_{4,\,5} = l_{3,\,5} + l_{4,\,4} = l_{2,\,5} + l_{3,\,4} + l_{4,\,3} = l_{2,\,5} + 3 \cdot l_{3,\,4} =$$

$$= l_{1,\,5} + l_{2,\,4} + 3 \cdot (l_{2,\,4} + l_{3,\,3}) = l_{1,\,5} + 4 \cdot l_{2,\,4} + 3 \cdot l_{3,\,3} =$$

$$= l_{1,\,5} + 4 \cdot (l_{1,\,4} + l_{2,\,3}) + 3 \cdot (l_{2,\,3} + l_{3,\,2}) = l_{1,\,5} + 4 \cdot l_{1,\,4} + 10 \cdot l_{2,\,3} =$$

$$= l_{1,\,5} + 4 \cdot l_{1,\,4} + 10 \cdot (l_{1,\,3} + l_{2,\,2}) = l_{1,\,5} + 4 \cdot l_{1,\,4} + 10 \cdot l_{1,\,3} + 10 \cdot (l_{1,\,2} + l_{2,\,1}) =$$

$$= l_{1,\,5} + 4 \cdot l_{1,\,4} + 10 \cdot l_{1,\,3} + 20 \cdot l_{1,\,2} = 1 + 4 \cdot 1 + 10 \cdot 1 + 20 \cdot 1 = 35.$$

Правило произведения. Это правило позволяет подсчитать число кортежей, которые можно составить из элементов данных конечных множеств. Для двух множеств X и Y это правило записывается в виде

$$|X \times Y| = |X||Y|. \qquad (2.3)$$

Из равенства (2.3) следует, что число упорядоченных пар, которые можно составить из элементов множеств X и Y равно произведению числа элементов в этих множествах.

Это правило можно сформулировать иначе:

если объект X можно выбрать m способами и после каждого из таких выборов другой объект Y можно выбрать n способами, то выбор «X и Y» можно осуществить mn способами.

По индукции правило умножения можно распространить на любое число сомножителей в декартовом произведении:

$$|X_1 \times ... \times X_n| = |X_1| ... |X_n|. \qquad (2.4)$$

Наиболее часто последнее равенство применяется, когда $X_1 = ... = X_n = X$:

$$|X^n| = |X|^n. \qquad (2.5)$$

В этом случае множество X называют алфавитом, его элементы – буквами, а элементы декартова произведения X^n – словами в алфавите X. Слова записывают как в обычном языке, то есть без разделения букв запятыми и без внешних скобок: $x_1x_2...x_n$. Число n называют длиной слова.

Пусть $|X| = m$. Тогда правило (2.5) можно сформулировать следующим образом:

число слов длины n в алфавите из m букв равно m^n.

Пример 2.2. Из 80 студентов 40 играют в футбол, а 50 – в волейбол, причем 27 студентов играют и в футбол и в волейбол. Сколько студентов играют хотя бы в одну из этих игр? Сколько студентов играют лишь в одну из этих игр? Сколько студентов не играют ни в одну из этих игр?

Пусть X – множество студентов, играющих в футбол, Y – множество студентов, играющих в волейбол. Тогда $|X| = 40$, $|Y| = 50$, $|A \cap B| = 27$.

Число студентов, играющих хотя бы в одну из этих игр, согласно формуле (2.1): $|X \cup Y| = 40 + 50 - 27 = 63$. Число студентов, играющих только в футбол: $|X| - |X \cap Y|$, а только в волейбол: $|Y| - |X \cap Y|$. Число студентов, играющих только в одну из этих игр: $|X| + |Y| - 2 \cdot |X \cap Y| = 36$. Число студентов, не играющих ни в одну из этих игр: $|\bar{X} \cap \bar{Y}| = 80 - |X \cap Y| = 17$.

Пример 2.3. Сколько существует 6-значных телефонных номеров?

Алфавит состоит из 10 цифр, номер – слово длины 6 в этом алфавите. Поэтому количество номеров равно 10^6.

Пример 2.4. Найти число слов, содержащих 4 буквы, в которых любые две соседние буквы различны (число букв в алфавите равно 33).

Первую букву можно выбрать 33-мя способами, вторую, третью и четвертую – 32 способами. Число слов равно $33 \cdot 32^3 = 1081344$.

Правило биекции. Это правило, которое называется также *принципом взаимно однозначного соответствия*, формулируется следующим образом:

если между множествами X *и* Y *можно установить взаимно однозначное соответствие (биекцию), то* $|X| = |Y|$.

В качестве примера применения этого принципа найдем мощность множества всех подмножеств данного множества X. Такое множество называют *булеаном* множества X и обозначают символом $B(X)$.

Пусть X – n-множество. Так как мощность множества не зависит от природы его элементов, то можно принять $X = \{1,\ 2,\ ...,\ n\}$.

Поставим в соответствие произвольному подмножеству $Y \subseteq X$ двоичное слово $\alpha_1\alpha_2...\alpha_n$ по следующему правилу:

$$\alpha_i = \begin{cases} 1, \text{ если } i \in Y; \\ 0, \text{ если } i \notin Y; \end{cases} \quad i = 1,\ 2,\ ...,\ n.$$

Это соответствие взаимно однозначное. Отсюда следует, что число всех подмножеств n-множества равно числу двоичных слов длины n, то есть $|B(X)| = 2^n$.

2.2. Размещения, перестановки, сочетания. Свойства чисел C_n^k

Набор элементов x_{i_1}, ..., x_{i_k} из множества $X = \{x_1,\ ...,\ x_n\}$ называется *выборкой* объема k из n элементов или (n, k)-*выборкой*. Выборка называется *упорядоченной*, если порядок следования элементов в ней задан. Две упорядоченные выборки, различающиеся лишь порядком следования элементов, считаются *различными*. Если порядок следования элементов не является существенным, то выборка называется *неупорядоченной*. В выборках могут допускаться или не допускаться повторения элементов. В зависимости от способа формирования все выборки в

комбинаторике классифицируют как размещения, перестановки и сочетания с повторениями и без повторений элементов.

k-размещением с повторениями из n элементов называется упорядоченная (n, k)-выборка, в которой элементы могут повторяться. Число всех таких выборок, которые отличаются друг от друга составом элементов или их порядком, равно числу векторов в декартовом произведении X^k. Это число обозначают $\overline{A}_n^k$ (от французского слова arrangement – размещение). По правилу произведения получаем

$$\overline{A}_n^k = |A|^k = n^k. \qquad (2.6)$$

Пример 2.5. Для запирания сейфа используется диск, на который нанесены 12 символов, а секретное слово состоит из 5 символов. Сколько неудачных попыток может быть сделано человеком, не знающим секретного слова?

Решение. Общее число комбинаций равно $\overline{A}_{12}^5 = 12^5 = 248832$. Значит, неудачных попыток может быть 248831.

k-размещением без повторений из n элементов называется упорядоченная (n, k)-выборка, в которой элементы не повторяются. Число всех упорядоченных k-множеств с различными элементами, которые можно составить из элементов n-множества X, обозначают A_n^k.

Для вычисления A_n^k необходимо сделать k выборов. 1-й элемент можно выбрать n способами, 2-й – $(n-1)$ способами и т. д. Последний k-й элемент можно выбрать $(n-k+1)$ способами. По правилу произведения

$$A_n^k = n(n-1)...(n-k+1) = \frac{n!}{(n-k)!}. \qquad (2.7)$$

Здесь $n! = 1 \cdot 2 \cdot ... \cdot (n-1) \cdot n$ – «n-факториал» (0!=1!=1).

Пример 2.6. Научное общество состоит из 25 человек. Надо выбрать президента общества, вице-президента, ученого секретаря и казначея. Сколькими способами может быть сделан

этот выбор, если каждый член общества может занимать лишь один пост?

Ответ: $A_{25}^4 = 25 \cdot 24 \cdot 23 \cdot 22 = 303600$.

Решим следующую комбинаторную задачу: сколькими способами можно упорядочить n-множество X ?

Перестановками без повторений называют различные упорядоченные n-множества, которые состоят из одних и тех же элементов, а отличаются друг от друга лишь порядком. Число таких перестановок обозначают P_n (от французского слова permutation – перестановка).

Формулу для P_n получаем из выражения (2.7) при $k = n$:

$$P_n = A_n^n = n!. \qquad (2.8)$$

Пример 2.7. Сколькими способами можно посадить на скамейку 9 человек?

Ответ: P_9 =9!=362880.

К *перестановкам с повторениями* приводит следующая задача.

Имеется n-множество X, состоящее из k различных элементов ($k \le n$). Сколько перестановок можно сделать из n_1 элементов первого типа, n_2 элементов второго типа,…, n_k элементов k-го типа?

Перестановки элементов каждого типа можно делать независимо друг от друга. Поэтому по правилу произведения элементы множества можно переставлять друг с другом $n_1!n_2!...n_k!$ способами так, что перестановки не изменятся. Тогда число различных перестановок с повторениями буде равно

$$P(n_1,\ n_2,\ ...,\ n_k) = \frac{n!}{n_1! \cdot n_2! \cdot ... \cdot n_k!}, \qquad (2.9)$$

где $n = n_1 + n_2 + ... + n_k$.

Пример 2.8. Сколько перестановок можно сделать из букв слова «Математика»?

В данном случае $n=10$, $n_1=2$ («М»), $n_2=3$ («а»), $n_3=2$ («т»), $n_4=1$ («е»), $n_5=1$ («и»), $n_6=1$ («к»). По формуле (2.9)

$$P(2,\ 3,\ 2,\ 1,\ 1,\ 1)=\frac{10!}{2!3!2!1!1!1!}=31920.$$

В тех случаях, когда не имеет значения порядок элементов в подмножестве некоторого множества, а лишь его состав, говорят о сочетаниях. К сочетаниям без повторений приводит следующая задача комбинаторики: сколько k-элементных подмножеств с различными элементами можно составить из элементов n-множества X?

Такие подмножества называют *сочетаниями без повторений из n элементов по k* или короче $(n,\ k)$-*сочетаниями*, а их число обозначают C_n^k (от французского слова combination – сочетание). Другими словами, сочетаниями без повторений называется неупорядоченная $(n,\ k)$-выборка, в которой элементы не повторяются. Для любого действительного n и целого неотрицательного k числа C_n^k называются *биномиальными коэффициентами*.

Формулу для числа сочетаний легко получить из формулы (2.7) для числа размещений. Выберем какое-нибудь k-элементное подмножество $Y\subseteq X$. Его можно упорядочить $k!$ способами, а число таких подмножеств есть C_n^k. Тогда справедлива формула

$$A_n^k=k!C_n^k, \tag{2.10}$$

откуда

$$C_n^k=\frac{A_n^k}{k!}=\frac{n!}{(n-k)!k!}. \tag{2.11}$$

Пример 2.9. Сколько всего партий играется в шахматном турнире с n участниками?

Ответ: C_n^2, так как каждая партия однозначно определяется двумя ее участниками.

Пусть множество X состоит из элементов n различных типов: $\{x_1,..., x_n\}$. Множество M, составленное из $x_1,..., x_n$, в котором элементы могут повторяться, называется *мультимножеством*. Для задания мультимножества надо указать число вхождений в него каждого элемента:

$$M = \begin{pmatrix} x_1 \ ... \ x_n \\ k_1 \ ... \ k_n \end{pmatrix},\ k_i \geq 0,\ i = 1,...,\ n,$$

где $k_1 + ... + k_n = k$ – мощность мультимножества.

Число мультимножеств мощности k, составленных из элементов n-множества X, называют *сочетаниями с повторениями* и обозначают $\overline{C}_n^k$. Другими словами, сочетаниями с повторениями называется неупорядоченная $(n,\ k)$-выборка, в которой элементы могут повторяться.

Зашифруем каждую комбинацию из k элементов с помощью нулей и единиц: для каждого типа напишем столько единиц, сколько элементов этого типа входит в комбинацию, а различные типы отделим друг от друга нулями. Если элементы какого-нибудь типа не вошли в комбинацию, то надо писать два или большее число нулей. При этом получим k единиц и $n-1$ нулей. Тогда число $\overline{C}_n^k$ будет равно числу перестановок с повторениями из k единиц и $n-1$ нулей: $\overline{C}_n^k = P(k,\ n-1)$. Так как $P(k,\ n-1) = \dfrac{(n+k-1)!}{k!(n-1)!}$, то

$$\overline{C}_n^k = C_{n+k-1}^k. \qquad (2.12)$$

Встречаются задачи, в которых на сочетания с повторениями налагается дополнительное условие – в них обязательно должны входить элементы r фиксированных типов $(r \leq m)$. В этом случае $\overline{C}_m^{k-r} = C_{m+k-r-1}^{k-r}$. В частности, если $m \leq k$ и требуется, чтобы в k-подмножество с повторениями входил по крайней мере один элемент каждого из типов $(r = m)$, то получим C_{k-1}^{k-m} мультимножеств.

Пример 2.10. Завод выпускает автомобили четырех марок. Сколькими способами можно купить 7 автомобилей?

Решение. Мощность мультимножества $k=7$, число различных элементов $n=4$. Число способов покупки автомобилей равно $\overline{C}_4^7 = C_{4+7-1}^7 = C_{10}^7 = \dfrac{10!}{7!3!} = 120$.

Числа C_n^k обладают целым рядом замечательных свойств, которые выражают различные соотношения между k-подмножествами n-множества X. Рассмотрим некоторые из них.

1) $\sum\limits_{k=0}^{n} C_n^k = 2^n$.

Это свойство непосредственно следует из определения чисел C_n^k: $\sum\limits_{k=0}^{n} C_n^k = \left|B(X)\right|$.

2) $C_n^k = C_n^{n-k}$.

Для доказательства поставим в соответствие подмножеству $Y \subseteq X$ его дополнение $\overline{Y} = X \setminus Y$. Это соответствие взаимнооднозначное. При этом, если $\left|Y\right| = k$, то $\left|\overline{Y}\right| = n-k$. Следовательно, по правилу биекции число подмножеств, составленных из k элементов столько же, сколько подмножеств, составленных из $n-k$ элементов.

Формально это свойство вытекает также из формулы (2.11).

3) $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$.

Для доказательства разобьем все k-элементные подмножества множества X на два класса:

а) подмножества, не содержащие элемент n. Это будут k-элементные подмножества $(n-1)$-множества, поэтому число их равно C_{n-1}^k;

б) подмножества, содержащие элемент n. Если из каждого такого подмножества удалить элемент n, то получим $(k-1)$-элементные подмножества $(n-1)$-множества, число которых

равно C_{n-1}^{k-1}. Свойство 3) вытекает, таким образом, из правила сложения.

Это свойство можно доказать также формально, используя формулу (2.11).

При $k=0$ имеем $C_n^0 = C_{n-1}^{-1} + C_{n-1}^0$. Так как $C_n^0 = C_{n-1}^0 = 1$, то полагают $C_{n-1}^{-1} = 0$. Кроме того, придерживаются соглашения $C_n^k = 0$ при $k > n$. Тогда свойство 3) позволяет последовательно вычислить числа C_n^k при $n = 0, 1, 2, \ldots$. Вычисления представляют в виде *треугольника Паскаля* по имени французского математика Б. Паскаля, в трудах которого он применяется. Это название исторически неточно, так как такую таблицу знал уже арабский математик Омар Хайям (XIII в.). Этот треугольник имеет вид:

$n=0$				1				
$n=1$			1		1			
$n=2$		1		2		1		
$n=3$	1		3		3		1	
$n=4$	1	4		6		4	1	
$n=5$	1	5	10		10	5	1	
$n=6$	1	6	15	20		15	6	1

……..……..

2.3. Метод включений и исключений

Поставим задачу подсчитать количество элементов в объединении конечных множеств $X_1, X_2, \ldots, X_m$, которые могут иметь непустые пересечения между собой, т. е. это объединение в общем случае не является разбиением.

Для двух множеств имеем формулу (2.1). Обозначим $X_1 \cup X_2 = X$ и применим эту формулу для трех множеств, используя дистрибутивность операций объединения и пересечения множеств:

$$|X_1 \cup X_2 \cup X_3| = |X \cup X_3| = |X| + |X_3| - |X \cap X_3| =$$
$$= |X_1 \cup X_2| + |X_3| - |X_1 \cap X_3 \cup X_2 \cap X_3| = \qquad (2.13)$$
$$= |X_1| + |X_2| + |X_3| - |X_1 X_2| - |X_1 X_3| - |X_2 X_3| + |X_1 X_2 X_3|.$$

В результате по индукции получаем *формулу включений и исключений*:

$$\left|\bigcup_{i=1}^{m} X_i\right| = \sum_{i=1}^{m} |X_i| - \sum_{1 \le i < j \le m} |X_i \cap X_j| + \sum_{1 \le i < j < k \le m} |X_i \cap X_j \cap X_k| - \ldots +$$
$$+ (-1)^{k+1} \sum_{1 \le i_1 < i_2 < \ldots < i_k \le m} \left|\bigcap_{j=1}^{k} X_{i_j}\right| + \ldots + (-1)^{m+1} \left|\bigcap_{i=1}^{m} X_i\right|. \qquad (2.14)$$

Название этой формулы подчеркивает использование последовательных включений и исключений элементов подмножеств.

Часто формулу (2.14) записывают в другом виде. Рассмотрим некоторое N-множество элементов Y и m-множество свойств $P = \{p_1, \ldots, p_m\}$, которыми элементы могут обладать или не обладать. Пусть подмножество $X_i \subseteq Y$ состоит из элементов, обладающих свойством p_i, $i = 1, \ldots, m$. Тогда подмножество $X = \left|\bigcup_{i=1}^{m} X_i\right|$ объединяет элементы из Y, которые обладают хотя бы одним из свойств множества P. Дополнение $\overline{X}$ составляют элементы, которые не обладают ни одним из свойств p_i, $i = 1, \ldots, m$. Пересечения вида $\bigcap_{j=1}^{k} X_{i_j}$ объединяют элементы, обладающие одновременно свойствами $p_{i_1}, \ldots, p_{i_k}$. Если обозначить число таких элементов через $N(p_{i_1}, \ldots, p_{i_k})$, то для числа элементов множества $\overline{X}$ имеем *формулу обращения*

$$N(\overline{p}_1, \ldots, \overline{p}_m) = N - \sum_{i=1}^{m} N(p_i) + \sum_{1 \le i < j \le m} N(p_i, p_j) - \ldots +$$
$$+ (-1)^k \sum_{1 \le i_1 < i_2 < \ldots < i_k \le m} N(p_{i_1}, \ldots, p_{i_k}) + \ldots + (-1)^m N(p_1, \ldots, p_m). \qquad (2.15)$$

Использование формулы (2.15) в комбинаторике называют *методом включений и исключений.* Как применение этого метода рассмотрим *задачу о беспорядках* или *задачу о встречах*, предложенную Монмором, которая представляет интерес для различных приложений. Сколькими способами можно разместить n элементов множества $A=\{a_1,\ a_2,\ ...,\ a_n\}$ в n ячеек множества $B=\{b_1,\ b_2,\ ...,\ b_n\}$ (по одному в каждой) так, чтобы никакой элемент a_i не попал в ячейку b_i?

Перестановку без повторений можно представить как биекцию множества $\{1,\ 2,\ ...,\ n\}$ на себя:

$$\pi:\{1,\ 2,\ ...,\ n\}\to\{1,\ 2,\ ...,\ n\},\ i\to\pi(i).$$

Перестановку π называют *беспорядком*, если в ней каждый элемент стоит не на своем месте, то есть $\pi(i)\neq i$, $i=1,\ 2,\ ...,\ n$.

Задача о беспорядках состоит в том, чтобы подсчитать число D_n перестановок-беспорядков. Например, $D_2=1$, $D_3=2$.

Обозначим через X_i множество все перестановок π, в которых элемент i стоит на i-ом месте:

$$X_i=\{\pi\mid\pi(i)=i\},\ i=1,\ 2,...,\ n.$$

Множество $\bigcup_{i=1}^{n}X_i$ состоит из перестановок, в которых хотя бы один элемент стоит на своем месте, а все остальные перестановки будут беспорядками. Тогда по формуле обращения

$$D_n=n!-\left|\bigcup_{i=1}^{n}X_i\right|. \qquad (2.16)$$

Подсчитаем число перестановок в каждом из множеств X_i и в их пересечениях. Если $\pi(i)=i$, то сужение отображения π на множество $\{1,\ 2,\ ...,\ n\}\setminus\{i\}$ является биекцией этого множества на себя. Следовательно, $|X_i|=(n-1)!$, $i=1,\ 2,\ ...,\ n$.

Множество $X_i\cap X_j$ составлено из перестановок π, в которых $\pi(i)=i$, $\pi(j)=j$. Сужение отображения π на множество $\{1,\ 2,\ ...,\ n\}\setminus\{i,\ j\}$ является биекцией этого множества на себя.

Следовательно, $\left|X_i \cap X_j\right| = (n-2)!,\ 1 \le i < j \le n$. По индукции находим, что $\left|\bigcap_{j=1}^{k} X_{i_j}\right| = (n-k)!,\ 1 \le i_1 < i_2 < \ldots < i_k \le n$. По формуле включений и исключений

$$\left|\bigcup_{i=1}^{n} X_i\right| = \sum_{i=1}^{n}(n-1)! - \sum_{i<j}(n-2)! + \ldots +$$

$$+(-1)^{k+1} \sum_{1 \le i_1 < i_2 < \ldots < i_k \le n} (n-k)! + \ldots + (-1)^{n+1} \cdot 1.$$

В каждой из этих сумм слагаемые не зависят от индекса суммирования, поэтому для вычисления надо знать только их количество. В первой сумме n слагаемых, число слагаемых во второй сумме равно количеству всевозможных пар $\{i, j\}$, которые можно составить из n индексов 1, 2, ..., n, то есть C_n^2. Аналогично в k-ой сумме имеем C_n^k слагаемых. Отсюда

$$\left|\bigcup_{i=1}^{n} X_i\right| = n \cdot (n-1)! - C_n^2 \cdot (n-2)! + \ldots + (-1)^{k+1} \cdot C_n^k \cdot (n-k)! + \ldots + (-1)^{n+1}.$$

После подстановки в уравнение (2.16) и несложных преобразований находим

$$D_n = n! \cdot \left(1 - 1 + \frac{1}{2!} - \frac{1}{3!} + \ldots + \frac{(-1)^n}{n!}\right).$$

Выражение, стоящее в скобках, представляет собой частичную сумму ряда, сходящегося к e^{-1}, поэтому $D_n \approx n!/e$. Можно показать, что D_n равно целому числу, ближайшему к $n!/e$: $D_n = E(n!/e)$, где $E(x)$ – целая часть числа x.

2.4. Метод рекуррентных соотношений

2.4.1. Основные определения и примеры рекуррентных соотношений

Часто решение одной комбинаторной задачи удается свести к решению аналогичных задач меньшей размерности с помощью некоторого соотношения, называемого рекуррентным (от латинского слова *recurrere* – возвращаться). Тем самым решение сложной задачи можно получить, последовательно находя решение более легких задач, и далее, пересчитывая по рекуррентным соотношениям, находить решение трудной задачи.

Рекуррентным соотношением k-го порядка между элементами последовательности чисел $a_1, a_2, \ldots, a_n$ называется формула вида

$$a_n = F(a_{n-1}, a_{n-2}, \ldots, a_{n-k}),\ n > 1,\ k = 1, 2, \ldots, n-1. \qquad (2.17)$$

Частным решением рекуррентного соотношения является любая последовательность, обращающая соотношение (2.17) в тождество. Это соотношение имеет бесконечно много частных решений, так как первые k элементов последовательности можно задать произвольно. Например, последовательность $2, 4, 8, \ldots, 2^n, \ldots$ является решением рекуррентного соотношения $a_{n+2} = 3a_{n+1} - 2a_n$, так как имеет место тождество $2^{n+2} = 3 \cdot 2^{n+1} - 2 \cdot 2^n$.

Решение рекуррентного соотношения k-го порядка называется *общим*, если оно зависит от k произвольных постоянных $C_1, \ldots, C_k$, и путем подбора этих постоянных можно получить любое решение данного соотношения. Например, для соотношения

$$a_{n+2} = 5a_{n+1} - 6a_n \qquad (2.18)$$

общим решением будет

$$a_n = C_1 2^n + C_2 3^n. \qquad (2.19)$$

Действительно, легко проверить, что последовательность (2.19) обращает соотношение (2.18) в тождество. Поэтому надо

только показать, что любое решение соотношения (2.18) можно представить в виде (2.19). Но любое решение этого соотношения однозначно определяется значениями a_1 и a_2. Поэтому надо доказать, что для любых чисел a и b найдутся такие значения C_1 и C_2, что

$$\begin{cases} 2c_1 + 3c_2 = a, \\ 2^2 c_1 + 3^2 c_2 = b. \end{cases}$$

Так как эта система имеет решение при любых значениях a и b, то решение (2.19) действительно является общим решением соотношения (2.18).

Рассмотрим задачи, приводящие к рекуррентным соотношениям.

Числа Фибоначчи. В 1202 г. знаменитым итальянским математиком Леонардо из Пизы, который известен больше по своему прозвищу Фибоначчи (Fibonacci – сокращенное filius Bonacci, т. е. сын Боначчи) была написана книга «Liber abacci» («Книга об абаке»). До нас эта книга дошла во втором своем варианте, который относится к 1228 г. Рассмотрим одну из множества приведенных в этой книге задач.

Пара кроликов приносит раз в месяц приплод из двух крольчат (самки и самца), причем новорожденные крольчата через два месяца после рождения уже приносят приплод. Сколько кроликов появится через год, если в начале года была одна пара кроликов?

Из условия задачи следует, что через месяц будет две пары кроликов. Через два месяца приплод даст только первая пара кроликов, и получится 3 пары. А еще через месяц приплод дадут и исходная пара кроликов, и пара кроликов, появившаяся два месяца тому назад. Поэтому всего будет 5 пар кроликов и т. д.

Обозначим через $F(n)$ количество пар кроликов по истечении n месяцев с начала года. Тогда через $n+1$ месяцев будут эти $F(n)$ пар и еще столько новорожденных пар кроликов, сколько было в конце $(n-1)$-го месяца, то есть еще $F(n-1)$ пар. Таким образом имеет место рекуррентное соотношение

$$F(n+1) = F(n) + F(n-1). \qquad (2.20)$$

Так как $F(0) = 1$ и $F(1) = 2$, то последовательно находим: $F(2) = 3$, $F(3) = 5$, $F(4) = 8$ и т. д. Эти числа входят в последовательность чисел

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377,…,

которую называют *рядом Фибоначчи*, а его члены – *числами Фибоначчи*. Они обладают целым рядом замечательных свойств.

Числа Фибоначчи связаны со следующей комбинаторной задачей: *найти число двоичных слов длины* n, *в которых никакие две единицы не идут подряд.* Будем называть такие слова *правильными* и обозначим их число через a_n. Разобьем множество этих правильных слов на два класса: слова, оканчивающиеся на ноль, и слова, оканчивающиеся на единицу. Обозначим количество слов в этих классах $a_n^{(0)}$ и $a_n^{(1)}$ соответственно. По правилу сложения

$$a_n = a_n^{(0)} + a_n^{(1)} \qquad (2.21)$$

Очевидно, что у слова, оканчивающегося на ноль, первые $n-1$ символов образуют правильное слово длины $n-1$, или другими словами, имеется биекция между множеством правильных слов длины n, оканчивающихся на ноль, и множеством правильных слов длины $n-1$, то есть $a_n^{(0)} = a_{n-1}$.

Если правильное слово длины n оканчивается на единицу, то предыдущий символ этого слова должен быть нулем, а первые $n-2$ символа должны образовывать правильное слово длины $n-2$. Как и в предыдущем случае, снова имеем биекцию между множеством правильных слов длины n, оканчивающихся на единицу, и множеством правильных слов длины $n-2$. Следовательно. $n_n^{(1)} = a_{n-2}$. Из формулы (2.21) получаем рекуррентное соотношение

$$a_n = a_{n-1} + a_{n-2}. \qquad (2.22)$$

Первые два значения находятся непосредственно ($a_1 = 2$ – слова 0 и 1; $a_2 = 3$ – слова 000, 010, 101), а остальные – по формуле (2.22).

Определение числа расстановок скобок в неассоциативном произведении. Пусть "$\circ$" обозначает некоторую бинарную операцию. Рассмотрим выражение $a_1 \circ a_2 \circ ... \circ a_n$, в котором символ $\circ$ обозначает некоторую бинарную неассоциативную операцию. Сколько имеется различных способов расстановки скобок в этом выражении?

Как пример неассоциативной операции можно привести векторное произведение. Другой пример – обычное сложение и умножение, выполняемое на компьютере. В силу того, что представление каждого числа в памяти компьютера ограничено определенным количеством разрядов, при выполнении каждой операции возникает погрешность и суммарный результат этих погрешностей зависит от расстановки скобок. Пусть ε – *машинный ноль*. Это означает, что $1+\varepsilon = 1$. Тогда $(1+\varepsilon)+\varepsilon = 1$, в то время как $1+(\varepsilon+\varepsilon) = 1+2\cdot\varepsilon \neq 1$.

Обозначим число всевозможных способов расстановки скобок через D_n. Тогда

$$D_1 = D_2 = 1\colon\ (a_1),\ (a_1, a_2); \quad D_3 = 2\colon\ (a_1 \circ a_2) \circ a_3,\ a_1 \circ (a_2 \circ a_3);$$

$$D_4 = 5\colon ((a_1 \circ a_2) \circ a_3) \circ a_4,\ (a_1 \circ (a_2 \circ a_3)) \circ a_4,\ (a_1 \circ a_2) \circ (a_3 \circ a_4),$$

$$a_1 \circ ((a_2 \circ a_3) \circ a_4),\ a_1 \circ (a_2 \circ (a_3 \circ a_4)).$$

Назовем операцию $\circ$ условно произведением. Для произвольного n разобьем все способы расстановки скобок на классы, включив в k-ый класс способы, при которых сначала вычисляется произведение первых k и последних $n-k$ операндов с какой-то расстановок скобок, а потом вычисляется их произведение:

$$(a_1 \circ ... \circ a_k) \circ (a_{k+1} \circ ... \circ a_n) \qquad (2.23)$$

где $k = 1,\ 2,...,\ n-1$.

По определению количество способов расстановки скобок для вычисления первых k операндов равно D_k, последних – D_{n-k}. По правилу произведения число расстановок скобок для выражения (2.23) равно $D_k \cdot D_{n-k}$. По правилу сложения

$$D_n = \sum_{k=1}^{n-1} D_k D_{n-k}\,,\ n = 2,\ 3,... \qquad (2.24)$$

Например,

$$D_5 = D_1 \cdot D_4 + D_2 \cdot D_3 + D_3 \cdot D_2 + D_4 \cdot D_1 = 1 \cdot 5 + 1 \cdot 2 + 2 \cdot 1 + 5 \cdot 1 = 14\,.$$

2.4.2. Линейные рекуррентные соотношения с постоянными коэффициентами

Пусть функция F в соотношении (2.17) является линейной

$$a_n = \alpha_1 a_{n-1} + \alpha_2 a_{n-2} + ... + \alpha_{n-k} a_{n-k}\,,\ n = k,\ k+1,\ ...\,, \qquad (2.25)$$

где $\alpha_1,\ \alpha_2, ...,\ \alpha_{n-k}$ – некоторые числа. Такие соотношения называют *линейными соотношениями k-го порядка с постоянными коэффициентами.*

Сначала исследуем подробно соотношения второго порядка, а затем перейдем к общему случаю. При $k = 2$ из формулы (2.25) получим

$$a_n = \alpha_1 a_{n-1} + \alpha_2 a_{n-2}\,,\ n = 3,\ 4,...\,. \qquad (2.26)$$

Решение этих соотношений основано на следующих утверждениях.

Лемма 1. *Пусть* $(a_n) = (a_1,\ a_2,\ ...,\ a_n,\ ...)$ *– решение соотношения (2.26), а* c *– любое число. Тогда последовательность* $c \cdot (a_n) = (ca_1,\ ca_2,\ ...,\ ca_n,\ ...)$ *также является решением этого соотношения.*

Лемма 2. *Пусть* $(a_n) = (a_1,\ a_2,\ ...,\ a_n,\ ...)$ *и* $(b_n) = (b_1,\ b_2,\ ...,\ b_n,\ ...)$ *– решения соотношения (2.26). Тогда последовательность* $(a_n + b_n) = (a_1 + b_1,\ a_2 + b_2,\ ...,\ a_n + b_n, ...)$ *также является решением этого соотношения.*

Из этих двух простых лемм можно сделать следующий важный вывод. Совокупность всевозможных последовательностей $(a_n) = (a_1,\ a_2,\ ...,\ a_n,\ ...)$ с операциями покоординатного сложения и умножения на скаляр образует векторное пространство. Совокупность последовательностей, являющихся решени-

ями соотношения (2.26), представляет собой подпространство этого пространства. Объемлющее пространство всевозможных последовательностей бесконечномерно, но подпространство решений линейного рекуррентного соотношения имеет конечную размерность, равную порядку уравнения.

Л е м м а 3. *Размерность пространства решений рекуррентного соотношения (2.26) равна двум.*

Из леммы 3 следует, что для определения всех решений уравнения (2.26) необходимо отыскать два линейно независимых решения. Любое другое решение будет представляться линейной комбинацией этих базисных решений.

Рассмотрим рекуррентное соотношение первого порядка

$$a_n = \lambda a_{n-1}, \tag{2.27}$$

где λ – константа.

Если $a_1 = 1$, то из (2.27) имеем

$$a_n = \lambda^n, \tag{2.28}$$

то есть решением рекуррентного уравнения первого порядка является геометрическая прогрессия.

Будем искать решение рекуррентного соотношения второго порядка также в виде (2.28). Тогда, подставляя (2.28) в (2.26), получим

$$\lambda^n = \alpha_1 \lambda^{n-1} + \alpha_2 \lambda^{n-2}. \tag{2.29}$$

При λ=0 имеем нулевое решение, которое не представляет интереса. Считая $\lambda \neq 0$, поделим последнее соотношение на λ^{n-2}:

$$\lambda^2 = \alpha_1 \lambda + \alpha_2. \tag{2.30}$$

Таким образом, геометрическая прогрессия (2.28) является решением рекуррентного соотношения (2.26), если знаменатель прогрессии λ является корнем квадратного уравнения (2.30). Это уравнение называется *характеристическим уравнением* для рекуррентного соотношения (2.26).

Построение базисных решений зависит от корней λ_1 и λ_2 характеристического уравнения (2.30).

1) $\lambda_1 \neq \lambda_2$ ($\lambda_1 \neq 0,\ \lambda_2 \neq 0$). В этом случае имеем два решения λ_1^n и λ_2^n, которые линейно независимы. Чтобы убедиться в этом, покажем, что из формулы

$$a_n = c_1\lambda_1^n + c_2\lambda_2^n \tag{2.31}$$

путем соответствующего выбора констант можно получить любое решение соотношения (2.26). Рассмотрим произвольное решение $(a_n^*) = (a_1^*,\ a_2^*, ...,\ a_n^*, ...)$. Выберем константы c_1^* и c_2^* так, чтобы $a_n = a_n^*$ при $n = 1$ и $n = 2$:

$$\begin{cases} c_1^*\lambda_1^1 + c_2^*\lambda_2^1 = a_1^*, \\ c_1^*\lambda_1^2 + c_2^*\lambda_2^2 = a_2^*. \end{cases} \tag{2.32}$$

Определитель линейной системы (2.32)

$$\begin{vmatrix} \lambda_1 & \lambda_2 \\ \lambda_1^2 & \lambda_2^2 \end{vmatrix} = \lambda_1\lambda_2(\lambda_2 - \lambda_1) \neq 0,$$

следовательно, система имеет единственное решение, а значит формула (2.31) – общее решения соотношения (2.26).

2) $\lambda_1 = \lambda_2$. В случае кратных корней характеристическое уравнение (2.30) имеет вид $(\lambda - \lambda_1)^2 = 0$ или $\lambda^2 = 2\lambda_1\lambda - \lambda_1^2$. Тогда $\alpha_1 = 2\lambda_1$, $\alpha_2 = -\lambda_1^2$, а для соотношения (2.26) получим уравнение $a_n = 2\lambda_1 a_{n-1} - \lambda_1^2 a_{n-2}$, которое дает два базисных решения λ_1^n и $n\lambda_1^n$. Общее решение представляется в виде

$$a_n = c_1\lambda_1^n + c_2 n\lambda_1^n. \tag{2.33}$$

В случае соотношения k-го порядка (2.25) имеют место утверждения, аналогичные тем, которые были рассмотрены для уравнений 2-го порядка.

1) Совокупность всех решений уравнения (2.25) является подпространством в пространстве всех последовательностей.

2) Размерность этого пространства равна k, то есть каждое решение однозначно определяется своими первыми k значениями.

3) Для определения базиса подпространства решений составляется характеристическое уравнение

$$\lambda^k = \alpha_1\lambda^{k-1} + \alpha_2\lambda^{k-2} + ... + \alpha_k . \qquad (2.34)$$

Многочлен

$$H(x) = x^k - \alpha_1 \cdot x^{k-1} - \alpha_2 \cdot x^{k-2} - ... - \alpha_k \qquad (2.35)$$

называется *характеристическим многочленом* рекуррентного соотношения (2.25).

4) Если характеристическое уравнение имеет k различных корней $\lambda_1, \lambda_2, ..., \lambda_k$, то общее решение рекуррентного соотношения (9) имеет вид

$$a_n = c_1\lambda_1^n + c_2\lambda_2^n + ... + c_k\lambda_k^n . \qquad (2.36)$$

При заданных начальных значениях решения $a_i = a_i^*$, $i = 1, 2, ..., k$, константы c_i находятся из системы

$$c_1\lambda_1^i + c_2\lambda_2^i + ... + c_k\lambda_k^i = a_i^*, \ i = 1, 2, ..., k.$$

5) Если λ – корень характеристического уравнения кратности e, то соотношение (2.25) имеет следующие решения

$$\lambda^n, \ n\lambda^n, \ ..., \ n^{e-1}\lambda^n .$$

Пусть характеристическое уравнение (2.34) имеет корни: $\lambda_1, \ \lambda_2, \ldots, \ \lambda_r$ кратности соответственно $e_1, \ e_2, \ldots, \ e_r$, причем $e_1 + e_2 + ... + e_r = k$. Тогда характеристический многочлен и общее решение соотношения (2.25) представятся в виде

$$H(x) = (x - \lambda_1)^{e_1}(x - \lambda_2)^{e_2}...(x - \lambda_r)^{e_r},$$

$$a_n = \sum_{i=1}^{r} (c_1^{(i)} + c_2^{(i)} n + ... + c_{e_i} n^{e_i - 1})\lambda_i^n .$$

Ф о р м у л а Б и н е. Поставим задачу получить формулу в явном виде для чисел Фибоначчи. Для этого найдем решение рекуррентного соотношения (2.20) при условии, что $F_0 = F_1 = 1$. Составим характеристическое уравнение $\lambda^2 = \lambda + 1$, найдем его корни $\lambda_{1,\,2} = (1 \pm \sqrt{5})/2$ и получим общее решение $F_n = c_1\lambda_1^n + c_2\lambda_2^n$. Константы c_1 и c_2 определим из начальных условий: $c_1 = \dfrac{1}{\lambda_1 - \lambda_2}$, $c_2 = -\dfrac{1}{\lambda_1 - \lambda_2}$. Тогда $F_n = \dfrac{\lambda_1^n - \lambda_2^n}{\lambda_1 - \lambda_2}$ или

$$F_n = \frac{1}{\sqrt{5}} \cdot \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right] = \frac{\phi^n - (-\phi)^{-n}}{\phi - (-\phi)^{-1}}, \quad (2.37)$$

где $\phi = \frac{1+\sqrt{5}}{2} \approx 1,618$ – золотое сечение. Формула (2.37) называется *формулой Бине*. При этом $(-\phi)^{-1} = 1-\phi$. Из формулы Бине следует, что $F_n = E\left(\frac{\phi_n}{\sqrt{5}} \right)$.

2.5. Производящие функции

2.5.1. Определение и свойства производящей функции

Производящей функцией, или обычной производящей функцией, последовательности чисел $\{a_n\} = (a_0,\ a_1, ...,\ a_n, ...)$ называется формальный ряд

$$A(t) = \sum_{n=0}^{\infty} a_n t^n, \quad (2.38)$$

где t – формальная переменная. При этом будем писать $a_n = coeff_{t^n}\{A(t)\}$.

Пусть, например, $\{a_n\} = (1,\ 1, ...,\ 1, ...)$. Тогда

$$A(t) = 1 + t + t^2 + ... + t^n + ... = \sum_{n=0}^{\infty} t^n = \frac{1}{1-t},\ 1 = coeff_{t^n}\left\{ \frac{1}{1-t} \right\}.$$

Аналогично

$$\{a_n\} = (1,\ \lambda,\ \lambda^2, ...,\ \lambda^n, ...) \to A(t) = \sum_{n=0}^{\infty} \lambda^n t^n;\ \lambda^n = coeff_{t^n}\left\{ \frac{1}{1-\lambda t} \right\}.$$

Экспоненциальной производящей функцией последовательности $\{a_n\}$ называется ряд

$$E(t) = \sum_{n=0}^{\infty} a_n \frac{t^n}{n!}. \quad (2.39)$$

Для обычных производящих функций $A(t)$ вводится алгебра формальных степенных рядов, или алгебра Коши, с операциями сложения, умножения, суперпозиции, подстановки, дифференцирования и интегрирования. Алгебра степенных рядов $E(t)$, определяющих экспоненциальные производящие функции, известна как символическое исчисление Блиссара. Далее под производящей функцией будем понимать обычную производящую функцию $A(t)$.

Производящие функции позволяют установить различные свойства последовательностей $\{a_n\}$, в том числе связанные с комбинаторными задачами. Кроме того, с помощью производящих функций можно решать рекуррентные соотношения.

Рассмотрим свойства производящих функций.

1) *Линейной комбинации последовательностей взаимно однозначно соответствует линейная комбинация их производящих функций*:

$$\{c_n\} = \sum_{i=1}^{k} \alpha_i \{a_n^{(i)}\} \Leftrightarrow C(t) = \sum_{i=1}^{k} \alpha_i A^{(i)}(t),\ n = 0,\ 1,\ \ldots$$

2) *Дифференцирование производящей функции* $A(t)$:

$$A'(t) = \sum_{n=1}^{\infty} n a_n t^{n-1}.$$

Например, дифференцируя функцию $A(t) = \sum_{n=0}^{\infty} t^n = \frac{1}{1-t}$, получим

$$1 + 2t + 3t^2 + \ldots + nt^{n-1} + \ldots = \frac{1}{(1-t)^2},$$

то есть производящей функцией последовательности $(1,\ 2,\ \ldots,\ n,\ \ldots)$ является функция $\frac{1}{(1-t)^2}$: $n+1 = coeff_{t^n}\left\{\frac{1}{(1-t)^2}\right\}$.

Дифференцируя m раз функцию $A(t) = \sum_{n=0}^{\infty} t^n = \frac{1}{1-t}$, будем иметь

$$m!+[(m+1)m...2]t+...+[(m+n)(m+n-1)...\cdot n+1)]t^n+...=$$

$$=\frac{m!}{(1-t)^{m+1}} \qquad .(2.40)$$

После деления на $m!$ получим производящую функцию для сочетаний

$$1+C_{m+1}^m t+C_{m+2}^m t^2+...+C_{m+n}^m t^n+...=\frac{1}{(1-t)^{m+1}}. \qquad (2.41)$$

3) *Умножение производящей функции на* t соответствует сдвигу членов последовательности $\{a_n\}$ на одну позицию вправо. Если $A(t)\to(a_0,\ a_1,...,\ a_n,...)$, то $tA(t)\to(0,\ a_0,\ a_1,...,\ a_n,...)$.

Например, производящей функцией последовательности (0, 1, 2, …, n, …) является функция $\frac{t}{(1-t)^2}$: $n=coeff_{t^n}\left\{\frac{t}{(1-t)^2}\right\}$.

4) *Интегрирование производящей функции* $A(t)$:

$$\int_0^t A(z)dz=\sum_{n=0}^{\infty}\frac{a_n}{n+1}t^{n+1}.$$

В качестве примера найдем $\sum_{k=0}^{n}\frac{C_n^k}{k+1}$. Используем формулу бинома Ньютона

$$(t+a)^n=\sum_{k=0}^{n}C_n^k t^k a^{n-k}. \qquad (2.42)$$

При $a=1$:

$$\sum_{k=0}^{n}C_n^k t^k=(1+t)^n. \qquad (2.43)$$

Из равенства (2.43) следует, что функция $(1+t)^n$ является производящей для последовательности $(C_n^0,\ C_n^1,\ ...,\ C_n^k,\ ...,\ C_n^n,\ 0,\ 0,\ ...)$. Можно также написать

$$C_n^k=coeff_{t^k}\left\{(1+t)^n\right\}. \qquad (2.44)$$

Интегрируя левую часть соотношения (2.43), получим

$$\int_0^t \sum_{k=0}^{n} C_n^k z^k dz = \sum_{k=0}^{n} C_n^k \int_0^t z^k dz = \sum_{k=0}^{n} C_n^k \frac{t^k}{k+1}.$$

Для правой части имеем

$$\int_0^t (z+1)^n dz = \frac{(t+1)^{n+1}-1}{n+1}.$$

При $t=1$ находим

$$\sum_{k=0}^{n} \frac{C_n^k}{k+1} = \frac{2^{n+1}-1}{n+1}.$$

Название формулы (2.42) биномом Ньютона исторически неверно, так как эту формулу хорошо знали среднеазиатские математики Омар Хайям, Гиясэддин и др. В Западной Европе задолго до Ньютона ее знал Паскаль. Заслуга же Ньютона заключалась в том, что ему удалось обобщить формулу для $(t+a)^n$ на случай произвольного вещественного показателя степени $n=\alpha\in\square$, если в качестве биномиальных коэффициентов использовать числа

$$C_\alpha^k = \frac{\alpha\cdot(\alpha-1)\cdot\ldots\cdot(\alpha-k+1)}{k!}, \qquad (2.45)$$

причем вместо конечного числа слагаемых мы имеем бесконечный ряд:

$$(t+a)^\alpha = \sum_{k=0}^{\infty} a^{\alpha-k}\cdot C_\alpha^k \cdot t^k. \qquad (2.46)$$

Из формулы (2.46) многие производящие функции получаются как частные случаи. Во-первых, при $\alpha=n$ имеем формулу (2.41), так как $C_n^k=0$ при $k>n$. Во-вторых, при $a=1$, $\alpha=-(m+1)$ и замене t на $-t$ приходим к формуле (2.40).

5) *Производящая функция для свертки последовательностей. Сверткой* последовательностей $\{a_n\}=(a_0,\ a_1,\ldots,\ a_n,\ldots)$ и $\{b_n\}=(b_0,\ b_1,\ldots,\ b_n,\ldots)$ называется последовательность $\{c_n\}=(c_0,\ c_1,\ldots,\ c_n,\ldots)$, элементы которой вычисляются по пра-

вилу: $c_0 = a_0 b_0$, $c_1 = a_0 b_1 + a_1 b_0$, $c_2 = a_0 b_2 + a_1 b_1 + a_2 b_0$, …, $c_n = \sum_{i=0}^{n} a_i b_{n-i}$, …

Операция свертки является основной в цифровой обработке сигналов: после свертывания последовательности отсчетов сигнала со специально подобранной последовательностью происходит *фильтрация* – усиление одних частот и подавление других.

Свертка обозначается звездочкой: $\{c_n\} = \{a_n\} * \{b_n\}$.

Производящая функция свертки равна произведению производящих функций свертываемых последовательностей:

$$\{c_n\} = \{a_n\} * \{b_n\} \Rightarrow C(t) = A(t)B(t).$$

Действительно, при перемножении $A(t)$ и $B(t)$ n-ая степень переменной t складывается из всевозможных произведений $a_i t^i b_{n-i} t^{n-i}$, в которых первый сомножитель из $A(t)$, а второй из $B(t)$.

В качестве примера приведем *формулу Вандермонда*. Пусть

$$A(t) = (t+1)^m = \sum_{i=0}^{m} C_m^i t^i,\ B(t) = (t+1)^n = \sum_{j=0}^{n} C_n^j t^j.$$

По правилу свертки $C(t) = A(t)B(t) = (t+1)^{m+n} = \sum_{k=0}^{m+n} C_{m+n}^k t^k$.

С другой стороны,

$$C(t) = \left(\sum_{i=0}^{m} C_m^i t^i \right) \left(\sum_{j=0}^{n} C_n^j t^j \right) = \sum_{i=0}^{m} \sum_{j=0}^{n} C_m^i C_n^j t^{i+j} \overset{(i+j=k)}{=}$$

$$= \sum_{i=0}^{m} \sum_{k=i}^{m+n} C_m^i C_n^{k-i} t^k = \sum_{k=0}^{m+n} \sum_{i=0}^{k} C_m^i C_n^{k-i} t^k$$

Следовательно,

$$\sum_{i=0}^{k} C_m^i \cdot C_n^{k-i} = C_{m+n}^k. \qquad (2.47)$$

2.5.2. Решение рекуррентных соотношений методом производящих функций

Предварительно решим задачу по определению числа расстановок скобок в неассоциативном произведении, рассмотренную в п. 2.4.1.

Введем для последовательности $\{D_n\}$ производящую функцию: $D(t)=\sum_{n=1}^{\infty} D_n \cdot t^n$. Заменим коэффициенты D_n их выражениями из рекуррентного соотношения (2.24). Так как это соотношение имеет место, начиная с $n=2$, то первый член $D_1 t = t$ отделим от суммы:

$$D(t)=t+\sum_{n=2}^{\infty}\left(\sum_{k=1}^{n-1} D_k D_{n-k}\right)t^n .$$

Последовательность $\left(\sum_{k=1}^{n-1} D_k D_{n-k}\right)$ представляет собой свертку последовательности $\{D_n\}$ с собой. В силу свойства 5) для $D(t)$ имеем квадратное уравнение

$$D(t)=t+[D(t)]^2, \qquad (2.48)$$

решение которого имеет вид

$$D(t)=(1-\sqrt{1-4t})/2. \qquad (2.49)$$

Перед корнем выбран знак минус, так как $D(0)=0$. Чтобы найти D_n, надо разложить в ряд по степеням t правую часть уравнения (2.49). Для этого используем формулу бинома Ньютона (2.46) при $a=1$ и $\alpha=1/2$:

$$\sqrt{1-4\cdot t}=(1-4\cdot t)^{1/2}=\sum_{n=0}^{\infty} C_{1/2}^n \cdot (-4\cdot t)^n ,$$

где

$$C_{1/2}^n=\frac{\frac{1}{2}\left(-\frac{1}{2}\right)\left(-\frac{3}{2}\right)\ldots\left(\frac{1}{2}-n+1\right)}{n!}=(-1)^{n-1}\frac{1\cdot 3\cdot\ldots\cdot(2n-3)}{2^n n!}.$$

Умножим числитель и знаменатель последней дроби на произведение последовательных четных чисел от 2 до $2n-2$: $2\cdot 4\cdot ...\cdot (2n-2)=2^{n-1}(n-1)!$.

Тогда

$$C_{1/2}^{n}=(-1)^{n-1}\frac{(2n-2)!}{2^{n}n!2^{n-1}(n-1)!}=(-1)^{n-1}\frac{2}{n}4^{-n}\frac{(2n-2)!}{(n-1)!(n-1)!}=$$

$$=(-1)^{n-1}\frac{2}{n}4^{-n}C_{2n-2}^{n-1}.$$

Из формулы (2.49) получим

$$D(t)=[1-C_{1/2}^{0}-\sum_{n=1}^{\infty}(-1)^{n-1}\frac{2}{n}4^{-n}C_{2\cdot n-2}^{n-1}(-4t)^{n}]/2=\sum_{n=1}^{\infty}\frac{1}{n}C_{2n-2}^{n-1}t^{n}.$$

Таким образом, число расстановок скобок в неассоциативном произведении равно

$$D_{n}=coeff_{t^{n}}D(t)=\frac{1}{n}C_{2n-2}^{n-1}.$$

Рассмотрим теперь общий метод решения линейных рекуррентных соотношений с постоянными коэффициентами с помощью производящих функций. Пусть последовательность $\{a_n\}=(a_0,\ a_1,...,\ a_n,...)$ является решением соотношения (2.25).

Построим производящую функцию этой последовательности. Обозначим начальные отрезки этого ряда

$$A_j(t)=\sum_{n=0}^{j-1}a_n t^n,\ \ j=1,\ 2,\ ...,\ k,\ A_0(t)=0.$$

Заменим коэффициенты, начиная с k-го, по формуле (2.25)

$$A(t)=A_k(t)+\sum_{n=k}^{\infty}\left(\sum_{j=1}^{k}\alpha_j a_{n-j}\right)t^n=A_k(t)+\sum_{j=1}^{k}\alpha_j t^j\sum_{n=k}^{\infty}a_{n-j}t^{n-j}\ .\ (2.50)$$

Внутреннюю сумму представим в виде

$$\sum_{n=k}^{\infty}a_{n-j}t^{n-j}=A(t)-A_{k-j}(t)$$

и подставим в (2.50):

$$A(t) = A_k(t) + \sum_{j=1}^{k} \alpha_j t^j [A(t) - A_{k-j}(t)].$$

Из этого уравнения найдем производящую функцию $A(t)$:

$$A(t) = P(t) / Q(t), \qquad (2.51)$$

где $P(t) = A_k(t) - \sum_{j=1}^{k} \alpha_j t^j A_{k-j}(t)$; $Q(t) = 1 - \alpha_1 t - ... - \alpha_k t^k$.

Сравнивая $Q(t)$ с характеристическим многочленом

$$H(t) = t^k - \alpha_1 t^{k-1} - \alpha_2 t^{k-2} - ... - \alpha_k,$$

найдем

$$Q(t) = t^k H(1/t).$$

Если характеристический многочлен имеет корни λ_1, λ_2 ,…, λ_r кратности соответственно e_1, e_2 ,…, e_r, то

$$H(t) = (t - \lambda_1)^{e_1} (t - \lambda_2)^{e_2} ... (t - \lambda_r)^{e_r}.$$

Тогда

$$Q(t) = (1 - \lambda_1 t)^{e_1} (1 - \lambda_2 t)^{e_2} ... (1 - \lambda_r t)^{e_r}.$$

Раскладывая дробь (2.51) на простые, получим

$$A(t) = \sum_{i=1}^{r} \left(\frac{C_{i,e_i}}{(1 - \lambda_i t)^{e_i}} + ... + \frac{C_{i,1}}{1 - \lambda_i t} \right), \qquad (2.52)$$

где $C_{i,\,e_i}$, i=1, ..., r – константы.

Используя степенные ряды (2.40) для простых дробей, получим

$$\frac{1}{(1 - \lambda_i \cdot t)^{m+1}} = \sum_{n=0}^{\infty} C_{m+n}^{m} \cdot \lambda_i^n \cdot t^n. \qquad (2.53)$$

Подставляя (2.53) в (2.52) и определяя коэффициент при t^n , можно убедиться, что $A_n = coeff_{t^n}\{A(t)\}$ представляется линейной комбинацией функций

$$C_{m+n}^{m} \lambda_i^n,\ m = 0,\ 1, ..., e_i - 1;\ i = 1, ..., r. \qquad (2.54)$$

Другими словами, функции (2.54) образуют базис в пространстве решений рекуррентного соотношения (2.25). Ранее без

доказательства было сформулировано, что решениями этого соотношения являются линейные комбинации функций

$$n^m \cdot \lambda_i^n,\ m = 0,\ 1,\ ...,\ e_i - 1;\ i = 1,\ ...,\ r. \qquad (2.55)$$

Можно показать, что функции (2.54) линейно выражаются через функции (2.55). Например, при $m = 2$

$$C_{n+2}^2 \lambda^n = \frac{(n+2)!}{2!n!}\lambda^n = \frac{(n+2)(n+1)}{2}\lambda^n = \frac{1}{2}n^2\lambda^n + \frac{3}{2}n\lambda^n + \lambda^n.$$

Таким образом, метод производящих функций позволил строго обосновать сформулированную ранее процедуру решения линейного рекуррентного соотношения.

3. Дискретные структуры: графы, сети, коды

Теория графов и сетей может быть отнесена к конечной геометрии и рассматриваться как раздел дискретной математики, исследующий свойства конечных множеств с заданными отношениями между их элементами.

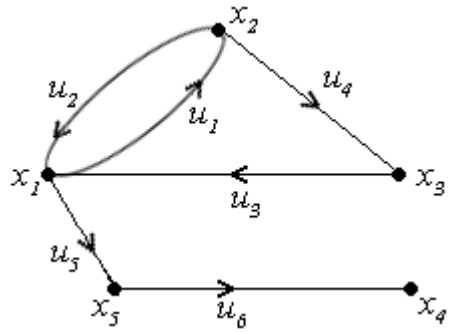
В последнее время теория графов и сетей стала мощным средством решения вопросов, относящихся к широкому кругу проблем. Это проблемы проектирования интегральных схем и схем управления, исследования автоматов, логических цепей, блок-схем программ, экономики и статистики, химии и биологии, теории расписаний и дискретной оптимизации.

3.1. Основные понятия теории графов

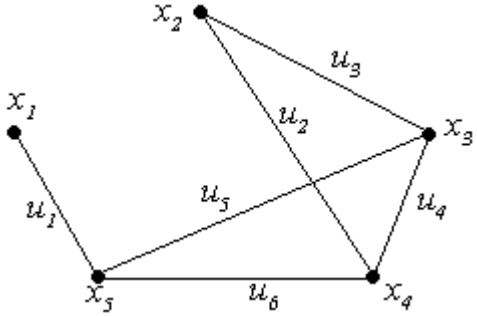
Граф отличается от геометрических конфигураций, которые также состоят из точек-вершин и сторон-линий тем, что в графе несущественны расстояния между точками, форма соединяющих линий и углы между ними. Важно, лишь соединена ли данная пара точек линией или нет. Поэтому граф иногда называют геометрическим объектом, т. е. объектом, свойства которого не изменяются при растяжении, сжатии, искривлении. По этой же причине граф является дискретным объектом, который может быть задан двумя дискретными множествами: множеством точек и множеством линий, соединяющих некоторые точки. Это можно осуществить теоретико-множественным, геометрическим и матричным способами.

Пусть задано непустое множество X. Обозначим через $V(X)$ множество всех пар (упорядоченных или неупорядоченных) его различных элементов.

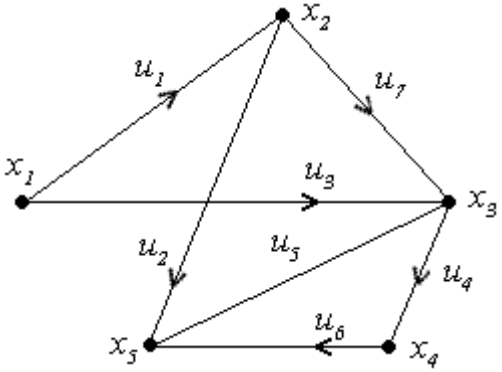
Графом G называется пара (X, U), где $U \subseteq V(X)$. Элементы из X называются *вершинами* графа, а элементы из U – его *ребрами*. Если U – множество упорядоченных пар, то граф называется *ориентированным* (*орграфом*) (рис. 3.1, а), в противном случае – *неориентированным* (рис. 3.1, б). Элементы множества U, то есть пары вершин, для неориентированного



а)



б)



в)

Рис. 3.1. (а) – ориентированный граф, (б) – неориентированный граф, (в) – смешанный граф

графа, называются *ребрами*, а для ориентированного – *дугами*. В случае когда в орграфе $G=(X,\ U)$ необходимо пренебречь направленностью дуг, то неориентированный граф, соответствующий G, обозначается как $\bar{G}=(X,\ \bar{U})$ и называется *неориентированным дупликатом* (или *неориентированным двойником*).

В ряде случаев естественно рассматривать *смешанные* графы (рис. 3.1, в), имеющие как ориентированные, так и неориентированные ребра. Например, план города можно рассматривать как граф, в котором ребра представляют улицы, а вершины – перекрестки; при этом по одним улицам может допускаться лишь одностороннее движение, и тогда на соответствующих ребрах вводится ориентация; по другим улицам движение двустороннее, и на соответствующих ребрах ориентация не вводится.

Граф, не имеющий ребер ($U=\varnothing$), называется *пустым* или *нуль-графом*. Все вершины пустого графа изолированные. Граф, в котором каждая пара вершин соединена ребром, называется *полным*. Полный n-вершинный неориентированный граф обозначается K_n. Для полного графа $U=V(X)$, а число ребер равно $C_n^2=n\cdot(n-1)/2$.

Граф с кратными ребрами называют *мультиграфом.* Если в мультиграфе имеются петли, то он называется *псевдографом*. Граф без петель и кратных ребер называется *простым* графом. В большинстве приложений теории графов можно отбрасывать петли и заменять кратные ребра одним ребром.

Важнейшими количественными характеристиками графа являются: число вершин $n=|X|$, определяющее *порядок* графа, и число ребер $m=|U|$. Граф с n вершинами и m ребрами называется $(n,\ m)$-*графом*.

Часто описание орграфа G состоит в задании множества вершин X и *соответствия* Γ, которое показывает как между собой связаны вершины. Соответствие Γ называется *отображением* множества X в X, а граф в этом случае обозначается парой $G=(X,\ \Gamma)$.

Для графа на рис. 3.1(а) имеем $\Gamma(x_1)=\{x_2,\ x_5\}$, т. е. вершины x_2 и x_5 являются конечными вершинами дуг, у которых начальной вершиной является x_1; $\Gamma(x_2)=\{x_1,\ x_3\}$; $\Gamma(x_3)=\{x_1\}$; $\Gamma(x_4)=\varnothing$; $\Gamma(x_5)=\{x_4\}$.

В случае неориентированного или смешанного графов предполагается, что соответствие Γ задает такой эквивалентный орграф, который получается из исходного графа заменой каждого неориентированного ребра двумя противоположно направленными дугами, соединяющими те же самые вершины. Так, например, для графа, приведенного на рис. 3.1, б имеем $\Gamma(x_5)=\{x_1,\ x_3,\ x_4\}$, $\Gamma(x_1)=\{x_5\}$ и т. д.

Поскольку $\Gamma(x_i)$ представляет собой множество таких вершин $x_j \in X$, для которых в графе G существует дуга $(x_i,\ x_j)$, то через $\Gamma^{-1}(x_i)$ естественно обозначить множество вершин x_k, для которых в графе G существует дуга $(x_k,\ x_i)$. Отношение $\Gamma(x_i)$ принято называть *обратным соответствием*. Для графа, изображенного на рис. 3.1(а), имеем $\Gamma^{-1}(x_1)=\{x_2,\ x_3\}$, $\Gamma^{-1}(x_2)=\{x_1\}$ и т. д.

Очевидно, что для неориентированного графа $\forall x_i \in X:\ \Gamma^{-1}(x_i)=\Gamma(x_i)$.

Когда отображение Γ действует на множество вершин $X_q=\{x_1,\ ...,\ x_q\}$, то под $\Gamma(X_q)$ понимают объединение $\bigcup_{i=1}^{q}\Gamma(x_i)$. Для графа на рис. 3.1(а) $\Gamma(\{x_2,\ x_5\})=\{x_1,\ x_3, x_4\}$, $\Gamma(\{x_1,\ x_3\})=\{x_2,\ x_5, x_1\}$.

Отображение $\Gamma(\Gamma(x_i))$ записывается как $\Gamma^2(x_i)$. Аналогично «тройное» отображение $\Gamma(\Gamma(\Gamma(x_i)))$ записывается как $\Gamma^3(x_i)$ и т. д. Для графа на рис. 3.1(а) имеем:

$$\Gamma^2(x_1)=\Gamma(\Gamma(x_1))=\Gamma(\{x_2,\ x_5\})=\{x_1,\ x_3, x_4\},$$

$$\Gamma^3(x_1) = \Gamma(\Gamma^2(x_1)) = \Gamma(\{x_1,\ x_3, x_4\}) = \{x_1,\ x_2, x_5\}$$

и т. д. Аналогично понимаются обозначения $\Gamma^{-2}(x_i)$, $\Gamma^{-3}(x_i)$ и т. д.

При наглядном представлении графа вершины изображаются точками, ребра – линиями, соединяющими точки, а дуги – направленными линиями. Зафиксируем на плоскости произвольным образом n точек, которые в любом порядке обозначим как $x_1, x_2, ..., x_n$. Затем для каждой пары точек (x_i, x_j) таких, что $(x_i, x_j) \in U$, проведем линию, соединяющую точки x_i и x_j В результате таких действий получим некоторый рисунок, который называется *геометрической интерпретацией* графа. Заметим, что одному и тому же графу соответствует много рисунков, которые могут быть его геометрическими интерпретациями.

Пусть в графе $G = (X, U)$ каждой вершине $x_i \in X$ ($i = 1, ..., n$) взаимно однозначно соответствует точка a_i в евклидовом пространстве, а каждому ребру $u = (x_i, x_j) \in U$ – непрерывная кривая l, соединяющая точки x_i и x_j и не проходящая через другие точки. Если все кривые, соответствующие ребрам, не имеют общих точек, кроме, быть может, концевых, то это множество точек и кривых называется *геометрической реализацией графа G в евклидовом пространстве*. Справедливо следующее утверждение.

Т е о р е м а 3.1. *Каждый конечный граф можно реализовать в трехмерном евклидовом пространстве.*

Если в графе $G = (X, U)$ пара вершин x_i, x_j такова, что $(x_i, x_j) \in U$, то вершины x_i, x_j называются *смежными*; в этой ситуации каждая из них называется *инцидентной ребру* (x_i, x_j), а ребро (x_i, x_j) называется *инцидентным* каждой из вершин x_i, x_j. Если вершина x_i и ребро u_j инцидентны, то пишут $x_i \in u_j$.

Количество ребер, инцидентных данной вершине x, называется ее *степенью* или *локальной степенью* графа в вершине x и обозначают через $d(x)$. Для неориентированного графа $d(x) \equiv |\Gamma(x)|$. Вершина с нулевой степенью называется *изолированной*. Вершина, степень которой равна единице, называется *висячей.*

Если ребро (дуга) инцидентно только одной вершине, то его называют *петлей*. Ребра называют *кратными*, если они инцидентны одним и тем же вершинам.

Исторически первой теоремой теории графов является утверждение, принадлежащее Эйлеру и связывающее количество ребер, вершин и их степеней.

Теорема 3.2. *Сумма степеней вершин* (n, m)*-графа равна удвоенному числу его ребер*:

$$\sum_{i=1}^{n} d(x_i) = 2m.$$

Доказательство. Поскольку любое ребро инцидентно двум вершинам, то в сумме степеней всех вершин графа каждое ребро учитывается дважды. ▪

Следствие. *В любом графе число вершин нечетной степени четно*.

Доказательство. Пусть X_1 и X_2 – множества вершин четной и нечетной степени соответственно. Очевидно, что

$$\sum_{x \in X_1} d(x) + \sum_{x \in X_2} d(x) = 2m.$$

Первое слагаемое четно (как сумма четных чисел). Тогда второе слагаемое также должно быть четным, а это при суммировании нечетных чисел возможно, если только их количество четно. ▪

Для орграфов вместо степени вершины вводят понятие полустепеней: *полустепень исхода* $d_{-}(x)$ – это число дуг, выходящих из вершины x; *полустепень захода* $d_{+}(x)$ – это число дуг, входящих в вершину x. Очевидно, что

$$\sum_{i=1}^{n} d_{-}(x) = \sum_{i=1}^{n} d_{+}(x) = m.$$

Матрицей смежности (n, m)-графа G называется квадратная матрица $S = [s_{ij}]$ размера n, которая определяется следующим образом:

$$s_{ij} = \begin{cases} 1, & (x_i, x_j) \in U, \\ 0, & (x_i, x_j) \notin U. \end{cases}$$

Для графа, изображенного на рис. 3.4, матрица смежности имеет вид:

$$S = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Матрица смежности полностью определяет структуру графа. Например,

$$\sum_{j=1}^{n} s_{ij} = d_{-}(x_i), \ \sum_{i=1}^{n} s_{ij} = d_{+}(x_j), \ i = 1, ..., n.$$

Множество столбцов, имеющих 1 в i-ой строке, есть множество $\Gamma(x_i)$, а множество строк, которые имеют 1 в j-ом столбце, совпадает с множеством $\Gamma^{-1}(x_j)$.

Для неориентированного графа без кратных ребер и петель матрица смежности симметрична и имеет нули по главной диагонали. Для такого графа

$$\sum_{j=1}^{n} s_{ij} = d(x_i), \ i = 1, ..., n.$$

Возведем матрицу смежности в квадрат. Элемент $s_{ij}^{(2)}$ матрицы $S^2(G)$ определяется по формуле

$$s_{ij}^{(2)} = \sum_{k=1}^{n} s_{ik} s_{kj}.$$

Слагаемое в этом уравнении равно 1 тогда и только тогда, когда оба числа равны 1, в противном случае оно равно 0. Поскольку из равенства $s_{ik} = s_{kj}$ следует существование пути длины 2 из вершины x_i в вершину x_j, проходящего через вершину x_k, то $s_{ij}^{(2)}$ равно числу путей длины 2, идущих из x_i в x_j.

Аналогично если $s_{ij}^{(p)}$ является элементом матрицы S^p, то $s_{ij}^{(p)}$ равно числу путей длины p, идущих от x_i к x_j.

Матрицей инциденций (n, m)-графа G называется прямоугольная матрица $C = [c_{ij}]$ размерности $[n \times m]$, определяемая следующим образом:

$$c_{ij} = \begin{cases} 1, \text{ если } x_i \text{ – конец дуги или ребра } u_j; \\ -1, \text{ если } x_i \text{ – начало дуги } u_j \text{ (для ребра +1)}; \\ 0, \text{ если } x_i \notin u_j. \end{cases}$$

Для графа, приведенного на рис. 3.2, матрица инциденций имеет вид:

$$C = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 \end{bmatrix}.$$

Поскольку каждая дуга (ребро) инцидентна двум различным вершинам, за исключением того случая, когда дуга (ребро) образует петлю (в этом случае элементы соответствующего столбца равны 0), то в каждом столбце матрицы – два ненулевых элемента.

Если в матрице инциденций нет нулевых элементов, то имеем полный орграф, который называется *турниром*.

Пусть $G_1 = (X_1, U_1)$, $G_2 = (X_2, U_2)$ – два графа, для которых $X_1 \subseteq X_2$, $U_1 \subseteq U_2$; тогда говорят, что G_1 является *подгра-*

фом графа G_2. В свою очередь, граф G_2 по отношению к своему подграфу G_1 является *надграфом*. Если $X_1 = X_2$ и $U_1 \subseteq U_2$, то

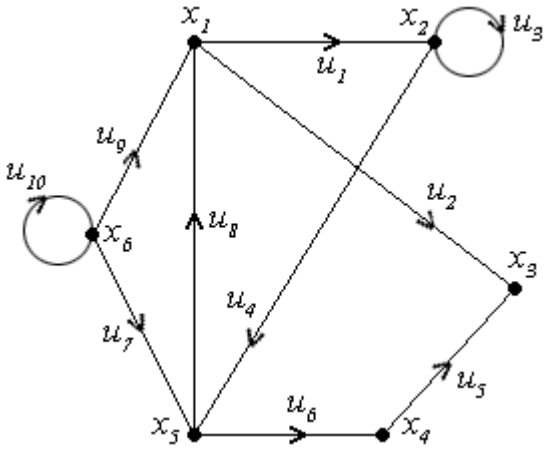


Рис. 3.2

такой подграф называется *остовным*. Любой остовный подграф $G_1 = (X,\ U_1)$ графа $G_2 = (X,\ U_2)$ может быть получен путем удаления подмножества $U_2 \setminus U_1$ ребер надграфа.

Подграфы другого типа получаются, если выбрать подмножество X' вершин надграфа и присоединить к ним все ребра, концы которых принадлежат X'. Это *вершинно-порожденные* подграфы. Любой вершинно-порожденный подграф $G_v = (X',\ U')$ графа $G = (X,\ U)$ можно получить путем удаления подмножества $U \setminus U'$ вершин и всех инцидентных им ребер надграфа.

Наконец, *реберно-порожденным* будем называть подграф $G_r = (X',\ U')$, полученный на основе произвольно выбранного подмножества U' ребер надграфа, причем множество вершин X' подграфа составляют концы только этих ребер. Любой реберно-порожденный подграф можно сформировать, если в

надграфе сперва удалить подмножество ребер $U \setminus U'$, а затем в получившемся остовном подграфе удалить все изолированные вершины.

Пусть имеются изображения графов одного порядка с одинаковым числом ребер. Требуется установить, разные это графы или один, только по разному изображенный. Для решения этой задачи используется понятие изоморфизма.

Изоморфизмом называют взаимно однозначное соответствие между множествами вершин двух графов G_1 и G_2, сохраняющее отношение смежности, а сами графы называются *изоморфными. Автоморфизмом* называется изоморфное отображение графа на себя. Для установления изоморфности достаточно составить списки вершин всех графов, указав для каждой из них все ее соседние вершины. В данном случае все эквивалентные вершины пронумерованы одинаково, что упрощает решение задачи. Задача усложняется, если эквивалентные вершины имеют разные номера. В этом случае процедуру работы со списками придется повторять неоднократно, каждый раз меняя нумерацию вершин одного из графов, пока не будет обнаружен изоморфизм, т. е. получены идентичные списки вершин. В худшем случае потребуется проверить $n!$ вариантов нумерации вершин. Доказать неизоморфизм графов можно, только выполнив все проверки. Существенно сократить их количество можно, если использовать инварианты.

Инвариантом называют некоторую числовую характеристику графа G, которая принимает одно и то же значение для любого графа, изоморфного G. Только при совпадении значений инвариантов переходят к перебору допустимых вариантов нумерации вершин. Тривиальными инвариантами графа являются $|X|$ и $|U|$. К числу инвариантов относится и степенная последовательность графа, т. к. при изоморфизме степень вершины не изменяется. Есть множество других инвариантов, однако не известна (возможно, не существует) система инвариантов, позволяющая решать задачу изоморфизма для всех видов графов.

Понятие изоморфизма графов имеет следующее наглядное представление. Представим ребра графа эластичными нитями, связывающими узлы – вершины графа. Тогда изоморфизм можно представить как перемещение узлов и растяжение нитей.

Оценку сверху для числа $\gamma(m)$ попарно неизоморфных графов дает следующая теорема.

Т е о р е м а 3.3. $\forall m \in N:\ \gamma(m) < (15 \cdot m)^m$.

Д о к а з а т е л ь с т в о. Очевидно, что число вершин в каждом из рассматриваемых графов не превосходит $2 \cdot m$.Занумеруем их числами 1, 2, …, $2m$. Каждое ребро определяется двумя вершинами (концами), не обязательно различными, так что для каждого ребра имеется не более $(2m)^2$ возможностей выбора концов, а для m ребер – не более, чем $\bar{C}^m_{4 \cdot m^2} = C^m_{4 \cdot m^2 + m - 1}$. Следовательно, $\gamma(m) \le C^m_{4 \cdot m^2 + m - 1} \le \dfrac{(5m^2)^m}{m!}$. Используя неравенство $m! > \left(\dfrac{m}{3}\right)^m$, получаем $\gamma(m) < \dfrac{(5m^2)^m 3^m}{m^m} = (15m)^m$. ▪

3.2. Достижимость и связность в графах

Ориентированным маршрутом (или *путем*) орграфа называется последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальной вершиной следующей. На рис. 3.3 последовательности дуг

$$u_1,\ u_6,\ u_5,\ u_9,\ u_{10},\ u_6,\ u_4, \qquad (3.1)$$

$$u_6,\ u_5,\ u_9,\ u_8,\ u_4, \qquad (3.2)$$

$$u_1,\ u_6,\ u_5,\ u_9 \qquad (3.3)$$

являются путями.

Ориентированной цепью (или *орцепью*) называется путь, в котором каждая дуга используется не больше одного раза. Так, например, пути (3.2) и (3.3) являются орцепями, а путь (3.1) не

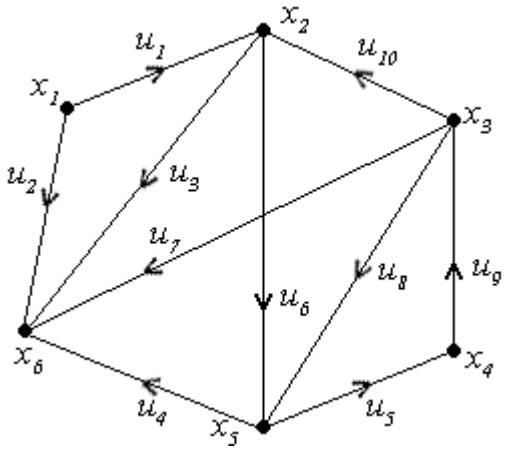


Рис. 3.3

является таким, поскольку дуга u_6 в нем используется дважды.

Простой называется орцепь, в которой каждая вершина используется не более одного раза. Например, орцепь (3.3) является простой, а орцепь (3.2) – нет.

Маршрут есть неориентированный двойник пути, т. е. последовательность ребер $\overline{u}_1$, $\overline{u}_2$, ..., $\overline{u}_l$, в которой каждое ребро $\overline{u}_i$, за исключением первого и последнего, связано концевыми вершинами с ребрами $\overline{u}_{i-1}$ и $\overline{u}_{i+1}$. Черта над символом дуги означает, что ее рассматривают как ребро.

Аналогично тому, как мы определили орцепи и простые орцепи, можно определить цепи и простые цепи.

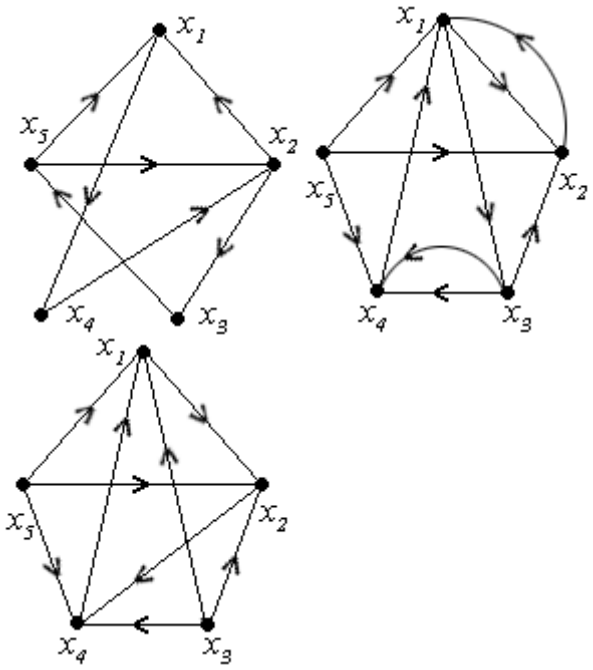
Путь или маршрут можно изображать также последовательностью вершин. Например, путь (3.1) можно записать в виде: $x_1, x_2, x_5, x_4, x_3, x_2, x_5, x_6$.

Путь $u_1, u_2, ..., u_l$ называется *замкнутым*, если в нем начальная вершина дуги u_1 совпадает с конечной вершиной дуги u_l. Замкнутые орцепи (цепи) называются *орциклами* (*циклами*). Орциклы называют также *контурами*. Замкнутые простые орцепи (цепи) называются *простыми орциклами* (*циклами*). *За-*

мкнутый маршрут является неориентированным двойником замкнутого пути.

Говорят, что вершина x_j в орграфе G достижима из вершины x_i, если существует путь $(x_i \to ... \to x_j)$. Если при этом x_i достижима из x_j, то об этих вершинах говорят, что они взаимно достижимы.

Орграф называют *сильно связным* или *сильным*, если любые две вершины в нем взаимно достижимы. Пример сильного орграфа приведен на рис. 3.4(а). Поскольку в графе имеется орцикл $(x_1, x_4, x_2, x_5, x_1)$, включающий все вершины, то любые две из них взаимно достижимы.



а б в

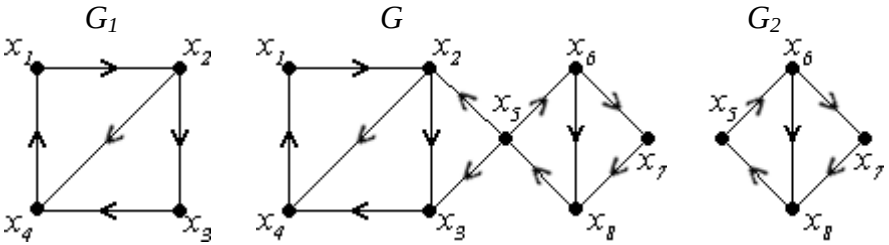
Рис. 3.4. (а) – сильный орграф, (б) – односторонний орграф, (в) – слабый орграф

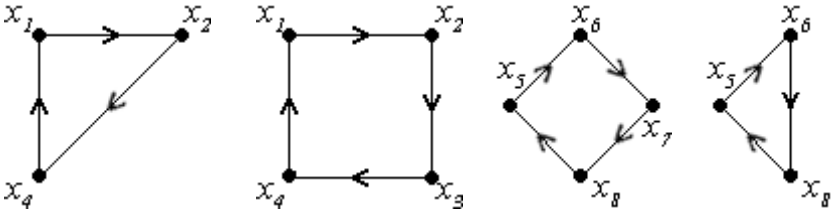
Орграф называется *односторонне связным* или *односторонним*, если в любой паре его вершин хотя бы одна достижима из другой. Пример одностороннего графа приведен на рис. 3.4(б). В этом графе есть орцикл $(x_1, x_4, x_3, x_1, x_2, x_1)$, включающий четыре взаимно достижимые вершины. Вершина x_5 имеет нулевую степень захода, а это значит, что не существует путей, ведущих в эту вершину. При этом из x_5 достижима любая из четырех остальных вершин.

Орграф называется *слабо связным* или *слабым*, если в нем любые две вершины соединены *полупутем*. Полупуть, в отличие от пути, может иметь противоположно направленные дуги. Пример слабого орграфа приведен на рис. 3.4(в). Очевидно, что в графе нет пути между вершинами x_3 и x_5, но существует полупуть, состоящий из противоположно направленных дуг (x_3, x_4) и (x_5, x_4).

Если для некоторой пары вершин не существует маршрута, соединяющего их, то такой орграф называется *несвязным*.

Для орграфа определены три типа компонент связности: *сильная компонента* – максимально сильный подграф, *односторонняя компонента* – максимальный односторонний подграф и *слабая компонента* – максимально слабый подграф. Понятие сильной компоненты иллюстрирует рис. 3.5.





G_1' G_1'' G_2' G_2''

Рис. 3.5.

Граф G, который является односторонне связным, содержит шесть сильных подграфов G_1, G_1', G_1'', G_2, G_2', G_2'', из которых только G_1 и G_2 являются сильными компонентами. Аналогично иллюстрируется понятие односторонней компоненты. В данном примере односторонняя компонента совпадает с самим графом. Если же сменить ориентацию дуги, например, (x_4, x_1) на противоположную, то получим слабый граф с двумя односторонними компонентами, одна из которых образована вершинами (x_1, x_4), а другая – вершинами (x_2, x_8).

Для неориентированного графа G определим на множестве его вершин X бинарное отношение, полагая $x \sqcup y$, если имеется цепь, связывающая x с y. Это отношение обладает свойствами рефлексивности, симметричности и транзитивности, то есть является отношением эквивалентности. Оно разбивает множество вершин на непересекающиеся классы: $X = \bigcup_{k=1}^{p} X_k$. Две вершины из одного класса эквивалентны, то есть в графе имеется цепь, соединяющая их, для вершин из разных классов такой цепи нет. Так как концы любого ребра находятся в отношении $\sqcup$, то множество ребер графа G также разобьется на непересекающиеся классы: $U = \bigcup_{k=1}^{p} U_k$, где через U_k обозначено множество всех ребер, концы которых принадлежат X_k, $k = 1, \ldots, p$.

Графы $G_k = (X_k, U_k)$, $k = 1, ..., p$ являются связными и в сумме дают граф $G = (X, U)$. Эти графы называются *компонентами связности* графа G. Число p – еще одна числовая характеристика графа. Для связного графа $p = 1$, если граф несвязный, то $p \geq 2$.

Если данный граф не является связным и распадается на несколько компонент, то решение какого-либо вопроса относительно этого графа, как правило, можно свести к изучению отдельных компонент, которые связны. Поэтому в большинстве случаев имеет смысл предполагать, что заданный граф связный.

Для связного графа G определим *расстояние* между двумя его вершинами x и y как длину самой короткой цепи, соединяющей эти вершины, и обозначим через $d(x, y)$. Длина цепи – это количество ребер, составляющих цепь. Нетрудно проверить, что введенное расстояние удовлетворяет аксиомам метрики:

1) $d(x, y) \geq 0$; $d(x, y) = 0 \Leftrightarrow x = y$;

2) $d(x, y) = d(y, x)$;

3) $d(x, y) \leq d(x, z) + d(z, y)$.

Определим расстояние от каждой вершины x графа G до самой далекой от нее вершины

$$e(x) = \max_y d(x, y),$$

которое называется *эксцентриситетом*. Очевидно, что эксцентриситет для всех вершин полного графа равен единице, а для вершин простого цикла C_n – $E\left(\left|U_{C_n}\right| / 2\right)$.

Максимальный эксцентриситет $d(G) = \max_x e(x)$ носит название *диаметра* графа, а минимальный $r(G) = \min_x e(x)$ – *радиуса* графа G. В полном графе имеем $d(K_n) = r(K_n) = 1$, а в простом цикле C_n – $d(C_n) = r(C_n) = E\left(\left|U_{C_n}\right| / 2\right)$.

Вершина x_0 называется центральной, если $e(x_0) = r(G)$. Граф может иметь несколько таких вершин, а в некоторых графах все вершины являются центральными. В простой цепи при

нечетном числе вершин только одна является центральной, а при четном их числе таких вершин две. В полном графе и для простого цикла центральными являются все вершины. Множество центральных вершин называется *центром* графа.

Понятия центральной вершины и центра графа появились в связи с задачами оптимального размещения пунктов массового обслуживания, таких как больницы, пожарные части, пункты охраны общественного порядка и т. п., когда важно минимизировать наибольшее расстояние от любой точки некоторой сети до ближайшего пункта обслуживания.

Пример 3.1. Найти диаметр, радиус и центр графа, приведенного на рис. 3.6.

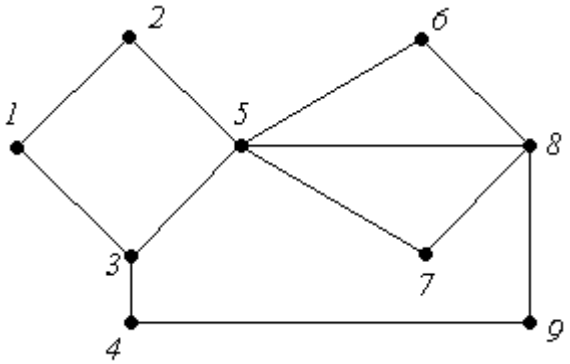


Рис. 3.6

Для решения этой задачи удобно предварительно вычислить *матрицу расстояний* между вершинами графа. В данном случае это будет матрица размером $[9\times 9]$, в которой на месте (i, j) стоит расстояние от вершины i до вершины j:

$$\left(\begin{array}{cccccccc|c} 0&1&1&2&2&3&3&3&3\\ 1&0&1&2&1&2&2&2&3\\ 1&1&0&1&1&2&2&2&2\\ 2&2&1&0&2&3&3&2&1\\ 2&1&1&2&0&1&1&1&2\\ 3&2&2&3&1&0&2&1&2\\ 3&2&2&3&1&2&0&1&2\\ 3&2&2&2&1&1&1&0&1\\ 3&3&2&1&2&2&2&1&0 \end{array}\begin{array}{c}|3\\|3\\|2\\|3\\|2\\|3\\|3\\|3\\|3\end{array}\right).$$

Для каждой строки матрицы находим наибольший элемент и записываем его справа от черточки. Наибольшее из этих чисел равно диаметру графа $d(G)=3$, наименьшее – радиусу графа $r(G)=2$. Центр графа составляют центральные вершины $x_0^1=3$ и $x_0^2=3$.

Матрица достижимостей $R=[r_{ij}]$ определяется следующим образом:

$$r_{ij}=\begin{cases}1, \text{ если вершина } x_j \text{ достижима из } x_i,\\ 0 \text{ в противном случае.}\end{cases}$$

Множество вершин $R(x_i)$ графа G, достижимых из заданной вершины x_i, состоит из таких элементов x_j, для которых (i, j)-й элемент в матрице R равен 1. Это множество можно представить в виде

$$R(x_i)=\{x_i\}\cup\Gamma^1(x_i)\cup\Gamma^2(x_i)\cup\ \ldots\ \cup\Gamma^p(x_i).$$

Матрица контрадостижимостей (*обратных достижимостей*) $Q=[q_{ij}]$ определяется следующим образом:

$$q_{ij}=\begin{cases}1, \text{ если вершина } x_i \text{ достижима из } x_j,\\ 0 \text{ в противном случае.}\end{cases}$$

Аналогично построению достижимого множества $R(x_i)$ можно сформировать множество $Q(x_i)$, используя следующее выражение:

$$Q(x_i)=\{x_i\}\cup\Gamma^{-1}(x_i)\cup\Gamma^{-2}(x_i)\cup\ \ldots\ \cup\Gamma^{-p}(x_i).$$

Из определений следует, что i-й столбец матрицы Q совпадает с i-й строкой матрицы R, т. е. $Q = R^t$, где R^t – матрица, транспонированная к матрице R.

Пример 3.2. Найти матрицы достижимостей и контрадостижимостей для графа, приведенного на рис. 3.7.

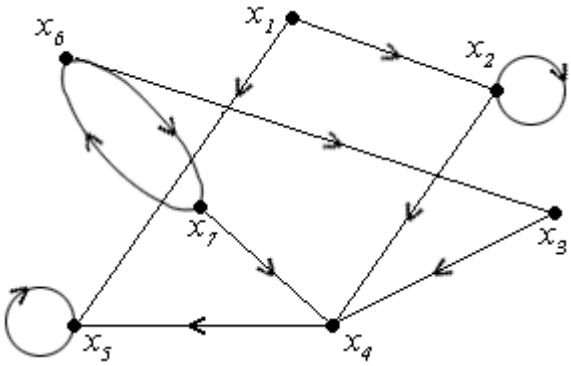


Рис. 3.7

Определим множества достижимостей для вершин графа:

$$R(x_1) = \{x_1\} \cup \{x_2,\ x_5\} \cup \{x_2,\ x_4,\ x_5\} \cup \{x_2,\ x_4,\ x_5\} = \{x_1,\ x_2,\ x_4,\ x_5\},$$

$$R(x_2) = \{x_2\} \cup \{x_2,\ x_4\} \cup \{x_2,\ x_4,\ x_5\} \cup \{x_2,\ x_4,\ x_5\} = \{x_2,\ x_4,\ x_5\},$$

$$R(x_3) = \{x_3\} \cup \{x_4\} \cup \{x_5\} \cup \{x_5\} = \{x_3,\ x_4,\ x_5\},$$

$$R(x_4) = \{x_4\} \cup \{x_5\} \cup \{x_5\} = \{x_4,\ x_5\},$$

$$R(x_5) = \{\{x_5\} \cup \{x_5\} = \{x_5\},$$

$$R(x_6) = \{x_6\} \cup \{x_3,\ x_7\} \cup \{x_4,\ x_6\} \cup \{x_3,\ x_5,\ x_7\} \cup \{x_4,\ x_5,\ x_6\} =$$

$$= \{x_3,\ x_4,\ x_5,\ x_6,\ x_7\},$$

$$R(x_7) = \{x_7\} \cup \{x_4,\ x_6\} \cup \{x_3,\ x_5,\ x_7\} \cup \{x_4,\ x_5,\ x_6\} =$$

$$= \{x_3,\ x_4,\ x_5,\ x_6, x_7\}.$$

Следовательно, матрицы достижимостей и контрадостижимостей имеют вид:

$$R=\begin{bmatrix}1&1&0&1&1&0&0\\0&1&0&1&1&0&0\\0&0&1&1&1&0&0\\0&0&0&1&1&0&0\\0&0&0&0&1&0&0\\0&0&1&1&1&1&1\\0&0&1&1&1&1&1\end{bmatrix},\ Q=R^t=\begin{bmatrix}1&0&0&0&0&0&0\\1&1&0&0&0&0&0\\0&0&1&0&0&1&1\\1&1&1&1&0&1&1\\1&1&1&1&1&1&1\\0&0&0&0&0&1&1\\0&0&0&0&0&1&1\end{bmatrix}.$$

Так как $R(x_i)\bigcap Q(x_j)$ – множество таких вершин, каждая из которых принадлежит по крайней мере одному пути, идущему от x_i к x_j, то вершины этого множества называются *существенными* или *неотъемлемыми* относительно концевых вершин x_i и x_j. Все остальные вершины $x_k \notin R(x_i)\bigcap Q(x_j)$ называются *несущественными* или *избыточными*, поскольку их удаление не влияет на пути от x_i к x_j.

Можно определить также матрицы *ограниченных* достижимостей и контрдостижимостей, если потребовать, чтобы длины путей не превышали некоторого заданного числа. Тогда p будет верхней границей длины допустимых путей.

Граф называют *транзитивным*, если из существования дуг (x_i, x_j) и (x_j, x_k) следует существование дуги (x_i, x_k). *Транзитивным замыканием* графа $G=(X, U)$ является граф $G_t=(X, U\bigcup U')$, где U' – минимально возможное множество дуг, необходимых для того, чтобы граф G_t был транзитивным. Очевидно, что матрица достижимостей R графа G совпадает с матрицей смежности S графа G_t, если в матрице S на главной диагонали поставить единицы.

3.3. Деревья

Одним из наиболее важных понятий, которое часто используется в различных приложениях теории графов, является дерево. С помощью деревьев легко описывается структура самых различных объектов: организаций и учреждений, книг и

документов, математических формул, химических соединений, компьютерных файловых систем, программ и многое другое.

Понятие дерева как математического объекта было впервые предложено Кирхгоффом в 1847 г. в связи с определением фундаментальных циклов, применяемых при анализе электрических цепей. Спустя десять лет химик Кели вновь (независимо от Кирхгоффа) ввел понятие дерева при изучении структуры углеводородных соединений и получил первые важные результаты в этом разделе теории графов.

Граф без циклов называется *ациклическим* или *лесом*. Связный ациклический граф называется *деревом*. Если G – лес, то каждая его компонента является деревом. *Листом* называют вершину, степень которой равна 1, если она не рассматривается как корень. В качестве корня в неориентированном дереве можно принять любую вершину.

Ребро графа G, через которое проходит хотя бы один цикл, называется *цикловым*. Ребро, которое не входит ни в один цикл, называется *перешейком* или *мостом*.

Приведем эквивалентные определения неориентированного дерева:

1) дерево – это связный граф, содержащий n вершин и $n-1$ ребер;

2) дерево – это граф, в котором каждая пара вершин соединена одной и только одной простой цепью;

3) дерево – это граф, в котором каждое ребро является мостом.

Рассмотрим связный граф $G=(X,\ U)$ и будем из него удалять по одному цикловые ребра до получения ациклического подграфа. В результате получим *остовное* дерево $T=(X',\ U')$ графа G, для которого $X'=X$, $U'\subseteq U$.

Так как удаление цикловых ребер можно вести разными способами, то один и тот же граф в общем случае имеет несколько остовных деревьев. На рис. 3.8 представлен граф G и три его остовных дерева.

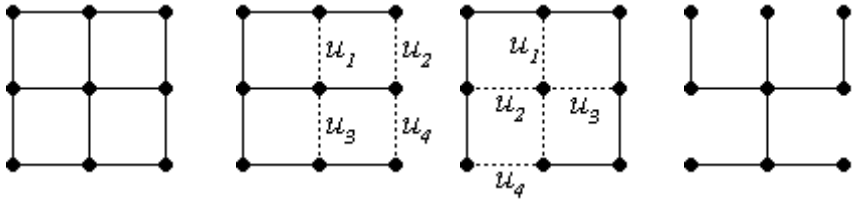


Рис. 3.8. Граф G и его остовные деревья T_1, T_2 и T_3

Ребра графа G, не вошедшие в его остовное дерево T, называются *хордами* дерева T.

Лемма 1. *В графе G для любого остовного дерева T и любой хорды h этого дерева существует единственный цикл, содержащий хорду h и не содержащий других хорд.*

Доказательство. Пусть $h=\{a, b\}$. В дереве T имеется единственная цепь, соединяющая вершины a и b. Присоединяя к этой цепи ребро h, получим требуемый цикл. ▪

Поставим в соответствие каждому ребру u связного графа $G=(X, U)$ целое число $l(u)\geq 0$ и будем называть его весом (длиной, стоимостью) ребра u. В результате получим граф с нагруженными ребрами или просто нагруженный граф, который обозначается $G=(X, U, L)$, где $L=[l_{ij}]$ – матрица весов. Для любой цепи μ ее вес равен сумме весов составляющих ее ребер: $l(\mu)=\sum_{u\in\mu} l(u)$.

Рассмотрим задачу поиска кратчайшего остова: для нагруженного графа G требуется построить остовное дерево T, сумма длин ребер которого минимальна.

Этой задаче можно дать такую интерпретацию: n пунктов на местности нужно связать сетью дорог, трубопроводов или линий телефонной связи. Для каждой пары пунктов i и j задана стоимость их соединения $l(i, j)$, которая представляет длину ребра $\{i, j\}$. Требуется построить связывающую сеть минимальной стоимости, которую называют *кратчайшей связывающей сетью*. Очевидно, что эта сеть будет остовным деревом графа

G, при этом среди всех остовных деревьев она будет иметь минимальную сумму длин входящих в нее ребер.

Алгоритм Краскала построения кратчайшей связывающей сети состоит из $n-1$ шагов, на каждом из которых присоединяется одно ребро. Правило для выбора этого ребра следующее: среди еще не выбранных ребер берется самое короткое, не образующее цикла с уже выбранными ребрами.

Пример 3.3. Дана матрица расстояний $C=[c_{ij}]$, в которой элемент c_{ij} – вес ребра, который указывает в условных единицах затраты, необходимые для того, чтобы связать пункт i с пунктом j. Требуется с наименьшими затратами связать все пункты друг с другом.

$$C=\begin{pmatrix} 0&5&6&2&1&7&5&4&6\\ 5&0&7&7&5&1&4&6&5\\ 6&7&0&5&6&6&2&5&4\\ 2&7&5&0&1&7&6&4&5\\ 1&5&6&1&0&5&6&3&7\\ 7&1&6&7&5&0&4&5&5\\ 5&4&2&6&6&4&0&6&2\\ 4&6&5&4&3&5&6&0&7\\ 6&5&4&5&7&5&2&7&0 \end{pmatrix}.$$

Применение сформулированного выше алгоритма выглядит следующим образом. В матрице C отыскивается минимальный элемент, который вычеркивается, а соответствующее ему ребро заносится в сеть, если при этом не образуется цикл. Затем эти действия повторяются. Таким образом, на первых пяти шагах работы алгоритма будут выбраны ребра {1, 5}, {2, 6}, {4, 5}, {3, 7}, {7, 9}. Из оставшихся ребер минимальную длину имеет ребро {1, 4}, но в сеть оно не включается, так как образует цикл с уже выбранными ребрами. На следующих этапах алгоритма в сеть будут включены ребра {5, 8}, {2, 7} и {1, 2}.

Для обоснования алгоритма предположим, что дерево T, которое он строит, состоит из ребер $u_1, ..., u_{n-1}$, для которых $l(u_1) \le ... \le l(u_{n-1})$.

Рассмотрим любое другое дерево T' и упорядочим его ребра по возрастанию длин. Пусть первые $k-1$ ребер дерева T' такие же, как в дереве T, а k-е ребро отличается от u_k ($1 \le k \le n-1$). Присоединим к дереву T' ребро u_k. Тогда возникнет цикл, в который входит ребро u_k и какие-то ребра из дерева T'. Среди этих ребер обязательно найдется ребро u, длина которого не меньше, чем длина ребра u_k: иначе ребро u_k образовало бы цикл с ребрами меньшей длины, что исключается правилом выбора очередного ребра в рассмотренном алгоритме. Удалим из дерева T' ребро u, заменив его ребром u_k. В результате получим дерево, длина которого не больше, чем длина дерева T'. Аналогичным путем вводим в дерево T' ребра u_{k+1}, ..., u_{n-1}, при этом всякий раз длина дерева не увеличится. Это означает, что дерево T действительно кратчайшее.

3.4. Кратчайшие пути в графе

В практических приложениях имеет большое значение задача о нахождении кратчайшего пути между двумя вершинами связного неориентированного графа.

Формулировка задачи может иметь несколько вариантов.

Вариант 1. Дана сеть автомобильных дорог, например, Тульской области. Найти кратчайшие пути от Тулы до каждого города области (если двигаться можно только по дорогам).

Вариант 2. Имеется некоторое количество авиарейсов между городами мира. Для каждого известна стоимость. Найти маршрут минимальной стоимости (возможно с пересадками), например, из Торонто в Новосибирск.

Вариант 3. Есть план города с нанесенными на него местами расположения пожарных частей. Определить ближайшую к каждому дому пожарную станцию.

В математике разработан ряд методов для решения подобных задач. Однако в большинстве случаев методы, основанные на использовании графов, оказываются наименее трудоемкими.

Рассмотрим связный ненагруженный граф $G = (X, U)$, ребра которого имеют одинаковый вес (длину), принимаемый за единицу. Решим для этого графа следующую задачу: найти кратчайшую цепь, связывающую заданную начальную вершину a с заданной конечной вершиной b.

Поскольку рассматриваемые графы сравнительно просты, то кратчайший путь нетрудно найти просто путем перебора возможных путей. Однако для сложных графов должен быть найден систематический метод.

Общее правило для нахождения кратчайшего пути в графе состоит в том, чтобы каждой вершине x_i присвоить индекс λ_i, равный длине кратчайшего пути из данной вершины в начальную. Присваивание индексов производится в следующем порядке:

1) начальной вершине a присваивается индекс $\lambda_a = 0$;

2) всем вершинам, смежным с вершиной a, присваиваем индекс 1;

3) всем вершинам, смежными с вершинами, имеющими индекс 1, присваиваем индекс 2, и так далее, пока не будет помечена вершина b. Сам кратчайший путь найдем, если будем двигаться из конечной вершины в направлении убывания индексов.

На рис. 3.9 показаны индексы, которые получают вершины в процессе работы алгоритма. Так как вершина b получила отметку 7, то длина кратчайшей цепи от a до b равна 7. Эта цепь выделена на рисунке.

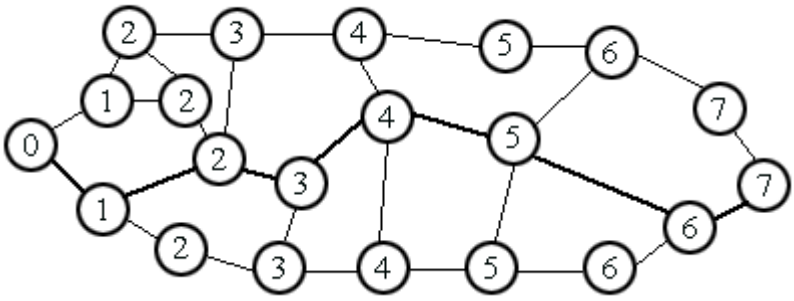


Рис. 3.9. Отыскание кратчайшего пути в ненагруженном графе

Описанный алгоритм называют *волновым*, так как процесс расстановки отметок напоминает распространение возмущения, которое возникает в вершине a и движется со скоростью одно ребро в единицу времени. Вершины, имеющие одинаковые отметки, представляют собой фронт волны.

Решим аналогичную задачу для связного нагруженного графа $G=(X,\ U,\ L)$. Алгоритм решения этой задачи, как и предыдущий, состоит в вычислении по определенным правилам индексов вершин. Индекс вершины x обозначим λ_x. После окончания процесса вычисления индексов они должны удовлетворять *условиям оптимальности*, которые заключаются в следующем:

$1^\circ.\ \lambda_a = 0$.

$2^\circ.\ \forall u=\{x,\ y\}:\ \lambda_y - \lambda_x \le l(x,\ y)$.

$3^\circ.\ \forall y \exists x: \lambda_y - \lambda_x = l(x,\ y)$, где x – смежная с y вершина.

При выполнении этих условий длина кратчайшей цепи между вершинами a и b равна λ_b, а сама кратчайшая цепь проходит по таким вершинам

$$a = x_0 \to x_1 \to \ldots \to x_{i-1} \to x_i \to x_{i+1} \to \ldots \to x_{k-1} \to x_k,$$

для которых $\lambda_{x_i} - \lambda_{x_{i-1}} = l(x_{i-1},\ x_i)$, $i=1,\ \ldots,\ k$.

Для доказательства этого утверждения построим цепь, используя свойство 3°. Длина такой цепи равна

$$l(a,\ x_1)+l(x_1,\ x_2)+\ldots+l(x_{k-1},\ b) = \lambda_{x_1} - \lambda_a + \lambda_{x_2} - \lambda_{x_1} + \ldots + \lambda_b - \lambda_{x_{k-1}} =$$
$$= \lambda_b - \lambda_a = \lambda_b.$$

Покажем, что длина любой другой цепи между вершинами a и b не меньше, чем λ_b.

Пусть

$$a = y_0 \to y_1 \to \ldots \to y_{i-1} \to y_i \to y_{i+1} \to \ldots \to y_{s-1} \to y_s$$

произвольная из таких цепей. В силу свойства 2°:

$$\begin{cases} \lambda_{y_1} - \lambda_a \le l(a,\ y_1); \\ \lambda_{y_2} - \lambda_{y_1} \le l(y_1,\ y_2); \\ \quad\quad ... \\ \lambda_b - \lambda_{y_{s-1}} \le l(y_{s-1},\ b). \end{cases}$$

Сложив эти неравенства, получим

$$\lambda_b \le l(a,\ y_1) + l(y_1,\ y_2) + ... + l(y_{s-1},\ b)\ .$$

Рассмотрим далее алгоритм Дейкстры, который обеспечивает присвоение вершинам графа отметок, удовлетворяющих условиям 1°–3°.

Алгоритм состоит из двух этапов:

1) инициализация алгоритма, которая заключается в начальной расстановке отметок;

2) циклически повторяющаяся процедура исправления отметок.

На каждом шаге алгоритма все отметки делятся на *предварительные* и *окончательные*. Алгоритм обрабатывает только предварительные отметки и заканчивает работу, когда все отметки станут окончательными.

Каждой вершине $x \in X$ графа $G = (X,\ U,\ L)$ поставим в соответствие метку – минимальное известное расстояние от вершины x до начальной вершины a. Алгоритм работает пошагово – на каждом шаге он «посещает» одну вершину и пытается уменьшить метки. Работа алгоритма завершается, когда все вершены посещены.

Инициализация (начальная расстановка отметок). Полагаем

$$\lambda_a = 0,\ \lambda_x = \infty\ .$$

Символ ∞ отражает тот факт, что расстояния от a до других вершин пока неизвестны. Все отметки объявляем предварительными.

Шаг алгоритма (исправление отметок). Из еще не просмотренных вершин выбираем вершину x, имеющую минимальную метку λ_x. Для каждой вершины y, смежной с x и

имеющей предварительную отметку λ_y, исправляем отметку λ_y по правилу

$$\lambda_y^{'} = \min\{\lambda_y,\ \lambda_x + l(x, y)\}.$$

Объявляем отметку вершины x окончательной и повторяем шаг алгоритма для следующей вершины. Число шагов равно числу вершин графа.

Для графа, изображенного на рис. 3.10, показаны окончательные отметки и кратчайшая цепь.

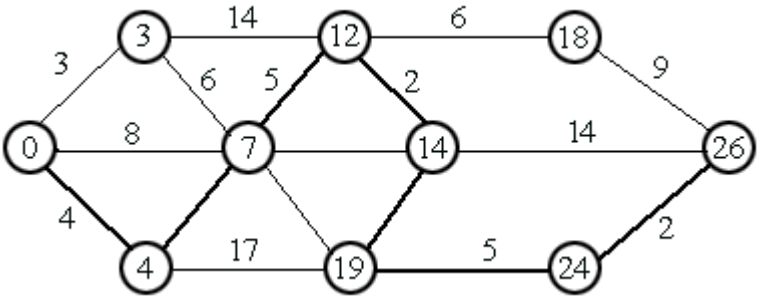


Рис. 3.10. Окончательные отметки и кратчайшая цепь в нагруженном графе

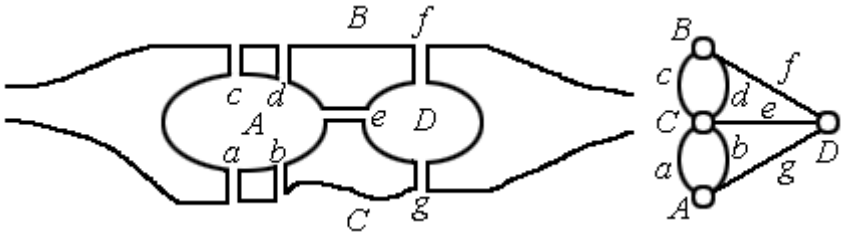
3.5. Цикломатика графов

3.5.1. Эйлеровы циклы и цепи

Началом математической теории графов послужила *задача о Кёнигсбергских мостах*, поставленная Леонардом Эйлером (1707 – 1783).

Кёнигсберг (теперь Калининград) расположен на обоих берегах реки Преголя и на двух островах этой реки. Берега реки и два острова соединены семью мостами, как показано на рис.3.11(а). Эйлер в 1736 г. поставил вопрос: можно ли, начав с

некоторой точки, совершить прогулку и вернуться в исходную точку, пройдя по каждому мосту ровно один раз. Если каждый берег и острова считать вершинами графа, а каждый мост – ребром, то карту города можно представить в виде мультиграфа (рис. 3.11(б)).



а б

Рис. 3.11 Карта города (а) и эквивалентный ей граф (б)

Эйлеровым циклом (*цепью*) в мультиграфе G называется цикл (цепь), содержащий все ребра мультиграфа по одному разу. Граф, имеющий эйлеров цикл, также будем называть э*йлеровым* . Граф, содержащий эйлерову цепь, называется *полуэйлеровым.* Эйлеров граф можно нарисовать на бумаге, не отрывая от нее карандаша. Замкнутые линии, которые можно получить, не отрывая карандаша от бумаги, проходя при этом каждый участок один раз, называются *уникурсальными*. С этими линиями связана задача минимизации холостого хода пера графопостроителя.

Эйлер установил, что задача кёнигсбергских мостов неразрешима, и этот результат ознаменовал возникновение теории графов.

Т е о р е м а 3 . 4 . *Мультиграф обладает эйлеровым циклом тогда и только тогда, когда он связный и все его локальные степени четны.*

Д о к а з а т е л ь с т в о . Н е о б х о д и м о с т ь . Пусть мультиграф G обладает эйлеровым циклом. Тогда связность мультиграфа очевидна, так как в эйлеров цикл входят все ребра, а значит, и все вершины. Следовательно, любые две вершины соединены цепью. Далее, каждый раз, когда эйлеров цикл проходит

через какую-то вершину, он должен войти в нее по одному ребру и выйти по другому, поэтому условие четности степеней вершин также необходимо.

Достаточность докажем индукцией по числу ребер m мультиграфа. При $m=2$ теорема справедлива. Пусть утверждение теоремы верно для всех мультиграфов с числом ребер, не превосходящем m. Для мультиграфа с числом ребер $m+1$ рассмотрим произвольный простой цикл. Хотя бы один такой цикл обязательно существует для графов с четными степенями вершин. Обозначим его μ'. Далее удалим из G все пройденные ребра. Получим мультиграф G', в котором все вершины попрежнему имеют четные степени, но он будет несвязным. Пусть G_1', ..., G_p' – компоненты связности G'. Каждая из этих компонент представляет собой связный мультиграф с четными степенями и с числом ребер, меньшим, чем $m+1$, поэтому по предположению индукции она обладает эйлеровым циклом. Обозначим эйлеровы циклы компонент μ_1', ..., μ_p' и присоединим их к циклу μ'. В результате получим эйлеров цикл мультиграфа G.▪

Следствия.

1°. Связный мультиграф, имеющий ровно две вершины с нечетной степенью, является полуэйлеровым графом.

Действительно, добавим к мультиграфу ребро, соединяющее вершины с нечетной степенью. В результате степени всех вершин станут четными. Построим в новом графе эйлеров цикл, а затем удалим добавленное ребро: цикл разорвется и станет цепью. Эта цепь будет начинаться и заканчиваться в вершинах нечетной степени.

2°. В общем случае число вершин мультиграфа, имеющих нечетную степень, всегда четно. Если мультиграф имеет $2\cdot s$ таких вершин, то все его ребра можно включить в s цепей. Другими словами, мультиграф можно нарисовать, $s-1$ раз отрывая карандаш от бумаги. Обоснование этого утверждения аналогично предыдущему.

3.5.2. Цикломатическое число графа

Ребро графа $G=(X,\ U)$, через которое проходит хотя бы один цикл, называется *цикловым*. Ребро, которое не входит ни в один цикл, называется *перешейком* или *мостом*. Например, в графе, изображенном на рис. 3.12, ребра u_1 и u_2 – перешейки, а остальные ребра цикловые.

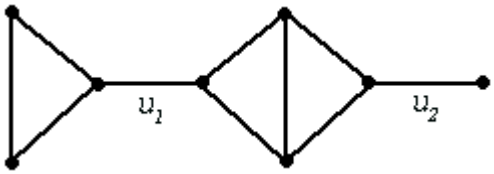


Рис. 3.12. Пример разбиения ребер графа на цикловые и перешейки

Справедливо следующее очевидное утверждение: при удалении из связного графа циклового ребра он остается связным; при удалении из связного графа перешейка граф распадается на две компоненты. Из этого утверждения следует, что при удалении из связного графа циклового ребра число связных компонент не изменяется. При удалении перешейка число связных компонент увеличивается на единицу.

Рассмотрим $(n,\ m)$-граф G, имеющий p компонентов связности. Величина

$$\rho = n - p$$

называется *коцикломатическим числом* графа. Оно равно общему числу ребер в остовах каждой из p связных компонент графа G. *Цикломатическим числом* (*дефектом* или *первым числом Бетти*) называется величина

$$\lambda = m - \rho = m - n + p.$$

Теорема 3.5. $\forall G:\ \lambda \geq 0.$

Доказательство. Будем удалять из графа по одному ребру и следить за изменением величины λ. Параметры исход-

ного графа обозначим m, n, p, а после удаления ребра – m', n', p'. В процессе удаления ребер возможны две ситуации:

1°. Удаляемое ребро – цикловое. Тогда

$$m' = m-1,\ n' = n,\ p' = p;\ \lambda' = m' - n' + p' = \lambda - 1.$$

2°. Удаляемое ребро – перешеек. В этом случае

$$m' = m-1,\ n' = n,\ p' = p+1;\ \lambda' = m' - n' + p' = \lambda.$$

Итак, при удалении ребра величина λ либо не изменяется, либо уменьшается на единицу. После удаления всех ребер получим пустой граф, для которого $m_0 = 0$, $n_0 = n$, $p_0 = n$, то есть $\lambda_0 = 0$. Следовательно, в исходном графе $\lambda \geq 0$.▪

Из теоремы следует, что при $\lambda > 0$ в графе имеется, по крайней мере, один цикл.

3.5.3. Понятие о гамильтоновых циклах

Простой цикл в графе G, проходящий через каждую вершину графа один и только один раз, называется *гамильтоновым*.

Распространенная интерпретация задачи о гамильтоновых циклах состоит в следующем. Обед накрыт на круглом столе. Среди гостей некоторые являются друзьями. При каких условиях можно рассадить всех так, чтобы по обе стороны каждого из присутствующих сидели его друзья?

Подобная задача сформулирована и решена Киркманом. Одиннадцать министров ежедневно садятся за круглый стол. Как их рассаживать, если желательно, чтобы каждый из них имел на каждом заседании новых соседей? Сколько дней это может продолжаться?

Эта задача сводится к отысканию наибольшего числа гамильтоновых циклов без общих ребер в полном графе с одиннадцатью вершинами $a, b, c, d, e, f, g, h, i, j, k$. В данном случае существует только пять циклов без общих ребер:

$$a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ a,$$
$$a\ c\ e\ b\ g\ d\ i\ f\ k\ h\ j\ a,$$
$$a\ e\ g\ c\ i\ b\ k\ d\ j\ f\ h\ a,$$

$$a\,g\,i\,e\,k\,c\,j\,b\,h\,d\,f\,a,$$
$$a\,i\,k\,g\,j\,e\,h\,c\,f\,b\,d\,a.$$

Эти циклы получаются вращением линии, проведенной на рис. 3.13 в направлении стрелок. В более общем случае $2n+1$ вершин можно найти n гамильтоновых циклов без общих ребер.

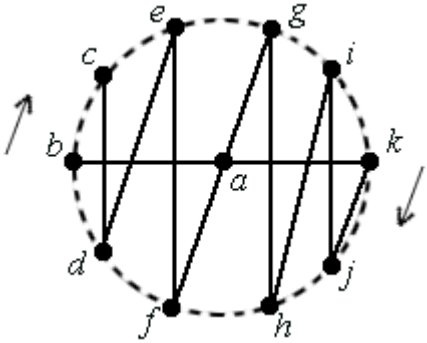


Рис. 3.13. Гамильтоновы циклы в задаче Киркмана

В применениях графов к играм вершины соответствуют различным позициям. Таким образом, существование гамильтонова цикла равносильно существованию циклической последовательности ходов, содержащей каждую позицию по одному разу. Примером является известная *задача о шахматном коне*: можно ли, начиная из произвольного поля на доске, ходить конем в такой последовательности, чтобы пройти через каждое из 64 полей и вернуться в исходное? Эта задача имеет несколько вариантов решений.

Гамильтоновой цепью в графе называется простая цепь, проходящая через все вершины по одному разу.

Если в графе не существует гамильтоновых циклов, то можно искать сумму непересекающихся простых циклов, проходящих через все вершины.

В орграфах можно искать орциклы, проходящие через каждую вершину по одному разу.

К гамильтоновым циклам относится так называемая *задача о коммивояжере*. Район, который должен посетить коммивояжер, содержит какое-то количество городов. Расстояния между ними известны, и нужно найти кратчайшую дорогу, про-

ходящую через все пункты и возвращающуюся в исходный. Эта задача имеет ряд приложений в исследовании операций, например в вопросах о наиболее эффективном использовании подвижного состава или оборудования.

В задаче о коммивояжере города можно представить как вершины графа G, в котором каждой паре вершин приписывается расстояние $\mu(a, b)$. Если вершины не инцидентны, то полагают $\mu(a, b) = \infty$. Тогда задача состоит в том, чтобы найти такой гамильтонов цикл P, для которого сумма

$$\mu(P) = \sum_{i=1}^{n} \mu(a_{i-1}, a_i)$$

минимальна. Так как обычно речь идет только о конечном числе вершин, задача может быть решена перебором. Однако, никакого эффективного алгоритма решения этой задачи до сих пор не известно. Имеются некоторые частные схемы для отдельных случаев. Например, частную задачу определения кратчайшей воздушной линии, соединяющей все столицы штатов в США, просчитали до конца Данциг, Фалкерсон и Джонсон.

В отличие от эйлеровых циклов критерий существования гамильтоновых циклов не известен. Более того, иногда даже для конкретных графов бывает очень трудно решить, можно ли найти такой цикл.

3.5.3. Пространство циклов графа

Рассмотрим множество G_X всех графов с множеством вершин X. Буквой O будем обозначать пустой граф из этого множества: $O = G(X, \varnothing)$.

Для графов $G_1 = (X, U_1)$ и $G_2 = (X, U_2)$ из G_X определим их *сумму по модулю 2* как граф $G_1 \oplus G_2 = (X, U_1 \oplus U_2)$, где $U_1 \oplus U_2$ обозначает симметрическую разность множеств U_1 и U_2. Иначе говоря, ребро принадлежит графу $G_1 \oplus G_2$ тогда и

только тогда, когда оно принадлежит одному из графов G_1 и G_2. Пример показан на рис. 3.14.

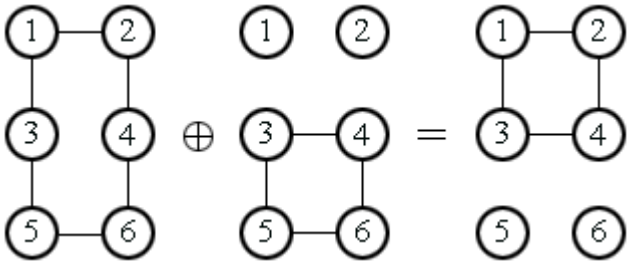


Рис. 3.14

Введенная операция обладает следующими свойствами:

1°. $\forall G_1, G_2 : G_1 \oplus G_2 = G_2 \oplus G_1$.

2°. $\forall G_1, G_2, G_3 : G_1 \oplus (G_2 \oplus G_3) = (G_1 \oplus G_2) \oplus G_3$.

3°. $\forall G : G \oplus O = G$.

4°. $\forall G : G \oplus G = O$.

Отсюда следует, что множество G_X относительно операции $\oplus$ образует *абелеву группу.* Нейтральным элементом («нулем») этой группы служит граф O, а противоположным к каждому графу является сам этот граф. Уравнение $G \oplus X = H$ с неизвестным X и заданными графами G и H имеет единственное решение $X = G \oplus H$. Благодаря свойству ассоциативности в выражении вида $G_1 \oplus G_2 \oplus ... \oplus G_k$ можно не использовать скобки для указания порядка действий. Очевидно, что ребро принадлежит графу $G_1 \oplus G_2 \oplus ... \oplus G_k$ тогда и только тогда, когда оно принадлежит нечетному количеству графов $G_1, G_2, ..., G_k$.

Рассмотрим бинарное множество {0, 1}. Оно является полем относительно операций умножения и сложения по модулю 2. Определим операцию умножения элементов этого поля на графы:

$$\forall G : 0 \cdot G = O,\ 1 \cdot G = G.$$

Множество G_X с введенными операциями сложения графов и умножения на элементы поля является линейным векторным пространством.

Зафиксируем некоторый граф $G \in G_X$ и рассмотрим множество всех его остовных подграфов, которое будем обозначать $S[G]$. Это множество состоит из 2^m элементов, среди них сам граф G и граф O. Оно замкнуто относительно сложения графов и умножения на элементы поля, следовательно, является подпространством пространства G_X. Его называют *пространством остовных подграфов* графа G.

Остовному подграфу $G' = (X, U')$ можно поставить в соответствие двоичное слово $\tilde{\alpha} = \alpha_1 ... \alpha_m$, в котором нули указывают, какие ребра удалены, а единицы – какие оставлены:

$$G' \leftrightarrow \alpha(G') = \alpha_1 \ ... \ \alpha_m, \ \alpha_i = \begin{cases} 1, \text{ если } u_i \in U', \\ 0, \text{ если } u_i \notin U', \ i = 1, ..., \ m. \end{cases}$$

Циклом будем называть граф, у которого одна компонента связности является простым циклом, а остальные – изолированными вершинами. В пространстве остовных подграфов графа G выделим подпространство, содержащее все циклы графа G. Остовный подграф, у которого степени всех вершин четны, называется *квазициклом*.

Отметим, что всякий цикл является квазициклом, в том числе и пустой граф. Покажем, что множество квазициклов замкнуто относительно операции сложения.

Лемма 2. *Сумма двух квазициклов есть квазицикл.*

Доказательство. Пусть C_1 и C_2 – квазициклы. Рассмотрим произвольную вершину $x \in X$, и пусть ее степени в C_1 и C_2 равны соответственно s_1 и s_2. Тогда степень вершины x в графе $C_1 \oplus C_2$ будет равна $s = s_1 + s_2 - 2s_{1,2}$, где $s_{1,2}$ – число вершин, с которыми x смежна в обоих графах C_1 и C_2. Отсюда видно, что s четно, если четны s_1 и s_2. ▪

Из леммы следует, что множество квазициклов является линейным векторным пространством над полем {0, 1}, которое называется *пространством циклов* графа G. Обозначим это пространство через $C[G]$. Очевидно, что $C[G]$ является подпространством векторного пространства остовных подграфов.

Компактное представление пространства дает его базис. Если выписать все простые циклы графа G, то в большинстве случаев они не образуют базис, так как некоторые из этих циклов могут быть суммами других. Построить базис пространства $C[G]$, состоящий из простых циклов, можно следующим образом. Выберем в графе G какой-нибудь остов (каркас) T. Пусть $h_1, ..., h_s$ – все хорды графа G. Если добавить к T хорду h_i, то в графе образуется единственный цикл C_i. Таким образом, получим семейство из s циклов, которые называются *фундаментальными (базисными*) циклами относительно остова T.

Теорема 3.6. *Множество всех фундаментальных циклов относительно любого остова T графа G образует базис пространства циклов этого графа.*

Доказательство.

Зафиксируем некоторый остов T и рассмотрим фундаментальные циклы $C_1, ..., C_s$ относительно этого остова. В каждом из этих циклов имеется хорда h_i, принадлежащая циклу C_i и не принадлежащая никакому из остальных. Поэтому при сложении этого цикла с другими фундаментальными циклами эта хорда будет присутствовать в суммарном графе. Следовательно, сумма различных фундаментальных циклов никогда не будет пустым графом, то есть фундаментальные циклы линейно независимы.

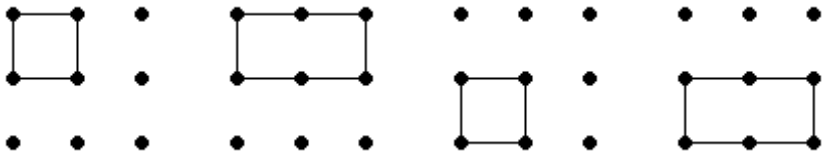
Покажем теперь, что любой квазицикл C графа G является суммой фундаментальных циклов. Пусть $h_{i_1}, ..., h_{i_k}$ – все ребра C, не принадлежащие T (хорды графа C). Рассмотрим квазицикл $C' = C \oplus C_{i_1} \oplus ... \oplus C_{i_k}$. Каждое из ребер h_{i_j} ($j = 1, ..., k$) входит ровно в два слагаемых этой суммы – в C и в C_{i_j}. Следовательно, при сложении эти ребра уничтожатся. Все остальные

ребра, присутствующих в графах-слагаемых, принадлежат T. Значит, $C^{'}$ – подграф графа T. Поскольку в T нет циклов, то $C^{'} = O$. Отсюда $C = C_{i_1} \oplus ... \oplus C_{i_k}$.▪

Из этой теоремы следует, что размерность пространства циклов графа равна числу ребер, не входящих в его остов, то есть числу хорд. Так как остов содержит $n - p$ ребер, то эта размерность равна $m - n + p$. Таким образом, *размерность пространства циклов графа равна его цикломатическому числу.*

Пример 3.4. Построим систему базисных циклов для графа, представленного на рис. 3.8.

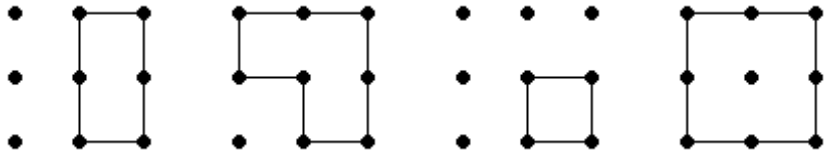
Выделим остовное дерево T_1 и присоединяя к нему по очереди хорды u_1, u_2, u_3 и u_4. В результате получим 4 базисных цикла (рис. 3.15).



C_1 C_2 C_3 C_4

Рис. 3.15. Базисные циклы графа G для дерева T_1

Для дерева T_2 получим другую систему базисных циклов (рис. 3.16).



$\tilde{C}_1$ $\tilde{C}_2$ $\tilde{C}_3$ $\tilde{C}_4$

Рис. 3.16. Базисные циклы графа G для дерева T_2

Циклы одной из систем можно выразить как линейные комбинации циклов из другой системы. В данном случае прямое и обратное преобразования имеют вид:

$$\begin{cases} \tilde{C}_1 = C_1 \oplus C_2 \oplus C_3 \oplus C_4, \\ \tilde{C}_2 = C_2 \oplus C_3 \oplus C_4, \\ \tilde{C}_3 = C_3 \oplus C_4, \\ \tilde{C}_4 = C_2 \oplus C_4; \end{cases} \quad \begin{cases} C_1 = \tilde{C}_1 \oplus \tilde{C}_2, \\ C_2 = \tilde{C}_2 \oplus \tilde{C}_3, \\ C_3 = \tilde{C}_2 \oplus \tilde{C}_4, \\ C_4 = \tilde{C}_2 \oplus \tilde{C}_3 \oplus \tilde{C}_4. \end{cases}$$

3.6. Потоки в сетях

3.6.1. Постановка задачи о максимальном потоке

Одной из наиболее интересных и важных задач теории графов является задача определения максимального потока, протекающего от некоторой вершины s графа (источника) к некоторой конечной вершине t (стоку). При этом каждой дуге $u_{ij} = (x_i, x_j)$ графа G приписана некоторая пропускная способность $q_{ij}(u_{ij})$, и эта пропускная способность определяет наибольшее значение потока, который может протекать по данной дуге. Эта задача и ее варианты могут возникать во многих практических приложениях, например при определении максимальной интенсивности транспортного потока между двумя пунктами на карте дорог, представленной графом. В этом примере решение задачи о максимальном потоке укажет также ту часть сети дорог, которая «насыщена» и образует «узкое место» в отношении потока между двумя указанными концевыми пунктами.

Метод решения *задачи о максимальном потоке* (от s к t) предложен Фордом и Фалкерсоном, и их «техника пометок» составляет основу других алгоритмов решения многочисленных задач, являющимися простыми обобщениями или расширениями указанной задачи. Рассмотрим возможные варианты задачи о максимальном потоке.

Задача нахождения допустимого потока минимальной стоимости. Допустим, что каждой дуге

графа приписана не только пропускная способность q_{ij}, дающая верхнюю границу потока через дугу $u_{ij} = (x_i, x_j)$, но также пропускная способность r_{ij}, дающая нижнюю границу потока через эту дугу. В общем случае может существовать много потоков, удовлетворяющим требованиям о максимальной и минимальной пропускных способностях дуг. Если в дополнение к пропускным способностям заданы также стоимости единицы потока, протекающего по дуге, то возникает задача нахождения допустимого потока минимальной стоимости.

Задача о многопродуктовом потоке. Эта задача возникает, если в сети имеется несколько источников и стоков, между которыми протекают потоки различных продуктов. В этой задаче пропускная способность $q_{ij}(u_{ij})$ является ограничением для суммы всех потоков всех видов продукции через эту дугу.

Задача о потоках с выигрышами. Во всех рассмотренных выше случаях неявно допускалось, что поток на входе дуги такой же, как и на выходе. Если рассмотреть граф, в котором выходной поток дуги равен ее входному потоку, умноженному на некоторое неотрицательное число, то задачу о максимальном потоке называют задачей о потоках с выигрышами. В такой задаче потоки могут «порождаться» и «поглощаться» самим графом, так что поток, входящий в s, и поток, покидающий t, могут изменяться совершенно независимо.

3.6.2. Определение сети, потока и разреза

Граф, некоторые вершины которого выделены, называется *сетью*. Выделенные вершины называются *полюсами* сети. Например, дерево с корнем можно рассматривать как однополюсную сеть.

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

(k, l)-*полюсником* называется сеть с $(k+l)$ полюсами, разбитыми на два класса: k входных и l выходных полюсов. (1,1)-полюсник называется также *двухполюсной сетью.*

Далее будут рассматриваться только двухполюсные сети, которые будем называть просто сетями. Будем называть также цепью (без указания концов) элементарную цепь между полюсами сети. В противном случае будем указывать концы цепи и называть ее *цепочкой.*

Транспортной называется двухполюсная сеть, в которой каждой дуге u приписано целое неотрицательное число $c(u)$, называемое пропускной способностью дуги.

Можно дать различные интерпретации транспортной сети. Пусть, например, в полюсе s имеется неограниченный запас некоторого продукта и нужно организовать доставку этого продукта в полюс t по сети путей сообщения с некоторыми промежуточными вершинами. В этом случае пропускная способность дуги – количество груза, которое можно перевезти в единицу времени по данной дуге. Тогда возникает задача: организовать перевозки по сети таким образом, чтобы, не превышая пропускных способностей дуг, перевозить из s в t максимальное количество груза в единицу времени.

Для каждой вершины x сети S обозначим через U_x^+ множество всех дуг, входящих в x, а через U_x^- – выходящих из x. Для источника s и стока t имеем $U_s^+ = U_t^- = \varnothing$.

Потоком в сети S называется целочисленная функция $\varphi(u)$, определенная на дугах сети и удовлетворяющая условиям:

1) $0 \le \varphi(u) \le c(u)$, $(u \in U)$;

2) $\sum_{u \in U_x^+} \varphi(u) - \sum_{u \in U_x^-} \varphi(u) = 0$, ($x \in X$, $x \ne s$, $x \ne t$).

Второе условие называют *уравнением сохранения*. Оно представляет собой для каждой внутренней вершины сети закон Кирхгофа, согласно которому сумма значений потока по ребрам, входящим в вершину, равна сумме значений потока по ребрам, исходящим из вершины.

На рис. 3.17 приведен пример сети $S = (X, U, c(u), \varphi(u))$, в которой каждой дуге u_i приписана двойка $(c(u_i), \varphi(u_i))$.

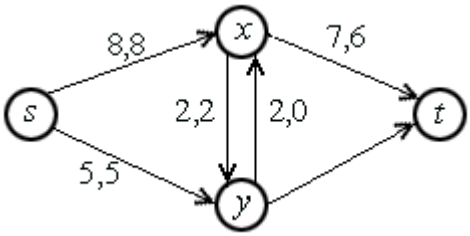


Рис. 3.17. Пример транспортной сети

Если сложить уравнения сохранения для всех вершин, то останутся только члены, соответствующие дугам U_s^- и U_t^+:

$$\sum_{u \in U_s^-} \varphi(u) - \sum_{u \in U_t^+} \varphi(u) = 0.$$

Таким образом, для любого потока величина груза, выходящего из источника s равна величине груза, прибывающего в сток t. Эту величину обозначают Φ и называют *величиной потока*:

$$\Phi = \sum_{u \in U_s^-} \varphi(u) = \sum_{u \in U_t^+} \varphi(u).$$

Поток в сети, имеющий наибольшую величину, называется *максимальным.*

Основная задача состоит в нахождении максимального потока для данной транспортной сети. Для ее решения используют специальные подмножества дуг сети, называемые *разрезами* или *сечениями*.

Разрезом сети называется множество ребер, при удалении которого сеть становится несвязной, причем полюсы попадают в разные компоненты связности. Очевидно, что любая цепь проходит через одно ребро разреза.

Пусть $A \subseteq X$ – некоторое подмножество вершин сети, для которого полюс $s \in A$, а другой полюс $t \notin A$. Обозначим $\overline{A} = X \setminus A$ – дополнение множества A до множества X. Тогда $s \notin \overline{A}$, а $t \in \overline{A}$. Множество дуг сети, имеющих начало в A, а ко-

нец в $\bar{A}$, называется *разрезом, порождаемым множеством вершин* A, и обозначается $(A, \bar{A})$. Например, для сети на рис. 3.17 выберем $A = \{s, x\}$. Тогда $\bar{A} = \{y, t\}$. Имеем разрез

$$(A, \bar{A}) = \{(x, t), (s, y), (x, y)\}.$$

Ребро разреза называется *прямым*, если оно ориентировано слева направо, и *обратным* – в противном случае. Сумма пропускных способностей всех прямых дуг разреза называется *пропускной способностью разреза* и обозначается $c(A, \bar{A})$. Разрезы, которые обладают наименьшей возможной пропускной способностью, называются *минимальными.*

3.6.3. Определение максимального потока в транспортной сети

В 1955 г. Форд и Фалкерсон доказали следующую теорему о максимальном потоке и минимальном разрезе.

Теорема 3.7. *Максимальная величина потока в сети равна пропускной способности любого минимального разреза.*

Доказательство. Докажем сначала, что величина любого потока не превосходит пропускной способности любого разреза: $\Phi \le c(A, \bar{A})$. Просуммируем уравнения сохранения по всем вершинам $x \in A$. Получим

$$\sum_{u \in A} \varepsilon_u \cdot \varphi(u) = 0, \qquad (3.4)$$

где значения коэффициентов $\varepsilon_u \in \{-1, 0, 1\}$ и зависят от расположения концов дуги $u = (x, y)$ относительно множеств A и $\bar{A}$. На рис. 3.18 показаны шесть возможных вариантов такого расположения.

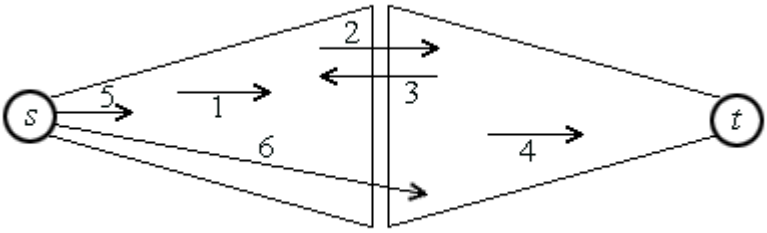


Рис. 3.18. Возможные расположения концов
произвольной дуги $u = (x, y)$

1°. $x \in A,\ x \neq s,\ y \in A$. В этом случае $\varphi(u)$ в ходит в складываемые уравнения дважды: со знаком "+" для вершины y и со знаком "–" для вершины x. Следовательно, $\varepsilon_u = 0$.

2°. $x \in A,\ x \neq s,\ y \in \bar{A}$. В этом случае $\varphi(u)$ в ходит только в уравнение сохранения для вершины x, поэтому $\varepsilon_u = -1$.

3°. $x \in \bar{A},\ y \in A$. В этом случае $\varphi(u)$ в ходит только в уравнение сохранения для вершины y, поэтому $\varepsilon_u = 1$.

4°. $x \in \bar{A},\ y \in \bar{A}$. В этом случае $\varepsilon_u = 0$, т. к. $\varphi(u)$ нет в складываемых уравнениях.

5°. $x = s,\ y \in A$. Результат аналогичен п. 3°.

6°. $x = s,\ y \in \bar{A}$. Результат аналогичен п. 4°.

Для дуг шестого типа в сумму (3) добавим и вычтем $\varphi(u)$. Тогда $\varepsilon_u = 1$ для дуг, выходящих из источника ($u \in U_s^-$), и дуг, идущих против разреза $(u \in (\bar{A}, A))$; $\varepsilon_u = -1$ для дуг разреза $(u \in (A, \bar{A}))$; $\varepsilon_u = 0$ для остальных дуг.

Перепишем уравнение (3.4) в виде

$$\sum_{u \in U_s^-} \varphi(u) - \sum_{u \in (A, \bar{A})} \varphi(u) + \sum_{u \in (\bar{A}, A)} \varphi(u) = 0,$$

откуда для любого потока и любого разреза

$$\Phi = \sum_{u \in U_s^-} \varphi(u) = \sum_{u \in (A,\, \bar{A})} \varphi(u) - \sum_{u \in (\bar{A},\, A)} \varphi(u) \le \sum_{u \in (A,\, \bar{A})} \varphi(u) \le$$

$$\le \sum_{u \in (A,\, \bar{A})} c(u) = c(A,\, \bar{A}).$$

Отсюда следует, что максимальное значение потока в сети равно минимальному значению пропускных способностей разрезов этой сети:

$$\max_{\varphi} \Phi = \min_{A} c(A,\, \bar{A}) .\blacksquare \qquad (3.5)$$

Анализ уравнения (3.5) показывает, что величина Φ потока φ совпадает с пропускной способностью разреза $(A,\, \bar{A})$ тогда и только тогда, когда выполняется условие:

$$\varphi(u) = \begin{cases} c(u), \ \forall u \in (A,\, \bar{A}), \\ 0, \ \forall u \notin (A,\, \bar{A}). \end{cases} \qquad (3.6))$$

Алгоритм, который приводится ниже, направлен на построение такого разреза, для которого выполняется условие (3.6). Если удается построить такой разрез, то задача решена, если нет, то производится увеличение потока с помощью прибавляющих цепей.

Дадим определение прибавляющей цепи. Рассмотрим в сети цепь из источника в сток, то есть последовательность вершин

$$s = x_0,\ x_1,\ ...,\ x_i,\ x_{i+1,}\ ...,\ x_{k-1},\ x_k = t$$

такую, что между вершинами x_i и x_{i+1} $(i = 0,\, 1,\, ...,\, k-1)$ есть дуга , которая может оказаться *прямой* $(x_i,\, x_{i+1})$ или *обратной* $(x_{i+1},\, x_i)$.

Пусть в сети задан поток φ. Цепь из s в t называется *прибавляющей*, если для каждой ее прямой дуги выполняется строгое неравенство $\varphi(u) < c(u)$, а для каждой обратной дуги – строгое неравенство $\varphi(u) > 0$.

Предположим, что для потока φ удалось найти прибавляющую цепь. Тогда увеличивая φ на максимально возможное число n единиц на прямых дугах (с обеспечением условия $\varphi(u) \le c(u)$) и уменьшая на столько же единиц на обратных ду-

гах (с обеспечением условия $\varphi(u) \geq 0$), получим новый поток, величина которого на n единиц больше, чем величина φ.

Алгоритм определения максимального потока в транспортной сети состоит в последовательном просмотре вершин сети и присвоении им отметок. На каждом шаге алгоритма любая вершина находится в одном из трех состояний: а) не помечена; б) помечена, но не просмотрена; в) помечена и просмотрена.

1°. Пометим источник s любой отметкой, например, звездочкой *. После этого вершина s помечена, но не просмотрена. Остальные вершины не помечены.

2°. Берем очередную помеченную, но не просмотренную вершину x. Просматриваем все дуги, инцидентные этой вершине. Если вторая вершины дуги не помечена, то помечаем ее отметкой "x" в следующих двух случаях:

а) дуга выходит из вершины x и поток по ней строго меньше пропускной способности;

б) дуга входит в вершину x и поток по ней строго больше нуля.

Далее вершина x объявляется помеченной и просмотренной, а вершины, получившие при просмотре отметку "x", объявляются помеченными, но не просмотренными.

Пункт 2° циклически повторяется до тех пор, пока не произойдет одно из следующих двух событий:

а) сток t получил отметку, например, "y". Переходим из t в вершину y, по отметке вершины y отыскиваем следующую вершину и т. д. до тех пор, пока не дойдем до вершины s. В результате получаем прибавляющую цепь, с помощью которой увеличиваем текущий поток. Далее стираем отметки всех вершин и повторяем выполнение алгоритма с пункта 1°;

б) процесс расстановки отметок закончился тем, что все помеченные вершины просмотрены, но сток t при этом не помечен. Пусть A – множество помеченных вершин. Так как $t \notin A$, а $s \in A$, то можно определить разрез $(A, \bar{A})$. Для $\forall u \in (A, \bar{A})$, то есть дуги, идущей из помеченной вершины в непомеченную,

$\varphi(u)=c(u)$, иначе другой конец этой дуги был бы помечен. По той же причине для $\forall u \notin (A, \bar{A})$: $\varphi(u)=0$ $\varphi(u)=0$. Следовательно, для построенного потока и разреза $(A, \bar{A})$, образованного помеченными вершинами, выполняются условия (3.6), т. е. имеем максимальный поток.

Пример 3.5. Пусть задан поток в транспортной сети (рис. 3.19).

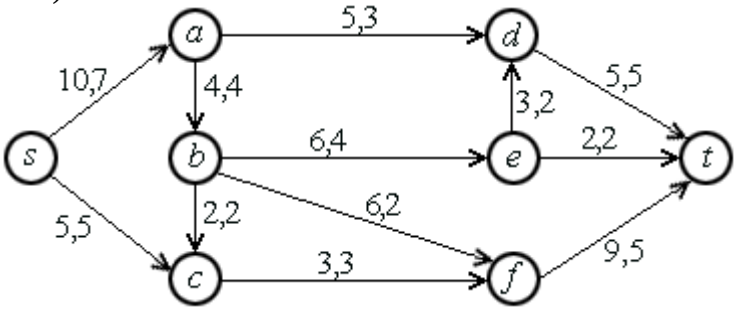


Рис. 3.19

Процесс расстановки отметок показан в таблице 3.1.

Таблица 3.1

Номер шага	Отметки вершин							
	s	*a*	*b*	*c*	*d*	*e*	*f*	*t*
1	*							
2	*	"*s*"						
3	*	"*s*"			"*a*"			
4	*	"*s*"			"*a*"	"*d*"		
5	*	"*s*"	"*e*"		"*a*"	"*d*"		
6	*	"*s*"	"*e*"		"*a*"	"*d*"	"*b*"	
7	*	"*s*"	"*e*"		"*a*"	"*d*"	"*b*"	"*f*"

Поскольку сток получил отметку, то строим прибавляющую цепь (рис. 3.20).

Рис. 3.20

Увеличиваем поток на две единицы на прямых дугах этой цепи и уменьшаем на две единицы на обратных. В результате получаем поток, изображенный на рис. 3.21.

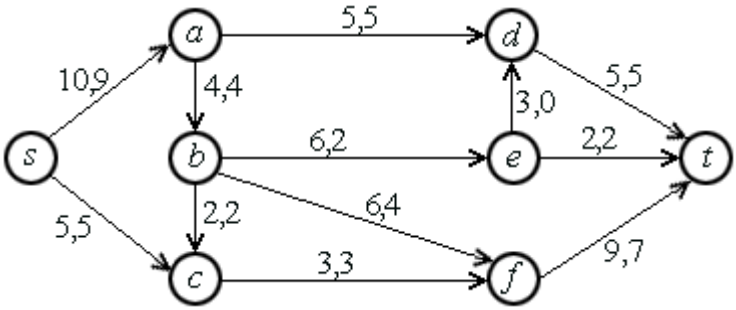


Рис. 3.21

В процессе расстановки отметок для нового потока удается пометить только вершины s и a. Тогда $A=\{s,\ a\}$ – множество помеченных вершин, которое порождает минимальный разрез $(A,\ \bar{A})=\{(s,c),\ (a,b),\ (a,d)\}$. Его пропускная способность

$$c(A,\ \bar{A})=c(s,c)+c(a,b)+c(a,d)=5+4+5=14$$

совпадает с величиной потока

$$\Phi=\sum_{u\in U_s^-}\varphi(u)=c(s,\ a)+c(s,\ c)=9+5=14$$

или

$$\Phi=\sum_{u\in U_t^+}\varphi(u)=c(d,\ t)+c(e,\ t)+c(f,\ t)=5+2+7=14.$$

3.6.4. Теорема Кёнига-Эгервари

Теорема Кёнига-Эгервари вытекает из теоремы Форда-Фалкерсона для транспортных сетей. Она является частным

случаем теоремы Хола о различных представителях в теории графов.

Пусть $A_i \subseteq A\ (i = 1,\ 2,\ ...,\ n)$. Совокупность элементов $\{a_i\}\ (i = 1,\ 2,\ ...,\ n)$ множества A называется *системой различных представителей* семейства множеств $A_i\ \ (i = 1,\ 2,\ ...,\ n)$ или *трансверсалью,* если выполняются два условия:

1) $\forall\ i \in \{1,\ 2,\ ...,\ n\}:\ a_i \in A_i$;

2) $a_i \neq a_j$, если $i \neq j$.

Один из подходов к изучению трансверсалей состоит в исследовании так называемой (0,1)-матрицы, то есть матрицы произвольной размерности $[m \times n]$, составленной из 0 и 1:

$$A = \| a_{ij} \|,\ a_{ij} \in \{0,\ 1\},\ i = 1,\ 2,\ ...,\ m;\ \mathrm{j} = 1,\ 2,\ ...,\ n.$$

Элементы такой матрицы интерпретируют как *разрешенные* ($a_{ij} = 1$) и *запрещенные* ($a_{ij} = 0$).

Множество разрешенных элементов называется *независимым*, если эти элементы расположены в различных строках и столбцах, то есть множество

$$M = \{(i_1,\ j_1),\ ...,\ (i_k,\ j_k)\}$$

независимо, если

1) $a_{i_1, j_1} = ... = a_{i_1, j_1} = 1$;

2) при $s \neq t\ \ i_s \neq i_t,\ \ j_s \neq j_t$.

Наибольшее число разрешенных элементов в матрице A называется *словарным рангом*.

Будем называть строки и столбцы матрицы *рядами* или *линиями*. Совокупность строк и столбцов матрицы, для которой каждый единичный элемент попадает в какой-либо ряд, образуют *покрывающее множество*, то есть множество

$$S = I \bigcup J = \{i_1,\ ...,i_k\} \bigcup \{j_1,\ ...,\ j_k\}$$

покрывающее, если

$$a_{i,j} = 1 \Rightarrow i \in I \text{ или } j \in J .$$

Для любых независимых множеств элементов M и покрывающих множеств рядов S выполняется условие

$$|M| \leq |S|,$$

т.к. в независимое множество может входить не более одного элемента из каждого ряда.

Теорема 3.8. *Словарный ранг (0, 1)-матрицы A равен минимальному числу рядов в покрывающем множестве.*

Доказательство. Для данной (0,1)-матрицы A размерности $[m \times n]$ построим транспортную сеть следующим образом.

Вершины: источник s и сток t; вершины-строки r_1, ..., r_m – по одной для каждой строки матрицы; вершины-столбцы c_1, ..., c_n – по одной для каждого столбца матрицы.

Дуги: источник s соединен дугой с каждой вершиной-строкой; сток t соединен дугой с каждой вершиной-столбцом; вершина-строка r_i соединяется дугой с вершиной-столбцом c_j в том и только том случае, когда $a_{i,j} = 1$.

Пропускные способности каждой дуги равна 1.

Решим для построенной сети задачу о максимальном потоке, применяя алгоритм расстановки отметок. Предположим, что построен максимальный поток, то есть процесс расстановки отметок закончился, но сток t не получил отметки. Обозначим величину максимального потока через f.

Из определения потока следует, что на любой дуге он не превышает ее пропускной способности, то есть может равняться 0 или 1. Пусть $(r_{i_1}, c_{j_1}),..., (r_{i_f}, c_{j_f})$ представляют дуги в «средней» части сети (без дуг, выходящих из источника, и дуг, входящих в сток), поток по которым равен 1. Из уравнения сохранения потока следует, что

$$M = \{(i_1, j_1), ..., (i_f, j_f)\}$$

есть независимое множество элементов матрицы. Действительно, каждая вершина-строка, по построению сети, имеет только

одну входящую дугу (от источника), поэтому для каждой такой вершины сумма значений потока по дугам, входящим в вершину, равна 0 или 1. Следовательно, и сумма значений потока по дугам, выходящим из вершины, также не превышает 1. Значит, для каждой вершины-строки r_{i_k} из (1) поток равен 1 только на одной из выходящих из этой вершины дуг, поэтому все индексы $i_1, ..., i_f$ различны. Аналогичные рассуждения имеют место для индексов $j_1, ..., j_f$.

Из уравнения сохранения для источника и стока имеем, что поток на дугах $(s, r_{i_1}), ..., (s, r_{i_f})$ и $(c_{j_1}, t), ..., (c_{j_f}, t)$ равен 1, а на остальных дугах в «средней» части сети поток равен нулю.

Рассмотрим процедуру отметки вершин сети.

На первом шаге алгоритма при просмотре вершины s все вершины-строки, входящие в множество $\{r_{i_1}, ..., r_{i_f}\}$, отметку не получат, так как поток по дугам, идущим из s в эти вершины равен 1, то есть пропускной способности дуги. Все остальные вершины- строки получат отметку "s", так как поток по дугам, идущим в них от источника, равен 0.

На втором шаге алгоритма в результате просмотра на первом шаге вершин-строк будут помечены вершины-столбцы, в которые ведут из них дуги. В силу максимальности потока, помеченным может оказаться только какой-либо столбец из множества $\{c_{j_1}, ..., c_{j_f}\}$, так как в противном случае происходит «прорыв» в вершину t и возможность увеличения потока, что противоречит максимальности ранее найденного потока.

На третьем шаге при просмотре помеченной вершины-столбца c_{j_k} пометку может получить только одна вершина-строка r_{j_k}, так как из всех дуг, входящих в вершину c_{j_k}, только по дуге (r_{i_k}, c_{j_k}) поток больше 0 (равен 1).

Пусть I – множество номеров непомеченных строк, а J – множество номеров помеченных столбцов. Из предыдущих рассуждений следует, что

$$I \subseteq \{i_1, \ldots, i_f\},\ J \subseteq \{j_1, \ldots, j_f\},$$

причем вершина-строка r_{j_k} является помеченной в том и только в том случае, когда помечена вершина-столбец c_{j_k}. Другими словами, в паре (r_{j_k}, c_{j_k}) либо обе вершины помечены, либо обе не помечены. Отсюда

$$|I| + |J| = f.$$

Так как процесс расстановки отметок остановился, то в «средней» части сети нет дуг, у которых начало помечено, а конец не помечен (это возможно только в вершине-стоке или вершине-источнике). Следовательно, каждая дуга либо имеет начало в множестве I, либо конец в множестве J, то есть покрывающее множество рядов $S = I \cup J$. Тогда

$$|S| = |I \cup J| = |I| + |J| = f.$$

Таким образом, построено независимое множество элементов и покрывающее множество рядов с одинаковым числом элементов.▪

3.7. Сетевые графики

Примерно с 60-х годов получила широкое распространение система сетевого планирования и управления (СПУ), основным элементом которой является сетевой график. *Сетевой график* представляет собой изображение хода проекта при помощи ациклической сети. Под сетью понимается ориентированный граф, в котором выделены две вершины: одна из них называется *началом* и не имеет входящих дуг, другая – *концом*, она не имеет выходящих дуг. Последовательность различных дуг, в которой начало каждой дуги совпадает с концом предыдущей, называется *путем*. Замкнутый путь называется *контуром* или ориентированным циклом. Если в сети нет ориентированных циклов, она называется *ациклической.* Дуги сети изображают отдельные работы, а вершины – события, состоящие в завершении одной или нескольких работ. Каждой дуге в сетевом графике приписано

целое неотрицательное число – продолжительность соответствующей работы.

Обозначим $t_{i,j}$ – продолжительность работы (i, j). Рассмотрим некоторый путь на сетевом графике: $i_1 - i_2 - i_3 - ... - i_{k-1} - i_k$. Длиной пути назовем сумму продолжительностей входящих в него работ: $t_{i_1,i_2} + t_{i_2,i_3} + ... + t_{i_{k-1},i_k}$.

Рассмотрим все пути из начальной вершины в конечную. Путь наибольшей длины называют *критическим*. Длина критического пути $l_{кр}$ – основная характеристика сетевого графика, смысл которой состоит в том, что если каждая работа (i, j) будет начинаться в тот момент, когда произойдет событие i (раньше она начинаться не может), и выполняться точно за время $t_{i,j}$, то вся совокупность работ будет выполнена за время, равное длине критического пути.

Опишем алгоритм для нахождения критического пути в сетевом графике. В процессе работы алгоритма для каждой вершины i рассчитывается величина $t_p(i)$ – максимальная длина пути из начала в вершину i.

1°. *Правильная нумерация сети*. Нумерация вершин сети называется *правильной*, если номер начала любой дуги сети меньше, чем номер ее конца. Правильная нумерация ациклической сети всегда возможна и производится следующим образом. Нумеруем начальную вершину сети нулем и удаляем ее из сети вместе со всеми выходящими из нее дугами. В получающейся сети непременно образуются вершины, не имеющие входящих дуг (это следует из ацикличности), которые назовем вершинами первого ранга и занумеруем их числами 1, 2, … Далее удалим все вершины первого ранга и выходящие из них дуги. Появившиеся вершины без входящих дуг назовем вершинами второго ранга и дадим им очередные номера. Этот процесс продолжается до тех пор, пока не будут занумерованы все вершины сети.

2°. *Расстановка отметок*. Пусть сеть правильно занумерована. Для каждой вершины i вычисляем отметку $t_p(i)$ по следующим правилам:

– полагаем $t_p(0) = 0$;

– просматриваем вершины в порядке их номеров и для j-ой вершины вычисляем $t_p(j)$ по формуле

$$t_p(j) = \max_i [t_p(i) + t_{ij}],$$

где максимум берется по всем вершинам i, имеющим дугу (i, j), направленную в вершину j.

По окончании процесса вычисления отметок величина $l_{кр}$ находится как отметка концевой вершины.

3°. Построение критического пути. Начиная с вершины-конца, последовательно находим дуги (i, j), для которых $t_p(j) - t_p(i) = t_{ij}$. Эти дуги и образуют критический путь.

Параметр $t_p(i)$, вычисляемый при построении критического пути, называется «ранний срок свершения события i». Для каждого события можно рассчитать также поздний срок его свершения $t_n(i)$. Смысл этого параметра состоит в следующем: если событие i «запоздает, однако произойдет не позднее $t_n(i)$, то длина критического пути (время выполнения всего проекта) не изменится.

Метод расчета $t_n(i)$ аналогичен методу расчета $t_p(i)$, если его обратным ходом. Пусть для правильной сети концевая вершина получила номер N. Тогда для каждой вершины i вычисляем отметку $t_n(i)$ по следующим правилам:

– полагаем $t_n(N) = l_{кр}$;

– просматриваем вершины в обратном порядке их номеров и для i-ой вершины вычисляем $t_n(i)$ по формуле

$$t_n(i) = \min_j [t_n(j) - t_{ij}],$$

где минимум берется по всем вершинам j, в которые ведут дуги (i, j) из вершины i.

Резервом времени работы (i, j) называется величина

$$P(i, j) = t_n(j) - t_p(i) - t_{ij}.$$

Задержка в выполнении работы на величину, не превышающую $P(i, j)$, не изменяет длину критического пути, то есть срока выполнения проекта. Резервы времени позволяют разбить всю совокупность работ на более важные и менее важные, что используется для управления выполнением проекта. Работы, лежащие на критическом пути, имеют резервы времени, равные нулю. Задержка в выполнении такой работы на время Δt ровно на такую же величину изменяет время выполнения всего проекта. Поэтому возможны варианты форсирования этих работ за счет работ, имеющих большой резерв времени.

При практическом применении сетевых графиков более удобными по сравнению с резервами времени являются *коэффициенты напряженности работ*, определяемые как отношение длины максимального пути из начала в конец, проходящего через данную работу, к длине критического пути:

$$k_н(i, j) = \frac{t_p(i) + t_{ij} + l_{кр} - t_n(j)}{l_{кр}} = 1 - \frac{P(i, j)}{l_{кр}}.$$

Из этой формулы следует, что $0 < k_н(i, j) \le 1$, причем для работ с нулевым резервом (лежащих на критическом пути) $k_н(i, j) = 1$, а для работ с большим резервом времени – $k_н(i, j) \approx 0$. На основании этого всю совокупность работ делят на зоны: *критическую* с $k_н(i, j) \approx 0{,}9 \div 1$, *подкритическую* с $k_н(i, j) \approx 0{,}6 \div 0{,}9$, и *резервную*.

Рассмотрим процедуру построения сетевого графика по заданной упорядоченности работ $U = \{1, 2, ..., n\}$, в совокупности составляющих некоторый проект.

Предположим, что для каждой работы j ($1 \le j \le n$) указаны ее непосредственные предшественники, то есть множество ра-

бот, выполнение которых является необходимым условием для начала работы *j*.

Процедура построения сетевого графика распадается на несколько этапов.

I. Транзитивное замыкание отношения предшествования. Если работа *i* предшествует работе *j*, а работа *j* предшествует работе *k*, то, очевидно, работа *i* должна завершиться до начала работы *k*. Множество работ называется замкнутым (по транзитивности), если вместе с каждой работой оно содержит и всех ее предшественников. Наименьшее замкнутое множество, содержащее множество работ *M*, называется *транзитивным замыканием M*.

На этапе I для каждого множества предшественников работы *j* строится его транзитивное замыкание. Оно будет состоять из всех работ *i*, для которых можно найти цепочку работ j_1, j_2,…j_p, такую, что *i* предшествует j_1, j_1 предшествует j_2, …, j_p предшествует *j*. Будем называть эти множества *полными предшественниками* работы *j*.

Найдем все различные множества полных предшественников и обозначим их P_1, P_2, ..., P_s. Каждому из этих множеств в сетевом графике будет соответствовать вершина.

II. Построение конституент для множеств полных предшественников. Разбиение множества *U* на непересекающиеся подмножества

$$U = C_1 + C_2 + \ldots + C_t$$

будем называть *разбиением на конституенты,* если каждое из множеств полных предшественников можно представить в виде объединения некоторых конституент. Каждой из конституент в сетевом графике будет соответствовать вершина.

III. Построение сетевого графика. Вершинами сетевого графика служат построенные множества P_i $(1 \le i \le s)$ и C_j, $(1 \le j \le t)$.

Работа k изображается дугой с началом в множестве полных предшественников и ведущей в конституенту, содержащую эту работу.

Сетевой график содержит также пустые дуги, помогающие обеспечить требуемую упорядоченность работ. Эти дуги не соответствуют работам (или можно считать, что они изображают работы с нулевым временем выполнения) и называются *фиктивными*. Для каждого представления множеств P_i в виде суммы C_j

$$P_i = C_{j1} + C_{j2} + \ldots + C_{jq}$$

проводятся фиктивные дуги из всех вершин C_{j1}, C_{j2}, …, C_{jq} в вершину P_i.

IV. Упрощение графа. Граф, построенный на предыдущем этапе можно упростить, удаляя из него некоторые фиктивные дуги (иногда все). Для этого применяются следующие два правила.

1°. *Если фиктивная дуга соединяет вершины, между которыми есть другой путь, то эту дугу следует удалить.*

2°. *Если единственная дуга, выходящая из вершины (входящая в вершину) является фиктивной, то эту дугу следует удалить и слить ее концы в одну вершину.*

Фактическое построение сетевого графика удобно совмещать с упрощениями 1° и 2°. Работы помещаются на сетевой график в порядке их появления в исходной таблице, при этом одновременно производятся упрощения.

Пример 3.6. $U = \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7\}$, то есть весь проект разбит на 7 работ, соотношения между которыми отображены в таблице 3.2.

Таблица 3.2

Работы	1	2	3	4	5	6	7
Предшественники	–	–	2	1,3	1,3	2	5,6

Построим множества полных предшественников:

$$P_1 = \varnothing,\ P_2 = \{2\},\ P_3 = \{1,\ 2,\ 3\},\ P_4 = \{1,\ 2,\ 3,\ 5,\ 6\}.$$

В качестве конституент выберем множества

$$C_1 = \{2\},\ C_2 = \{1,3\},\ C_3 = \{5,\ 6\},\ C_4 = \{4,\ 7\}.$$

Тогда

$$P_1 = \varnothing,\ P_2 = C_1,\ P_3 = C_1 + C_2.$$

Совмещая III и IV этапы, получим сетевой график в окончательном виде, представленном на рис. 3.22.

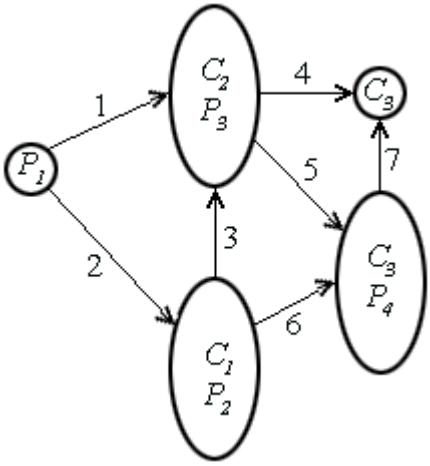


Рис. 3.22

3.8. Введение в теорию кодирования

Вопросы кодирования играют существенную роль в математике. Кодирование позволяет изучение одних объектов сводить к изучению других. Хорошо известно, какую роль сыграло изображение чисел в десятичной системе счисления. Весьма важным в развитии математики было появление метода координат, который позволил кодировать геометрические объекты при помощи аналитических выражений. Однако, здесь средства кодирования являлись вспомогательным аппаратом и не были средством изучения. Совсем другое значение получили коды в связи с изучением управляющих систем. Появилась необходи-

мость систематического исследования в области теории кодирования. Основной круг задач может быть рассмотрен на примере процесса передачи информации по каналу связи.

3.8.1. Самокорректирующиеся коды

Процесс передачи информации по каналу связи можно представить следующим образом. На вход канала поступает длинная последовательность нулей и единиц, разбитая на блоки (например, на байты). На другом конце канала эта последовательность принимается с возможными искажениями: переданная единица может быть принята как ноль и наоборот.

Если уровень помех в канале высокий, то передача данных без каких-либо средств борьбы с помехами становится в принципе невозможной.

Основным средством борьбы с ошибками при передаче по каналу связи является введение избыточности, при этом происходит снижение скорости передачи.

В общем случае процесс передачи информации с введением избыточности можно представить в виде следующей схемы.

1°. Передаваемая информация – длинная последовательность нулей и единиц – разбивается на блоки по k символов.

2°. Каждый блок кодируется двоичным словом из n символов, где $n > k$.

3°. Коды передаются по каналу связи и принимаются на другом его конце с возможными искажениями.

4°. Искаженные коды подвергаются процедуре декодирования, в результате которой восстанавливается первоначальный блок из k символов.

Первым примером такой схемы передачи информации был 7-разрядный код Хемминга, исправляющий единичные ошибки.

Передаваемая информация разбивается на блоки по 4 символа (полубайты), блоки кодируются словами из 7 символов:

$$h_1h_2h_3h_4 \to c_1c_2c_3c_4c_5c_6c_7 .$$

Разряды кодового слова определяются так. Полубайт записывается в разряды 3, 5, 6, 7 (информационные):

$$c_3 = h_1,\ c_5 = h_2,\ c_6 = h_3,\ c_7 = h_4.$$

Остальные разряды кодового слова (контрольные) определяют так, чтобы выполнялись соотношения:

$$c_4 \oplus c_5 \oplus c_6 \oplus c_7 = 0,$$
$$c_2 \oplus c_3 \oplus c_6 \oplus c_7 = 0,$$
$$c_1 \oplus c_3 \oplus c_5 \oplus c_7 = 0.$$

Другими словами, количество единиц в группах разрядов

$$\{4,\ 5,\ 6,\ 7\},\ \{2,\ 3,\ 6,\ 7\} \text{ и } \{1,\ 3,\ 5,\ 7\}$$

должно быть четным. После приема слова $c_1^{'}c_2^{'}c_3^{'}c_4^{'}c_5^{'}c_6^{'}c_7^{'}$ с возможными искажениями подсчитываются суммы

$$s_4 = c_4^{'} \oplus c_5^{'} \oplus c_6^{'} \oplus c_7^{'},$$
$$s_2 = c_2^{'} \oplus c_3^{'} \oplus c_6^{'} \oplus c_7^{'},$$
$$s_1 = c_1^{'} \oplus c_3^{'} \oplus c_5^{'} \oplus c_7^{'}.$$

Если ошибок не было, все три суммы равны нулю. Если был искажен один разряд, то $(s_4 s_2 s_1)_2$ – его номер.

3.8.2. Основные характеристики кода. Алгоритмы декодирования

Кодом длины n называется любое подмножество двоичных слов длины n:

$$C \subseteq \{0,\ 1\}^n.$$

При передаче информации информационные разряды входят в слова $\tilde{\alpha} \in C$. С учетом искажений принимаются любые слова $\tilde{\beta} \in \{0,\ 1\}^n$. После приема слова $\tilde{\beta}$ к нему применяется некоторый *алгоритм декодирования*, сопоставляющий слову $\tilde{\beta}$ слово $\tilde{\alpha}^{'} \in C$. Если оказалось, что $\tilde{\alpha} = \tilde{\alpha}^{'}$, то говорят, что код исправляет ошибки.

Чтобы описать действие помех при передаче и процедуру декодирования, на множестве двоичных слов вводится метрика. *Расстоянием* между двоичными словами называется количество разрядов, в которых эти слова различаются:

$\tilde{\alpha} = \alpha_1\alpha_2 \ldots \alpha_n$, $\tilde{\beta} = \beta_1\beta_2 \ldots \beta_n$, $d(\tilde{\alpha}, \tilde{\beta}) = |\{i \mid \alpha_i \neq \beta_i\}|$.

Определенное таким образом рас стояние удовлетворяет аксиомам метрики:

1°. $d(\tilde{\alpha}, \tilde{\beta}) \geq 0$, $d(\tilde{\alpha}, \tilde{\beta}) = 0 \Leftrightarrow \tilde{\alpha} = \tilde{\beta}$.

2°. $d(\tilde{\alpha}, \tilde{\beta}) = d(\tilde{\beta}, \tilde{\alpha})$.

3°. $d(\tilde{\alpha}, \tilde{\beta}) \leq d(\tilde{\alpha}, \tilde{\gamma}) + d(\tilde{\gamma}, \tilde{\beta})$.

Таким образом, множество двоичных слов становится метрическим пространством и в нем возможны обычные для метрических пространств построения: шар, сфера, ближайшие к данной точки и т. д.

Если передавалось слово $\tilde{\alpha}$, а принято слово $\tilde{\beta}$, то $d(\tilde{\alpha}, \tilde{\beta})$ равно количеству разрядов, искаженных при передаче (число ошибок).

Будем предполагать, что вероятность искажения символа при передаче невелика (в противном случае таким каналом связи нельзя пользоваться). Основываясь на некоторой вероятностной модели можно строго доказать, что наилучшим алгоритмом декодирования при сделанных предположениях будет следующий:

если принято слово $\tilde{\beta}$, то считаем, что передавалось слово $\tilde{\alpha} \in C$, ближайшее к слову $\tilde{\beta}$.

Такой метод принятия решения в условиях неопределенности называют *методом максимального правдоподобия.* В рассматриваемой ситуации это название можно объяснить так: после того как принято слово $\tilde{\beta}$, для каждого слова $\tilde{\alpha}$ вычисляются вероятности того, что слово $\tilde{\beta}$ получено из слова $\tilde{\alpha}$, из этих вероятностей выбирается наибольшая (максимум правдоподобия).

В дальнейшем будем считать, что на приемном конце канала используется именно этот алгоритм декодирования.

Рассмотрим главные параметры, характеризующие код.

1°. *Мощность кода* $|C|$, то есть количество кодовых точек. Если код имеет мощность $|C|$, то блоки информации, которые он кодирует должны иметь длину k, такую, что $2^k \le |C|$, то есть $k \approx \log_2 |C|$. Примем физическую скорость передачи канала за единицу: передача одного двоичного символа занимает одну единицу времени. Применив кодирование, получим, что k двоичных символов передаются за n единиц времени, то есть скорость передачи информации равна

$$\frac{k}{n} = \frac{\log_2 |C|}{n} \text{ бит/ед. вр.}$$

Следовательно, чем больше мощность кода, тем большую скорость передачи информации он обеспечивает.

2°. *Кодовое расстояние*, равное минимальному расстоянию между кодовыми словами*:*

$$d(c) = \min_{\tilde{\alpha},\, \tilde{\beta} \in C,\, \tilde{\alpha} \ne \tilde{\beta}} d(\tilde{\alpha},\, \tilde{\beta}).$$

Роль этого параметра состоит в следующем. Будем говорить, что код *исправляет e ошибок*, если передаваемые слова, в которых искажено не более e разрядов принимаются правильно.

Теорема 3.9. *Если* $d(c) \ge 2 \cdot e + 1$*, то код* C *исправляет* e *ошибок.*

Доказательство. Пусть $\tilde{\alpha} \in C$ – переданное слово, $\tilde{\beta}$ – принятое слово, в котором не более e ошибочных разрядов, то есть $d(\tilde{\alpha},\, \tilde{\beta}) \le e$.

Очевидно, что $\tilde{\alpha}$ и будет ближайшим к $\tilde{\beta}$ кодовым словом, так как для любого другого кодового слова $\tilde{\gamma} \in C$ имеем

$$d(\tilde{\beta},\, \tilde{\gamma}) \ge d(\tilde{\alpha},\, \tilde{\gamma}) - d(\tilde{\alpha},\, \tilde{\beta}) \ge 2 \cdot e + 1 - e = e + 1. \blacksquare$$

Основной задачей теории кодирования является следующая: *для заданных* n*,* e *построить код* $C \subseteq \{0, 1\}^n$ *максимальной мощности такой, что* $d(c) \ge 2 \cdot e + 1$.

3.8.3. Линейные коды

Совокупность $\{0, 1\}^n$ двоичных слов длины n вместе с операцией поразрядного сложения по модулю 2 образует n-мерное векторное пространство над полем $\{0, 1\}$.

Код C называется *линейным*, если он является подпространством этого пространства.

Рассмотрим способы задания линейного кода. Первый способ состоит в задании базиса $\tilde{\gamma}_1, ..., \tilde{\gamma}_k$ из k векторов подпространства размерности k. Тогда

$$C = \{\tilde{x} = \alpha_1\tilde{\gamma}_1 + ... + \alpha_k\tilde{\gamma}_k \mid \alpha_i \in \{0, 1\},\ \text{i=1, ..., } k\}.$$

Второй способ задания подпространства заключается в том, что подпространство определяют как совокупность решений линейной однородной системы уравнений, связывающих координаты векторов

$$\begin{cases} h_{11}x_1 + \ ... + h_{1n}x_n = 0, \\ \qquad\qquad ... \\ h_{n-k1}x_1 + \ ... + h_{n-kn}x_n = 0. \end{cases}$$

Если эти уравнения независимы, то размерность пространства решений равна $n-(n-k)=k$. Бинарная матрица системы H имеет размерность $[(n-k)\times n]$. Тогда

$$C = \{\tilde{x} \mid H\tilde{x} = \tilde{0}\}.$$

Матрица H называется *проверочной* матрицей кода C.

Пример 3.7. Описанный выше 7-разрядный код Хемминга представляет собой линейный код с проверочной матрицей

$$H = \begin{pmatrix} 0\ 0\ 0\ 1\ 1\ 1\ 1 \\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\ 1\ 0\ 1\ 0\ 1\ 0\ 1 \end{pmatrix}.$$

Так как

$$H\tilde{x} = \begin{pmatrix} x_4 + x_5 + x_6 + x_7 \\ x_2 + x_3 + x_6 + x_7 \\ x_1 + x_3 + x_5 + x_7 \end{pmatrix},$$

то из условия $H\tilde{x} = 0$, получаем соотношения, которым удовлетворяют разряды кодовых слов в коде Хэмминга

$$x_4 + x_5 + x_6 + x_7 = 0,$$
$$x_2 + x_3 + x_6 + x_7 = 0,$$
$$x_1 + x_3 + x_5 + x_7 = 0.$$

Эти соотношения представляют собой проверки на четность. Например, первое из них выполняется, когда количество единиц в разрядах 4, 5, 6 и 7.

Будем далее считать, что линейный код C задается проверочной матрицей H. Рассмотрим теперь корректирующие возможности линейного кода, то есть его способность исправлять ошибки. Как было установлено выше, это зависит от параметра $d(C)$. Для линейного кода он вычисляется проще, чем в общем случае. *Весом* двоичного слова $\tilde{\alpha}$ называется число разрядов слова, равных единице:

$$\|\tilde{\alpha}\| = |\{i \mid \alpha_i = 1\}|.$$

Теорема 3.10. *Если C – линейный код, то*

$$d(C) = \min_{\tilde{\alpha} \in C \setminus 0} \|\alpha\|,$$

то есть минимум берется по всем ненулевым словам кода.

Это утверждение очевидным образом следует из следующих двух фактов

$$d(\tilde{\alpha},\ \tilde{\beta}) = \|\tilde{\alpha} + \tilde{\beta}\|;\ \tilde{\alpha} \in C,\ \tilde{\beta} \in C \Rightarrow \tilde{\alpha} + \tilde{\beta} \in C.$$

Теорема 3.11. *Линейный код C исправляет e ошибок, если в его проверочной матрице H любые $2e$ столбцов линейно независимы.*

Доказательство. Обозначим столбцы проверочной матрицы $\tilde{h}_1, ..., \tilde{h}_n$. Условие $H\tilde{x} = 0$ для ненулевого слова $\tilde{x}$, принадлежащего коду, можно записать в виде

$$x_1\tilde{h}_1 + ... + x_1\tilde{h}_n = 0. \qquad (3.7)$$

Количество коэффициентов, равных единице, в последнем выражении равно $\|\tilde{x}\|$, нулевые коэффициенты можно отбросить. Тогда равенство (3.7) представляет собой соотношение линейной зависимости между $\|\tilde{x}\|$ столбцами проверочной матрицы.

При выполнении условия теоремы получаем, что $\|\tilde{x}\| \geq 2e+1$, следовательно, код исправляет e ошибок. ▪

4. Алгебра логики и ее приложения

4.1. Функции алгебры логики

Булевы функции составляют один из классов функциональных систем, на основе которых описывают работу дискретных преобразователей. К другим классам относятся функции k-значной логики, автоматные функции и вычисляемые функции. С каждым из этих классов связывают операции, позволяющие из одних функций данного класса строить другие функции этого же класса. Такими операциями являются: операция суперпозиции, операция обратной связи, операция примитивной рекурсии и μ-операция. В результате получаем функциональные системы с операциями, то есть некоторые классы алгебр. Роль теории функциональных систем в дискретной математике можно сравнить с ролью математического анализа в непрерывной математике.

Функцией алгебры логики n переменных называют функцию $f(x_1, x_2, ..., x_n)$, у которой все переменные и сама функция принимают только два значения. Эти функции называют также *логическими*, б*улевыми* или *переключательными функциями.* Два возможных значения функции и ее аргументов обозначают поразному: И (истина), Л (ложь); 0, 1; да, нет. В дальнейшем будем использовать 0 и 1.

Математическая логика, как самостоятельная область науки, сформировалась в середине XIX века, прежде всего благодаря работам ирландского математикам из г. Корке Джорджа Буля. Первая работа Буля по логике – «Математический анализ логики» вышла в 1847 г. В 1854 г. вышел основной труд Буля – «Исследование законов мысли». Первые применения алгебры логики связаны с решением следующей задачи: выяснить, истинно или ложно сложное высказывание, если известна истинность или ложность составляющих высказываний.

Сложные высказывания образуются из исходных простых при помощи ограниченного набора логических операций (связок).

Будем обозначать простые высказывания буквами $A, B, C, \ldots$. Из них можно составить новые высказывания, например:

«A и B», обозначается $A \& B$;

«A или B», обозначается $A \vee B$;

«не A», обозначается $\bar{A}$;

«если A, то B, обозначается $A \Rightarrow B$.

Пусть буквами A, B, C обозначены такие высказывания:

A – «числовой ряд сходится»;

B – «все члены ряда положительны»;

C – «общий член ряда стремится к нулю».

Образуем некоторые составные высказывания:

$A \Rightarrow C$ – «если числовой ряд сходится, то его общий член стремится к нулю»;

$B \& C$ – «члены ряда положительны и общий член ряда стремится к нулю;

$\bar{C} \Rightarrow \bar{A}$ – «если общий член ряда не стремится к нулю, то ряд расходится».

С помощью логических связок можно образовать довольно сложные высказывания, например, $(A \Rightarrow \bar{C}) \& (\bar{B} \Rightarrow C) \& (A \vee \bar{B})$, относительно которых возникает вопрос их истинности или ложности. Таким образом, каждому высказыванию соответствует булева функция.

Дадим более строгое определение булевой функции. Обозначим через $E = \{0, 1\}$ – булево множество, которое задает область значений функции. Областью определения функции $f(x_1, x_2, \ldots, x_n)$ является совокупность всех выборок $\alpha_1, \alpha_2, \ldots, \alpha_n$, где $\alpha_i \in E$, $i = \overline{1, n}$, то есть декартово произведение $\underbrace{E \times E \times \ldots \times E}_{n \text{ раз}} = E^n$. Таким образом, булевой функцией от n переменных называется отображение

$$f : E^n \to E. \qquad (4.1)$$

В 30-х годах XX века булеву алгебру стали использовать для анализа структуры релейных схем (советский ученый В.И.

Шестаков и американский математик и инженер Клод Шеннон). С развитием вычислительной техники булеву алгебру стали применять в качестве математического аппарата для описания работы дискретных устройств переработки информации. Такое устройство можно представить в виде «черного ящика» с входами и выходами (рис. 4.1).

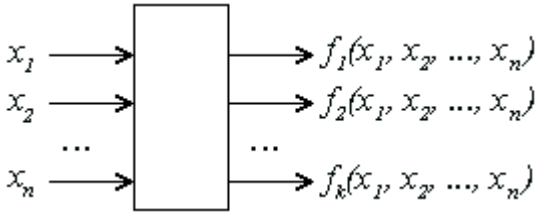


Рис. 4.1

В процессе работы такого устройства каждый вход и каждый выход может находиться в одном из двух состояний: высокий или низкий уровень напряжения, намагниченность той или иной полярности, одно из двух положений переключателя и т.д. Не вдаваясь в подробности конкретной реализации устройства, говорят, что на входы поступает комбинация нулей и единиц и устройство преобразует ее в некоторую комбинацию нулей и единиц на выходе. При помощи двоичных слов информация (буквы, цифры, знаки препинания, служебные знаки) кодируется и поступает на вход устройства, которое производит обработку и на выходе появляется двоичное слово, которое затем можно снова перевести в естественную форму. Каждый выход дискретного преобразователя представляет собой булеву функцию.

Основными способами задания булевых функций являются табличный и аналитический.

Поскольку область определения состоит из конечного числа элементов ($|E^n| = 2^n$), то булеву функцию можно задать при помощи *таблицы истинности* (соответствия), в которой для

каждого набора значений аргументов указывается значение функции (табл. 4.1).

Таблица 4.1. Таблица истинности булевой функции

$x_1x_2...x_{n-1}x_n$	$f(x_1, x_2,..., x_{n-1}, x_n)$
00…00	$f(0, 0, \ldots, 0, 0)$
00…01	$f(0, 0, \ldots, 0, 1)$
00…10	$f(0, 0, \ldots, 1, 0)$
…	…
11…10	$f(1, 1, \ldots, 1, 0)$
11…11	$f(1, 1, \ldots, 1, 1)$

В качестве примера в таблице 4.2 задана функция $f(x_1, x_2, x_3)$, которая равна 1 при нечетном количество переменных, и 0 – в остальных случаях.

Таблица 4.2. Пример задания булевой функции

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

Отметим, что наборы значений аргументов в таблице записывают в естественной форме, то есть i-й по порядку набор представляет собой двоичную запись числа i, i=0, 1, 2, …, 2^n-1.

Обозначим через P_2 систему всех булевых функций от n переменных. Число $p_2(n)$ всех функций из P_2 равно числу перестановок с повторениями значений функции {0, 1} на выборке из 2^n входных наборов переменных, то есть 2^{2^n}. Числа $p_2(n)$ с ростом n быстро растут:

$$p_2(1)=4,\ p_2(2)=16,\ p_2(3)=256,\ p_2(4)=65536,\ \ldots$$

Следовательно, уже при сравнительно небольших значениях n ($n \geq 6$) перебор функций из данного множества становится практически невозможен даже с использованием вычислительной техники. Кроме того, с ростом числа аргументов таблица истинности сильно усложняется. Так, например, уже при не очень большом числе аргументов, скажем при n=10, таблица становится громоздкой (имеет 1024 строки), а при n=20 – практически необозримой. Поэтому используют другие способы задания функции, среди которых основным является аналитический способ, то есть при помощи формул. При этом способе некоторые функции выделяются и называются *элементарными*, а другие функции строят из элементарных с помощью суперпозиции. Такой способ задания функции хорошо известен в математическом анализе. Например, функция $\cos^4\sqrt{x^2-3}$ построена суперпозицией многочлена x^2-3, квадратного корня, косинуса и функции $y=x^4$.

Рассмотрим элементарные функции алгебры логики. Булевы функции одного аргумента представлены в таблице 4.3.

Таблица 4.3. Булевы функции одного аргумента

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Среди этих функций $f_0(x) \equiv 0$ и $f_3(x) \equiv 1$ представляют собой константы, $f_1(x) = x$, а $f_2(x) = \overline{x}$ называется отрицанием (инверсией, логическим НЕ):

$$\overline{x} = \begin{cases} 1, \text{ если } x = 0; \\ 0, \text{ если } x = 1. \end{cases}$$

В таблице 4.4 приведены все 16 функций двух аргументов.

Таблица 4.4. Булевы функции двух аргументов

x_1x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Рассмотрим более подробно эти функции.

К о н с т а н т ы :

$f_0(x_1, x_2) \equiv 0$ – *тождественная ложь*,

$f_{15}(x_1, x_2) \equiv 1$ – *тождественная истина*.

У н а р н ы е ф у н к ц и и .

Функции тождественности:

$f_3(x_1, x_2) = x_1$ или $f_3(x_1, x_2) \square x_1$, $f_5(x_1, x_2) = x_2$ или $f_5(x_1, x_2) \square x_2$.

Отрицание (инверсия, логическое НЕ):

$$f_{10}(x_1, x_2) = \overline{x_1}, \quad f_{12}(x_1, x_2) = \overline{x_2}.$$

Б и н а р н ы е ф у н к ц и и .

Конъюнкция (логическое умножение, логическое И):

$$f_1(x_1, x_2) = x_1 \,\&\, x_2, \; f_1(x_1, x_2) = x_1 \wedge x_2.$$

Читается "x_1 и x_2". Так как эта операция совпадает с операцией умножения в элементарной алгебре, то для конъюнкции используют также обозначение $x_1 \cdot x_2$ или x_1x_2. Конъюнкция

двух высказываний истинна только в том случае, когда истинны оба высказывания.

Дизъюнкция (логическое сложение, логическое ИЛИ):

$$f_7(x_1, x_2) = x_1 \vee x_2 .$$

Читается "x_1 или x_2". Дизъюнкция двух высказываний ложна только в том случае, когда ложны оба высказывания.

Для конъюнкции и дизъюнкции можно записать

$$x_1 \cdot x_2 = \min\{x_1, x_2\},\ x_1 \vee x_2 = \max\{x_1, x_2\}.$$

Импликация (функция логического следования):

$$f_{13}(x_1, x_2) = x_1 \to x_2 \ (\text{импликация от } x_1 \text{ к } x_2).$$

Читается "если x_1, то x_2", "x_1 влечет x_2", "из x_1 следует x_2". x_1 – условие импликации, а x_2 – её заключение. Это важная функция, особенно в математической логике. Её можно рассматривать следующим образом. Из ложного условия можно вывести и истинное и ложное заключение (и это правильно). Если заключение истинно, то его можно вывести как из истинного так и из ложного условия (и это тоже правильно). Импликация ложна только в случае, когда условие x_1 истинно, а заключение x_2 ложно.

Аналогично имеем импликацию от x_2 к x_1:

$$f_{11}(x_1, x_2) = x_2 \to x_1 .$$

Функция неравнозначности (сумма по модулю 2, исключающее ИЛИ):

$$f_6(x_1, x_2) = x_1 \oplus x_2 .$$

Читается "либо x_1, либо x_2" или "x_1 не эквивалентно x_2".

Функция эквивалентности (равнозначности, подобия):

$$f_9(x_1, x_2) = x_1 \square x_2,\ f_9(x_1, x_2) = x_1 \leftrightarrow x_2,\ f_9(x_1, x_2) = x_1 \equiv x_2 .$$

Читается "x_1 в том и только том случае, если x_2".

Стрелка Пирса (функция Вебба, функция Даггера, штрих Лукасевича, антидизъюнкция):

$$f_8(x_1, x_2) = x_1 \downarrow x_2 .$$

Эта функция является отрицанием дизъюнкции и поэтому ее называют также "*НЕ ИЛИ*". Читается "не x_1 или x_2".

Штрих Шеффера (антиконъюнкция):

$$f_{14}(x_1, x_2) = x_1 \mid x_2 .$$

Эта функция – отрицание конъюнкции и поэтому ее называют также "*НЕ И*". Читается "не x_1 и x_2".

Функция запрета (отрицание импликации):

$$f_2(x_1, x_2) = x_1 \Delta x_2 .$$

Читается "x_1, но не x_2". Аналогично

$$f_4(x_1, x_2) = x_2 \Delta x_1 .$$

4.2. Формулы. Суперпозиция. Основные тавтологии

В булевой алгебре, как и в элементарной алгебре, можно строить формулы, исходя из элементарных функций. Ниже приводится индуктивное определение формул.

Пусть B – некоторое подмножество функций из P_2.

а) *Базис индукции*. Каждая функция $f(x_1, x_2, ..., x_n) \in B$ называется *формулой над* B.

б) *Индуктивный переход*. Пусть $f_0(x_1, x_2, ..., x_n)$ – функция из B и $A_1, ..., A_n$ – выражения, являющиеся либо формулами над B, либо символами переменных из исходного алфавита переменных $X = \{x_1, x_2, ..., x_n\}$. Тогда выражение $f_0(A_1, A_2, ..., A_n)$ называется *формулой над* B.

Пример 4.1. Пусть B – множество элементарных функций. Следующие выражения являются формулами над B:

1) $\{[(x_1 \cdot x_2) + x_1] + x_2\}$;

2) $[(\overline{x}_1 \cdot (x_2 + x_3)]$;

3) $\overline{\{x_1 \vee [(x_2 \to x_3) \& (x_3 \to x_2)]\}}$.

Далее будем обозначать формулы заглавными буквами с квадратными скобками, в которых перечисляются функции, необходимые для их построения. Так $U[f_1, ..., f_k]$ означает, что формула U построена из функций $f_1, ..., f_k$. В тех случаях, когда нужно обратить внимание на множество тех переменных, которые участвуют в построении формулы, пишут $U(x_1, ..., x_n)$.

Пусть U – произвольная формула над B, тогда формулы, которые использовались для ее построения, будем называть *подформулами формулы* U.

Пусть U_1 является формулой над множеством $B_1 = \{f_1(x_1, ..., x_n), ..., f_k(x_1, ..., x_n)\}$. Возьмем множество функций $B_2 = \{g_1(x_1, ..., x_n), ..., g_k(x_1, ..., x_n)\}$.

Рассмотрим формулу $U_2 = U_2[g_1, ..., g_k]$, которая получается из U_1 путем подстановки $\begin{pmatrix} f_1, ..., f_k \\ g_1, ..., g_k \end{pmatrix}$. Говорят, что формулы U_1 и U_2 имеют одно и то же *строение*.

Пример 4.2. Следующие формулы U_1 и U_2 имеют одинаковое строение:

1) $B_1 = \{\overline{x}_1, (x_1 \& x_2)\}$, $U_1 = [x_1 \& \overline{(x_2 \& x_3)}]$;

2) $B_2 = \{\overline{x}_1, (x_1 \to x_2)\}$, $U_2 = [x_1 \to \overline{(x_2 \to x_3)}]$.

Строение формулы обозначается через C и формула U однозначно определяется строением C и упорядоченной совокупностью $(f_1, ..., f_k)$. Поэтому можно писать

$$U = C[f_1, ..., f_k].$$

Сопоставим теперь каждой формуле $U(x_1, ..., x_n)$ над B функцию $f(x_1, ..., x_n)$ из P_2, опираясь на индуктивное определение формул.

а) *Базис индукции*. Если $U(x_1, ..., x_n) = f(x_1, ..., x_n)$, где $f \in B$, то формуле $U(x_1, ..., x_n)$ сопоставим функцию $f(x_1, ..., x_n)$.

б) *Индуктивный переход*. Пусть $U(x_1, ..., x_n) = f_0(A_1, ..., A_k)$, где A_i $(i = 1, ..., k)$ является либо формулой над B, либо символом переменной x_j. Сопоставим формуле $U(x_1, ..., x_n)$ функцию $f(x_1, ..., x_n) = f_0(f_1, ..., f_k)$.

Если функция f соответствует формуле U, то говорят также, что формула U *реализует функцию* f.

Введенное ранее понятие булевой функции не позволяет рассматривать функции от меньшего числа аргументов как функции от большего числа переменных. Для устранения этого недостатка введем следующее определение.

Функция $f(x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n) \in P_2$ зависит существенным образом от аргумента x_i, если существуют такие значения $\alpha_1, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_n$ переменных $x_1, ..., x_{i-1}, x_{i+1}, ..., x_n$, что

$$f(\alpha_1, ..., \alpha_{i-1}, 0, \alpha_{i+1}, ..., \alpha_n) \neq f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n).$$

В этом случае x_i называется *существенной переменной.* В противном случае – *несущественной* или *фиктивной.*

Функции f_1 и f_2 называются *равными*, если функцию f_2 можно получить из f_1 путем добавления или изъятия фиктивных аргументов.

Поскольку функции рассматриваются с точностью до фиктивных переменных, то формула U реализует любую функцию, равную f.

Функцию f, соответствующую формуле U, будем называть *суперпозицией* функций из B, а процесс получения функции f из B *– операцией суперпозиции.*

Операция суперпозиции включает две простые операции.

1) *Операция подстановки переменных*. Пусть

$$S = \begin{pmatrix} x_1, \ ..., x_n \\ x_{i_1}, \ ..., x_{i_n} \end{pmatrix}$$

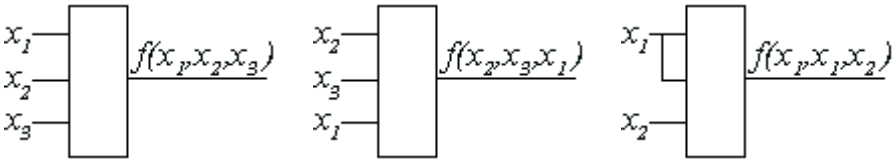
– подстановка переменных, которая позволяет получить в результате функцию $f(x_{i_1}, ..., x_{i_n})$, где $x_{i_1}, ..., x_{i_n}$ – любые переменные, не обязательно различные. Подстановка переменных включает в себя переименование, перестановку и отождествление переменных. Обозначим эту операцию буквой P:

$$P(f(x_1, ..., x_n) = f(x_{i_1}, ..., x_{i_n}).$$

2) *Операция бесповторной подстановки функций*. Она позволяет строить выражения $f(A_1, ..., A_k)$, где A_i – либо формула, либо переменная из U, причем хотя бы одно из A_i отлично от переменной. Обозначим эту операцию буквой S.

Таким образом, всякая формула над B может быть получена из функций, принадлежащих B, с помощью суперпозиции, то есть путем применения сначала операции бесповторной подстановки функций S (многократной), а затем операции подстановки переменных P (однократной).

Пример 4.3. Пусть имеется элемент, работа которого описывается некоторой булевой функцией $f(x_1, x_2, x_3)$ (рис. 4.2(а)). Результаты применения операции P приведены на рис. 4.2(б) (изменение обозначения входов) и рис. 4.2,(в) (подача одного и того же сигнала на несколько входов).



а) б) в)

Рис. 4.2. Иллюстрация применения операции P

Пример 4.4. Выразим операцию отрицания через штрих Шеффера:

$$f(x_1,\ x_2) = x_1 \mid x_2\,;\ P(f(x_1,\ x_2)) = f(x,\ x) = \overline{x \,\&\, x} = \overline{x}\,;\ \ x \mid x = \overline{x}$$

.

Пример 4.5. Пусть работа двух элементов описывается функциями $f(x_1, x_2, x_3)$ и $g(y_1, y_2)$. Соединим элементы, как показано на рис. 4.3.

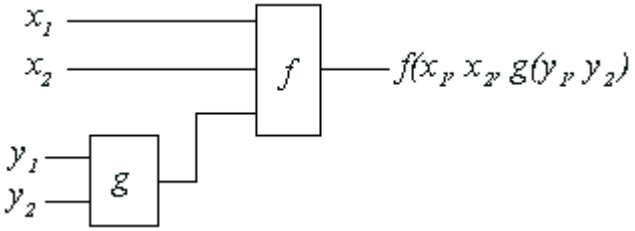


Рис. 4.3. Иллюстрация применения операции S

Функция на выходе схемы получается в результате подстановки функции g на место третьего аргумента функции f:

$$S(f(x_1,\ x_2, x_3),\ g(y_1,\ y_2)) = f(x_1,\ x_2, g(y_1,\ y_2))\,.$$

Пример 4.6. Рассмотрим систему функций $B = \{f_1,\ f_2,\ f_3\}$, где

$$f_1(x_1,\ x_2) = x_1 \vee x_2\,,\ \ f_2(x_1,\ x_2) = x_1 \,\&\, x_2\,,\ \ f_3(x) = \overline{x}\,.$$

Представим в виде суперпозиции этих функций сумму по модулю 2:

$$f(x_1,\ x_2) = x_1 \oplus x_2 = \overline{x}_1 \,\&\, x_2 \vee x_1 \,\&\, \overline{x}_2\,, \text{ или}$$

$$f(x_1,\ x_2) = f_1(f_2(f_3(x_1),\ x_2),\ f_2(x_1,\ f_3(x_2)))\,.$$

Для стрелки Пирса по аналогии можно записать:

$$x_1 \downarrow x_2 = \overline{x_1 \vee x_2} = f_3(f_1(x_1,\ x_2))\,, \text{ или}$$

$$x_1 \downarrow x_2 = \overline{x}_1 \,\&\, \overline{x}_2 = f_2(f_3(x_1),\ f_3(x_2))\,.$$

Ранее было показано, что каждой формуле над B соответствует функция алгебры логики, причем различным формулам могут соответствовать равные функции.

Формулы U_1 и U_2 называются *эквивалентными*, если соответствующие им функции f_{U_1} и f_{U_2} равны, т. е. $f_{U_1} = f_{U_2}$. Запись $U_1 = U_2$ означает, что формулы U_1 и U_2 эквивалентны.

Приведем список основных эквивалентностей (тождеств, тавтологий) алгебры логики, которые соответствуют теоретико-множественным тождествам, если заменить дизъюнкцию на объединение, конъюнкцию – на пересечение, отрицание – на дополнение.

1. $(x_1 \vee x_2) = (x_2 \vee x_1)$ (коммутативность дизъюнкции).

2. $(x_1 \vee (x_2 \vee x_3)) = ((x_1 \vee x_2) \vee x_3)$ (ассоциативность дизъюнкции).

3. $(x_1 \,\&\, x_2) = (x_2 \,\&\, x_1)$ (коммутативность конъюнкции).

4. $(x_1 \,\&\, (x_2 \,\&\, x_3)) = ((x_1 \,\&\, x_2) \,\&\, x_3)$ (ассоциативность конъюнкции).

5. $(x_1 \vee x_1) = x_1$ (правило повторения для дизъюнкции).

6. $(x_1 \,\&\, x_1) = x_1$ (правило повторения для конъюнкции).

7. $(x_1 \vee 1) = 1$
8. $(x_1 \vee 0) = x_1$
9. $(x_1 \,\&\, 0) = 0$
10. $(x_1 \,\&\, 1) = x$

(операции с 0 и 1 для дизъюнкции и конъюнкции).

11. $\overline{\overline{x}} = x$ (закон двойного отрицания).

12. $\overline{0} = 1$
13. $\overline{1} = 0$

(правила инверсии).

14. $(x_1 \,\&\, (x_2 \vee x_3)) = (x_1 \,\&\, x_2 \vee x_1 \,\&\, x_3)$ (дистрибутивность конъюнкции).

15. $(x_1 \vee x_2 \,\&\, x_3) = ((x_1 \vee x_2) \,\&\, (x_1 \vee x_3))$ (дистрибутивность дизъюнкции).

16. $\overline{(x_1 \vee x_2)} = (\overline{x}_1 \,\&\, \overline{x}_2)$

17. $\overline{(x_1 \,\&\, x_2)} = (\overline{x}_1 \vee \overline{x}_2)$

(законы де Моргана).

18. $(x_1 \vee \overline{x}_1) = 1$.

19. $(x_1 \,\&\, \overline{x}_1) = 0$.

Справедливость тождеств устанавливается сопоставлением таблиц истинности для функций, записанных в левых и правых частях тождеств.

С целью упрощения записи формул принимают следующие соглашения.

1°. Приоритет логических связок. Пусть σ – множество логических связок:

$$\sigma = \{\neg,\ \&,\ \vee,\ \oplus,\ \square,\ \to,\ |,\ \downarrow,\ \Delta\}.$$

а) считают, что связка $\neg$ сильнее любой двухместной связки из σ;

б) связка $\&$ считается сильнее, чем любая из оставшихся связок.

2°. Обозначим через $\circ$ любую из связок $\&$, $\vee$. В силу закона ассоциативности можно вместо формул $(x_1 \circ (x_2 \circ x_3))$, $((x_1 \circ x_2) \circ x_3)$ использовать выражение $(x_1 \circ x_2 \circ x_3)$, которое не является формулой, но может быть превращено в нее путем расстановки скобок.

3. Внешние скобки у формул опускаются. Опускаются также скобки у выражения, над которым стоит знак отрицания.

Эти соглашения позволяют, например, формулу $((\neg x) \to ((x \,\&\, y) \vee z))$ записать в виде $\overline{x} \to (x \,\&\, y \vee z)$.

Рассмотрим некоторые следствия из тождеств 1 – 19.

Тождества 2 и 3 позволяют рассматривать логические суммы и произведения с любым числом слагаемых и результат не зависит от расстановки скобок. Поэтому далее будем использовать следующие обозначения:

$$\bigvee_{i=1}^{n} x_i = x_1 \vee x_2 \vee \ ... \vee x_n,\ \ \&_{i=1}^{n} x_i = x_1 \,\&\, x_2 \,\&\, ... \,\& x_n.$$

Удобно сформулировать ряд правил, вытекающих из тавтологий.

1) Если в логическом произведении один из сомножителей равен 0, то и произведение равно 0.

2) Если в логическом произведении, содержащем не менее двух сомножителей, имеется сомножитель, равный 1, то этот сомножитель можно зачеркнуть.

3) Если в логической сумме, содержащей не менее двух слагаемых, имеется слагаемое, равное 0, то это слагаемое можно зачеркнуть.

4) Если в логической сумме одно из слагаемых равно 1, то сумма равна 1.

Пример 4.7. Правило поглощения:

$$x_1 \vee x_1 \,\&\, x_2 = x_1 \cdot 1 \vee x_1 \cdot x_2 = x_1 \cdot (x_2 \vee \overline{x}_2) \vee x_1 \cdot x_2 =$$

$$= x_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2 \vee x_1 \cdot x_2 = x_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2 = x_1 \cdot (x_2 \vee \overline{x}_2) = x_1 \cdot 1 = x_1 .$$

4.3. Принцип двойственности. Разложение булевых функций по переменным. Совершенные дизъюнктивная и конъюнктивная нормальные формы

Функция $f^*(x_1, x_2, ..., x_n)$, равная $\overline{f}(\overline{x}_1, \overline{x}_2, ..., \overline{x}_n)$, называется *двойственной* функцией к функции $f(x_1, x_2, ..., x_n)$.

Очевидно, что таблица истинности для двойственной функции f^* получается из таблицы истинности для функции f инвертированием (т. е. заменой 0 на 1 и 1 на 0) значений переменных и функции. Например, $f(0, 1, 1, 0) \Rightarrow f^*(1, 0, 0, 1)$.

Легко установить для функций 0, 1, x, $\overline{x}$, $x_1 \,\&\, x_2$, $x_1 \vee x_2$:

1) функция 0 двойственна 1;
2) функция 1 двойственна 0;
3) функция x двойственна x;
4) функция $\overline{x}$ двойственна $\overline{x}$;
5) функция $x_1 \,\&\, x_2$ двойственна $x_1 \vee x_2$;

6) функция $x_1 \vee x_2$ двойственна.

Из определения двойственности следует, что

$$f^{**} = (f^*)^* = f,$$

т. е. функция f является двойственной к f^* (свойство взаимности).

Принцип двойственности. Если формула $U = C[f_1, ..., f_r]$ реализует функцию $f(x_1, ..., x_n)$, то формула $U^* = C[f_1^*, ..., f_r^*]$, т. е. формула, полученная из U заменой функций $U^* = C[f_1^*, ..., f_r^*]$ соответственно на $f_1^*, ..., f_r^*$, реализует функцию $f^*(x_1, ..., x_n)$.

Формулу U^* будем называть формулой, двойственной к U.

Для доказательства этого утверждения необходимо проверить его справедливость для элементарных шагов суперпозиции P и S.

Пусть, например, функция $f(x_1, x_2)$ получается из функции $g(x_1, x_2, x_3, x_4)$ в результате следующей подстановки переменных P:

$$f(x_1, x_2) = g(x_2, x_1, x_1, x_2).$$

Тогда

$$f^*(x_1, x_2) = \overline{f}(\overline{x}_1, \overline{x}_2) = \overline{g}(\overline{x}_2, \overline{x}_1, \overline{x}_1, \overline{x}_2) = g^*(x_2, x_1, x_1, x_2)$$

т. е. функция f^* получается из g^* в результате той же самой подстановки переменных.

Доказательство справедливости принципа двойственности для шага S проведем на примере. Пусть

$$f(x_1, x_2, x_3, x_4) = f_1(x_1, x_2, f_2(x_3, x_4)).$$

Тогда

$$f^*(x_1, x_2, x_3, x_4) = \overline{f}(\overline{x}_1, \overline{x}_2, \overline{x}_3, \overline{x}_4) = \overline{f}_1(\overline{x}_1, \overline{x}_2, f_2(\overline{x}_3, \overline{x}_4)) =$$

$$= \overline{f}_1(\overline{x}_1, \overline{x}_2, \overline{\overline{f}}_2(\overline{x}_3, \overline{x}_4)) = \overline{f}_1(\overline{x}_1, \overline{x}_2, \overline{f}_2^*(x_3, x_4)) = f_1^*(x_1, x_2, f_2^*(x_3, x_4)),$$

т. е. функция f^* получается из f_1^* и f_2^* так же, как функция f из f_1 и f_2.

Принцип двойственности позволяет упростить вывод основных тавтологий и имеет целый ряд полезных применений, которые будут рассмотрены далее.

Пример 4.8. Из тождества $\overline{x_1 \,\&\, x_2} = \bar{x}_1 \vee \bar{x}_2$ следует тождество $\overline{x_1 \vee x_2} = \bar{x}_1 \,\&\, \bar{x}_2$.

Действительно,

$$f_1 = \overline{x_1 \,\&\, x_2} \Rightarrow f_1^* = \overline{\overline{\bar{x}_1 \,\&\, \bar{x}_2}} = \bar{x}_1 \,\&\, \bar{x}_2;$$

$$f_2 = \bar{x}_1 \vee \bar{x}_2 \Rightarrow f_2^* = \overline{\bar{\bar{x}}_1 \vee \bar{\bar{x}}_2} = \overline{x_1 \vee x_2};\ f_1 = f_2 \Rightarrow f_1^* = f_2^*.$$

Пример 4.9. *Построение формулы для отрицания функции.*

Из определения двойственной функции следует

$$\bar{f}(x_1,\ x_2,\ ...,\ x_n) = f^*(\bar{x}_1,\ \bar{x}_2,\ ...,\ \bar{x}_n).$$

Получаем следующее правило: *пусть формула* $U = C[f_1,\ ...,\ f_r]$ *реализует функцию* $f(x_1,\ ...,\ x_n)$. *Чтобы получить формулу для функции* $\bar{f}$ *нужно в формуле* $U^* = C[f_1^*,\ ...,\ f_r^*]$ *заменить все переменные на их отрицания.*

Найдем отрицание для функции

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

Так как $(\vee)^* = \&,\ (\&)^* = \vee,\ (\neg)^* = \neg$, то

$$\bar{f}(x_1,\ x_2,\ x_3) = \bar{x}_1 \vee \bar{x}_2 \,\&\, \overline{\bar{x}_2 \,\&\, (\bar{x}_1 \vee \bar{x}_3)}.$$

Введем обозначение

$$x^\sigma = x \cdot \sigma \vee \bar{x} \cdot \bar{\sigma},$$

где σ – параметр, равный либо 0, либо 1. Очевидно, что

$$x^\sigma = \begin{cases} \bar{x} \text{ при } \sigma = 0, \\ x \text{ при } \sigma = 1. \end{cases}$$

Легко видеть, что $x^\sigma = 1$ тогда и только тогда, когда $x = \sigma$.

Теорема 4.1 (о разложении функций по переменным). *Каждую функцию алгебры логики* $f(x_1,\ ...,\ x_n)$

при любом m $(1 \le m \le n)$ *можно представить в следующей форме:*

$$f(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n) = \\ = \bigvee_{(\sigma_1,\ldots,\sigma_m)} x_1^{\sigma_1} \& \ldots \& x_m^{\sigma_m} \& f(\sigma_1,\ldots,\sigma_m, x_{m+1}, \ldots, x_n), \quad (4.1)$$

где дизъюнкция берется по всевозможным наборам значений переменных $x_1, \ldots, x_m$.

Это представление называется *разложением функции по* m *переменным* $x_1, \ldots, x_m$.

Доказательство. Рассмотрим произвольный набор значений переменных $(\alpha_1, \ldots, \alpha_n)$ и покажем, что левая и правая части соотношения (1) принимают на нем одно и то же значение. Левая часть дает $f(\alpha_1, \ldots, \alpha_n)$. Правая –

$$\bigvee_{(\sigma_1,\ldots,\sigma_m)} \alpha_1^{\sigma_1} \& \ldots \& \alpha_m^{\sigma_m} \& f(\sigma_1,\ldots,\sigma_m, \alpha_{m+1}, \ldots, \alpha_n) =$$

$$= \alpha_1^{\alpha_1} \& \ldots \& \alpha_m^{\alpha_m} \& f(\alpha_1,\ldots,\alpha_m, \alpha_{m+1}, \ldots, \alpha_n) = f(\alpha_1, \ldots, \alpha_n).$$

В качестве следствий из теоремы рассмотрим два специальных случая разложения.

1) *Разложение по переменной:*

$$f(x_1, \ldots, x_{n-1}, x_n) = x_n \& f(x_1, \ldots, x_{n-1}, 1) \vee \bar{x}_n \& f(x_1, \ldots, x_{n-1}, 0).$$

Функции $f(x_1, \ldots, x_{n-1}, 0)$ и $f(x_1, \ldots, x_{n-1}, 1)$ называются *компонентами разложения.* Данное разложение полезно, когда какие-либо свойства устанавливаются по индукции.

2) *Разложение по всем* n *переменным:*

$$f(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n) = \bigvee_{(\sigma_1,\ldots,\sigma_n)} x_1^{\sigma_1} \& \ldots \& x_n^{\sigma_n} \& f(\sigma_1,\ldots,\sigma_n).$$

При $f(x_1, \ldots, x_n)$ тождественно не равной 0 оно может быть преобразовано:

$$\bigvee_{(\sigma_1,\ldots,\sigma_n)} x_1^{\sigma_1} \& \ldots \& x_n^{\sigma_n} \& f(\sigma_1,\ldots,\sigma_n) = \bigvee_{\substack{(\sigma_1,\ldots,\sigma_n) \\ f(\sigma_1,\ldots,\sigma_n)=1}} x_1^{\sigma_1} \& \ldots \& x_n^{\sigma_n}.$$

В результате окончательно получим

$$f(x_1, \ldots, x_n) = \bigvee_{\substack{(\sigma_1,\ldots,\sigma_n) \\ f(\sigma_1,\ldots,\sigma_n)=1}} x_1^{\sigma_1} \& \ldots \& x_n^{\sigma_n}. \qquad (4.2)$$

Такое разложение называется *совершенной дизъюнктивной нормальной формой* (совершенной д. н. ф.).

Непосредственно к понятию совершенной д. н. ф. примыкает следующая теорема.

Т е о р е м а 4 . 2 . *Каждая функция алгебры логики может быть представлена формулой в базисе* $S = \{\neg, \&, \vee\}$.

Д о к а з а т е л ь с т в о . 1) Пусть $f(x_1, \ldots, x_n) \equiv 0$. Тогда, очевидно,

$$f(x_1, \ldots, x_n) = x_1 \& \bar{x}_1.$$

2) Пусть $f(x_1, \ldots, x_n)$ тождественно не равна 0. Тогда ее можно представить формулой (2).▪

Данная теорема носит конструктивный характер, так как она позволяет для каждой функции построить реализующую ее формулу в виде совершенной д. н. ф. Для этого в таблице истинности для каждой для функции $f(x_1, \ldots, x_n)$ отмечаем все строки $(\sigma_1,\ldots,\sigma_n)$, в которых $f(\sigma_1,\ldots,\sigma_n) = 1$. Для каждой такой строки образуем логическое произведение $x_1^{\sigma_1} \& \ldots \& x_n^{\sigma_n}$, а затем все полученные конъюнции соединим знаком дизъюнкции.

П р и м е р 4 . 1 0 **.** Найти совершенную д. н. ф. для функции $x_1 \to x_2$.

$x_1\ x_2$	$x_1 \to x_2$
0 0	1
0 1	1
1 0	0
1 1	1

$$x_1 \to x_2 = x_1^0 \& x_2^0 \vee x_1^0 \& x_2^1 \& x_1^1 \& x_2^1 = \\ = \bar{x}_1 \& \bar{x}_2 \vee \bar{x}_1 \& x_2 \vee x_1 \& x_2.$$

Совершенная д. н. ф. есть выражение типа ΣП. Покажем, что при f тождественно не равной 1 ее можно представить в виде ΠΣ. Запишем для двойственной функции (очевидно не равной тождественно 0) разложение в виде совершенной д. н. ф.:

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

Из принципа двойственности следует

$$f^{**}(x_1, \ldots, x_n) = \bigwedge_{\substack{(\sigma_1,\ldots,\sigma_n) \\ f^*(\sigma_1,\ldots,\sigma_n)=1}} x_1^{\sigma_1} \vee \ldots \vee x_n^{\sigma_n} = \bigwedge_{\substack{(\sigma_1,\ldots,\sigma_n) \\ f(\bar{\sigma}_1,\ldots,\bar{\sigma}_n)=0}} x_1^{\sigma_1} \vee \ldots \vee x_n^{\sigma_n} =$$

$$= \bigwedge_{\substack{(\sigma_1,\ldots,\sigma_n) \\ f(\sigma_1,\ldots,\sigma_n)=0}} x_1^{\bar{\sigma}_1} \vee \ldots \vee x_n^{\bar{\sigma}_n}.$$

Таким образом, получаем разложение, которое называется *совершенной конъюнктивной нормальной формой* (совершенной к. н. ф.):

$$f(x_1, \ldots, x_n) = \bigwedge_{\substack{(\sigma_1,\ldots,\sigma_n) \\ f(\sigma_1,\ldots,\sigma_n)=0}} x_1^{\bar{\sigma}_1} \vee \ldots \vee x_n^{\bar{\sigma}_n}. \qquad (4.3)$$

Пример 4.11. Построить совершенную к. н. ф. для функции $x_1 \to x_2$.

Имеем: $x_1 \to x_2 = x_1^1 \vee x_2^0 = x_1 \vee \bar{x}_2$.

4.4. Полнота и замкнутость. Полином Жегалкина. Важнейшие замкнутые классы. Теорема о полноте

Система функций $\{f_1, f_2, \ldots, f_r\}$ из P_2 называется *функционально полной*, если любая булева функция может быть представлена в виде формулы через функции этой системы.

Примеры полных систем.

1) Система P_2 – множество всех булевых функций.

2) Система $B = \{\neg x,\ x_1 \& x_2,\ x_1 \vee x_2\}$

Очевидно, что не каждая система является полной, например, система $B = \{0, 1\}$ не полная. Следующая теорема позволяет сводить вопрос о полноте одних систем к вопросу о полноте других систем.

Теорема 4.3. Пусть даны две системы функций из P_2:

$$B_1 = \{f_1, f_2, \ldots, f_r\},\ B_2 = \{g_1, g_2, \ldots, g_s\},$$

относительно которых известно, что система B_1 полна и каждая ее функция выражается в виде формулы через функции системы B_2. Тогда система B_2 является полной.

Доказательство. Пусть h – произвольная функция из P_2. В силу полноты системы B_1 можно выразить h формулой над B_1, т. е. $h = C[f_1, f_2, ..., f_r]$.

По условию теоремы

$$f_1 = C_1[g_1, g_2, ..., g_s],\ f_2 = C_1[g_1, g_2, ..., g_s], \ldots,$$
$$f_r = C_1[g_1, g_2, ..., g_s].$$

Поэтому в формуле $h = C[f_1, f_2, ..., f_r]$ можно исключить вхождения функций $f_1, f_2, ..., f_r$, заменив их формулами над B_2:

$$h = C[f_1, f_2, ..., f_r] = C[C_1[g_1, g_2, ..., g_s], ..., C_r[g_1, g_2, ..., g_s]] = C'[g_1, g_2, ..., g_s],$$

То есть мы выразили h в виде формулы над B_2. ▪

Опираясь на эту теорему, можно установить полноту еще ряда систем и тем самым расширить список примеров полных систем.

1) Система $B = \{\bar{x}, x_1 \& x_2\}$ является полной. Для доказательства возьмем за систему B_1 систему 2), а за систему B_2 – систему 3) и используем тождество

$$x_1 \vee x_2 = \overline{\bar{x}_1 \& \bar{x}_2}.$$

2) Система $B = \{\bar{x}, x_1 \vee x_2\}$ является полной. Доказательство аналогично предыдущему, либо через принцип двойственности.

3) Система $B = \{x_1 | x_2\}$ является полной. Для доказательства возьмем за систему B_1 систему 3), а за систему B_2 – систему 5). Тогда

$$\bar{x}_1 = x_1 | x_1,\ x_1 \& x_2 = \overline{x_1 | x_2} = (x_1 | x_2) | (x_1 | x_2).$$

4) Система $B=\{0,\ 1, x_1 \& x_2, x_1 \oplus x_2\}$ является полной. Для доказательства возьмем за систему B_1 систему 3), а за систему B_2 – систему 6). Имеем

$$\overline{x}_1 = x_1 \oplus 1.$$

С понятием полноты тесно связано понятие замыкания и замкнутого класса.

Пусть M – некоторое подмножество функций из P_2. *Замыканием* M называется множество всех булевых функций, представимых в виде формул через функции множества M. Замыкание множества M обозначается через $[M]$. Очевидно, что замыкание инвариантно относительно операций введения и удаления фиктивных переменных.

Свойства замыкания:

1) $[M] \supseteq M$;

2) $[[M]] = [M]$;

3) если $M_1 \subseteq M_2$, то $[M_1] \subseteq [M_2]$;

4) $[M_1 \cup M_2] \subseteq [M_1] \cup [M_2]$.

Класс (множество) M называется *функционально замкнутым*, если $[M] = M$.

Очевидно, что всякий класс $[M]$ будет замкнутым. Это дает возможность получать многочисленные применения замкнутых классов.

В терминах замыкания и замкнутого класса можно дать другое определение полноты, эквивалентное исходному: M – полная система, если $[M] = P_2$.

Рассмотрим полную систему функций $B=\{0,\ 1, x_1 \& x_2, x_1 \oplus x_2\}$. Формула, построенная из функций этой системы, после раскрытия скобок и алгебраических преобразований переходит в полином по mod 2, который в честь русского логика конца XIX века И.И. Жегалкина называют *полиномом Жегалкина*. Возможность такого преобразования следует из следующих тождеств:

$$x_1 \oplus x_2 = x_2 \oplus x_1; \quad (4.4)$$

$$x_1 \oplus (x_2 \oplus x_3) = (x_1 \oplus x_2) \oplus x_3; \quad (4.5)$$

$$x_1 \,\&\, (x_2 \oplus x_3) = x_1 \,\&\, x_2 \oplus x_1 \,\&\, x_3; \quad (4.6)$$

$$x \oplus 0 = x; \quad (4.7)$$

$$x \oplus x = 0. \quad (4.8)$$

Действительно, из тождеств (4.4) – (4.8) следует, что формулы, содержащие операции & и $\oplus$, можно преобразовывать по обычным алгебраическим законам: переставлять слагаемые и множители, раскрывать скобки, выносить общий множитель за скобки.

Пример 4.12. Представим функцию $f = x_1 \vee x_2$ над множеством связок $\sigma = \{\&, \oplus\}$:

$$f = x_1 \vee x_2 = \overline{\overline{x}_1 \,\&\, \overline{x}_2} = ((x_1 \oplus 1) \,\&\, (x_2 \oplus 1)) \oplus 1 = \\ = x_1 \,\&\, x_2 \oplus x_1 \oplus x_2 \oplus 1 \oplus 1 = x_1 \,\&\, x_2 \oplus x_1 \oplus x_2. \quad (4.9)$$

Рассмотрим произвольную булеву функцию $f(x_1, x_2, ..., x_n)$. Представим ее формулой в базисе $\{\neg, \&, \vee\}$, например, совершенной д. н. ф., после чего заменим в этой формуле каждую операцию $\vee$ по формуле (4.9), а каждое отрицание по формуле $\overline{x} = x \oplus 1$. В полученном выражении раскроем скобки и приведем подобные члены. В результате получим полином

$$f(x_1, x_2, ..., x_n) = a_0 \oplus a_1 \cdot x_1 \oplus a_2 \cdot x_2 \oplus ... \oplus a_n \cdot x_n \oplus a_{1,2} \cdot x_1 \cdot x_2 \oplus$$

$$\oplus ... \oplus a_{n-1,n} \cdot x_{n-1} \cdot x_n \oplus a_{1,2,3} \cdot x_1 \cdot x_2 \cdot x_3 \oplus ... \oplus a_{1,2,...,n} \cdot x_1 \cdot x_2 \cdot ... \cdot x_n,$$

где коэффициенты полинома либо равны 1, либо 0 в зависимости от наличия или отсутствия соответствующего члена.

Приведенное выше рассуждение доказывает справедливость следующей теоремы, принадлежащей И.И. Жегалкину.

Теорема 4.4. *Каждая функция из P_2 представима полиномом Жегалкина.*

Подсчитаем число полиномов Жегалкина от переменных $x_1, x_2, ..., x_n$, то есть число выражений вида

$$\sum_{(i_1, ..., i_s)} a_{i_1, ..., i_s} \cdot x_{i_1} \cdot ... \cdot x_{i_s}.$$

Число членов $x_{i_1} \cdot ... \cdot x_{i_s}$ равно количеству подмножеств $\{i_1, ..., i_s\}$ множества $N = \{1, ..., n\}$, то есть $B(N) = 2^n$. Поскольку коэффициент $a_{i_1, ..., i_s}$ равен 0 или 1, то искомое число полиномов равно 2^{2^n}, то есть числу всех булевых функций от тех же переменных. Отсюда, как следствие, получаем *единственность представления функций полиномом Жегалкина*.

Пример 4.13. Представить функцию $f = x_1 \vee x_2$ в виде полинома Жегалкина.

Будем искать выражение для данной функции в виде полинома с неопределенными коэффициентами:

$$f = x_1 \vee x_2 = a \cdot x_1 \cdot x_2 \oplus b \cdot x_1 \oplus c \cdot x_2 \oplus d .$$

При $x_1 = x_2 = 0$ имеем $0 = d$,

при $x_1 = 0,\ x_2 = 1$ имеем $1 = c$,

при $x_1 = 1,\ x_2 = 0$ имеем $1 = b$,

при $x_1 = 1,\ x_2 = 1$ имеем $1 = a \oplus b \oplus c$, то есть $a = 1$.

Отсюда $x_1 \vee x_2 = x_1 \cdot x_2 \oplus x_1 \oplus x_2$.

Пример 4.14. Представить полиномом Жегалкина функцию, заданную таблицей истинности:

$x_1\ x_2\ x_3$	$f(x_1\ x_2\ x_3)$
0 0 0	1
0 0 1	0
010	1
011	1
100	0
101	0
110	0
111	1

Рассматриваемый далее алгоритм построения полинома Жегалкина применим для любой булевой функции.

I. Представим заданную функцию совершенной д.н.ф.:

$$f(x_1, x_2, x_3) = \overline{x}_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot x_3 . \quad (4.9)$$

Из формулы $x_1 \vee x_2 = x_1 \cdot x_2 \oplus x_1 \oplus x_2$ следует, что для булевых функций f_1 и f_2, одновременно не обращающихся в 1 ($f_1 \cdot f_2 = 0$), выполняется равенство $f_1 \vee f_2 = f_1 \oplus f_2$. Такие функции называются *ортогональными.* Из последнего равенства следует, что логическая сумма для ортогональных слагаемых не изменяется, если операцию $\vee$ заменить на $\oplus$. Далее выполним следующий шаг, воспользовавшись этим результатом.

II. Заменим в выражении (4.9) все знаки $\vee$ на $\oplus$:

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

III. Заменим все отрицания по формуле $\overline{A} = A \oplus 1$, раскроем скобки и приведем подобные члены:

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

Из приведенных примеров следует, что существует целый ряд полных систем. Таким образом, для задания булевых функций можно использовать различные языки формул. Какой именно из языков является более удобным, зависит от характера рассматриваемой задачи.

5. Приложения алгебры логики к кибернетике

5.1. Дизъюнктивные нормальные формы

5.1.1. Понятие ДНФ. Проблема минимизации булевых функций

Пусть задан алфавит переменных $\{x_1, \ldots, x_n\}$. Выражение

$$K = x_{i_1}^{\sigma_1} \& \ldots \& x_{i_r}^{\sigma_r} \quad (i_\nu \neq i_\mu \text{ при } \nu \neq \mu)$$

называется *элементарной конъюнкцией*, а число r – ее *рангом*. По определению считаем константу 1 элементарной конъюнкцией ранга 0.

Выражение

$$R = \bigvee_{i=1}^{s} K_i \ (K_i \neq K_j \text{ при } i \neq j),$$

где K_i $(i = 1, ..., s)$ – элементарная конъюнкция ранга r_i, называется *дизъюнктивной нормальной формой (ДНФ.).*

Число различных элементарных конъюнкций над алфавитом $\{x_1, ..., x_n\}$ равно 3^n.

Действительно, число конъюнкций ранга r равно $C_n^r \cdot 2^r$. Отсюда

$$\sum_{r=0}^{n} C_n^r \cdot 2^r = \sum_{r=0}^{n} C_n^r \cdot 2^r \cdot 1^{n-r} = (1+2)^n = 3^n.$$

Для каждой булевой функции $f(x_1, ..., x_n)$ существует ДНФ R такая, что

$$f(x_1, ..., x_n) = R.$$

Говорят, что функция f представлена в виде ДНФ или ДНФ реализует функцию f. Однако, представление функции в виде ДНФ не единственно. Число булевых функций от n переменных равно 2^{2^n}, а число ДНФ – 2^{3^n}. Например, функция

$$f_1(x_1, x_2, x_3) = \begin{cases} 0, \text{ если } x_1 = x_2 = x_3; \\ 1 \text{ в остальных случаях} \end{cases}$$

может быть представлена совершенной ДНФ

$$R_1 = x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \vee \overline{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x}_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x}_3,$$

а также следующими ДНФ

$$R_2 = x_1 \cdot \overline{x}_2 \vee \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \vee \overline{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x}_3,$$

$$R_3 = x_1 \cdot \overline{x}_2 \vee x_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot x_3,$$

которые можно получить из совершенной, применяя булевы тождества.

В связи с этим возникает возможность выбора более предпочтительной реализации функций алгебры логики. Для этого

вводится индекс простоты $L(R)$, характеризующий сложность ДНФ.

Приведем примеры индексов простоты для ДНФ.

1°. $L_Б(R)$ – число букв переменных, встречающихся в записи ДНФ R.

Для предыдущего примера $L_Б(R_1)=18$, $L_Б(R_3)=6$, то есть в смысле этого индекса ДНФ R_3 проще, чем ДНФ R_1.

2°. $L_к(R)$ – число элементарных конъюнкций, входящих в R.

Для ДНФ R_1 и R_3 $L_к(R_1)=6$, $L_к(R_3)=3$, то есть в смысле этого индекса ДНФ R_3 проще, чем ДНФ R_1.

3°. $L_0(R)$ – число символов, встречающихся в записи ДНФ R.

Для ДНФ R_1 и R_3 $L_0(R_1)=9$, $L_0(R_3)=3$, то есть в смысле этого индекса ДНФ R_3 опять проще, чем ДНФ R_1.

ДНФ, реализующая функцию $f(x_1, ..., x_n)$ и имеющая минимальный индекс $L(R)$, называется *минимальной относительно* L (*МДНФ относительно* L).

Поскольку дальнейшее изложение связано главным образом с индексом простоты $L_б(R)$, то минимальную д. н. ф. относительно этого индекса будем называть *минимальной ДНФ* (*МДНФ.*). ДНФ, минимальную относительно индекса $L_б(R)$, будем называть *кратчайшей.*

Задача минимизации булевых функций состоит в построении минимальной ДНФ для произвольной функции алгебры логики $f(x_1, ..., x_n)$. Эта задача называется *проблемой минимизации булевых функций*. Рассмотрим геометрическую интерпретацию этой задачи.

Обозначим через E^n множество всех наборов $(\alpha_1, ..., \alpha_n)$ из 0 и 1. Его можно рассматривать как множество всех вершин единичного n-мерного куба, который в дальнейшем будем

называть просто n-мерным кубом, а наборы $(\alpha_1, ..., \alpha_n)$ – вершинами куба. На рис. 5.1 представлена проекция 3-мерного куба на плоскость.

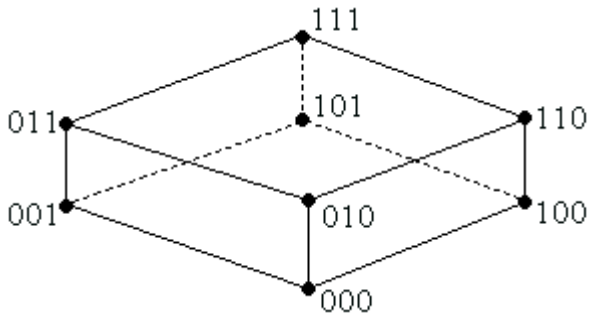


Рис. 5.1

Пусть $\sigma_{i_1}, ..., \sigma_{i_r}$ – фиксированная система чисел из 0 и 1 такая, что $1 \le i_1 < ... < i_r \le n$. Множество всех вершин $(\alpha_1, ..., \alpha_n)$ куба E^n таких, что

$$\alpha_{i_1} = \sigma_{i_1}, ..., \alpha_{i_r} = \sigma_{i_r},$$

называется $(n-r)$-*мерной гранью*, то есть имеем подмножество $N_{K_r} \subseteq E^n$:

$$N_{K_r} = \{(\alpha_1, ..., \alpha_n) \in E^n \mid \alpha_{i_1} = \sigma_{i_1}, ..., \alpha_{i_r} = \sigma_{i_r}\}.$$

Очевидно, что $(n-r)$-мерная грань является $(n-r)$-мерным подкубом куба E^n.

Число $(n-r)$-мерных граней куба E^n равно $C_n^r \cdot 2^r$.

Число r называют *рангом* грани N_K и обозначают $rang(N_K)$. Таким образом, сумма размерности и ранга грани равна n. Размерности граней куба (ранги) могут изменяться в пределах от 0 до n. Нульмерные грани ($r = n$) – это вершины куба E^n; одномерные грани называют *ребрами*; грань размерности n – это сам куб E^n.

Куб и совокупность его граней образуют топологический объект, называемый *кубическим комплексом*. Относительно этого достаточно сложного объекта многие вопросы остаются нерешенными до настоящего времени. Например, неизвестно каково минимальное множество вершин, обладающим тем свойством, что в этом множестве есть по крайней мере один представитель от каждой k-мерной грани (задача Лупанова о «протыкании» граней).

Сопоставим каждой функции $f(x_1, ..., x_n)$ алгебры логики подмножество вершин $N_f \subseteq E^n$, в которых эта функция равна 1:

$$N_f = \{(\alpha_1, ..., \alpha_n) \mid f(\alpha_1, ..., \alpha_n) = 1\}.$$

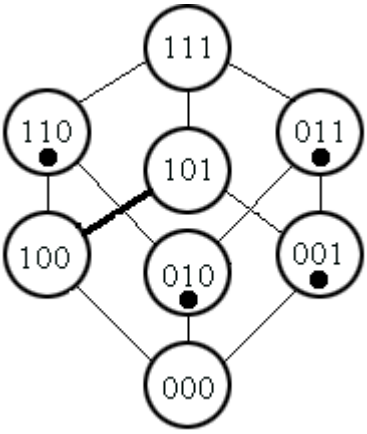
При этом булевым операциям над функциями будут соответствовать теоретико-множественные операции над подмножествами E^n. Следующая таблица устанавливает соответствие между аналитическими и геометрическими объектами.

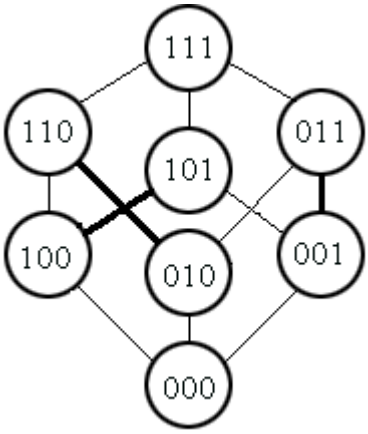
Таблица 5.1. Соответствие аналитических и геометрических объектов

Аналитический объект	Геометрический объект
Двоичные слова длины n	Вершины n-мерного куба E^n
Булева функция $f(x_1, ..., x_n)$	Подмножество вершин $N_f \subseteq E^n$
Элементарная конъюнкция $K = x_{i_1}^{\sigma_1} \& ... \& x_{i_r}^{\sigma_r}$	N_k-грань куба E^n $x_{i_1} = \sigma_{i_1}, ..., x_{i_r} = \sigma_{i_r}$
Формулы $f = g \vee h$ $f = g \cdot h$ $f = \bar{g}$	Соотношения $N_f = N_g \bigcup N_h$ $N_f = N_g \bigcap N_h$ $N_f = E^n \setminus N_g$
Д. н. ф. $\bigvee_{i=1}^{s} K_i$	Объединение граней $\bigvee_{i=1}^{s} N_{K_i}$
Представление функции ДНФ $f = \bigvee_{i=1}^{s} K_i$	Покрытие N_f гранями $N_f = \bigvee_{i=1}^{s} N_{K_i}$

Из таблицы следует, что каждому представлению булевой функции $f(x_1, ..., x_n)$ в виде ДНФ соответствует покрытие под-

множества вершин $N_f \subseteq E^n$ гранями. Например, двум ДНФ R_2 и R_3 функции $f_1(x_1, x_2, x_3)$ из рассмотренного выше примера соответствуют покрытия множества N_f, представленные на рис. 5.2.





$$R_2 = x_1 \cdot \overline{x}_2 \vee \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \vee \vee \overline{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x}_3$$

$$R_3 = x_1 \cdot \overline{x}_2 \vee x_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot x_3$$

Рис. 5.2

Первое покрытие состоит из четырех точек (нульмерных граней) и ребра (одномерной грани), второе – из трех ребер (одномерных граней).

Пусть $r_i = rang(N_{K_i})$. Число $r = \sum_{i=1}^{s} r_i$ будем называть *рангом покрытия*. Для любого покрытия множества N_f гранями, очевидно, что число переменных в соответствующей этому покрытию д. н. ф. будет равно сумме рангов граней, то есть рангу покрытия. Следовательно, МДНФ отвечает покрытие с наименьшим рангом.

Сформулируем геометрическую задачу о покрытии, эквивалентную задаче о минимизации булевой функции.

Дано подмножество вершин единичного n-мерного куба. Найти покрытие этого подмножества содержащимися в нем гранями с наименьшим рангом.

Переход от аналитического к геометрическому языку в ряде случаев помогает быстро найти МДНФ. Например, для функции f_1 из рассмотренного выше примера гранями, содержащимися в N_{f_1}, являются одномерные (покрывают две вершины) и нульмерные (покрывают одну вершину) грани. Множество N_{f_1} состоит из 6 вершин, поэтому его покрытие включает не менее трех граней. Следовательно, ДНФ R_2 – минимальная.

5.1.2. Тупиковые и сокращенные ДНФ. Геометрические и аналитические методы их построения

Определение тупиковой ДНФ. Пусть R – произвольная ДНФ, которую можно представить следующим образом:

$$R = R' \vee K,\ R = R' \vee x_i^{\sigma_i} \cdot K'.$$

Здесь K – элементарная конъюнкция из R, R' – ДНФ, образованная из остальных конъюнкций, входящих в R; $x_i^{\sigma_i}$ – некоторый множитель из K; K' – произведение остальных множителей из K.

Рассмотрим два типа преобразования ДНФ.

I. Операция удаления элементарной конъюнкции. Переход от ДНФ R к ДНФ R' – преобразование, осуществляемое путем удаления элементарной конъюнкция K. Данное преобразование определено тогда и только тогда, когда $R' = R$.

II. Операция удаления множителя. Переход от ДНФ R к ДНФ $R' \vee K'$ – преобразование, осуществляемое путем удаления множителя $x_i^{\sigma_i}$. Данное преобразование определено тогда и только тогда, когда $R' \vee K' = R$.

ДНФ R, которую нельзя упростить при помощи преобразований I и II, называется *тупиковой ДНФ* (*ТДНФ*).

Например, очевидно, что ДНФ $R = x_1 \vee \overline{x}_2 \cdot \overline{x}_3$ будет тупиковой.

Построение тупиковых ДНФ методом упрощения совершенной ДНФ. Рассмотрим алгоритм упрощения ДНФ, приводящий к тупиковой ДНФ. Отметим, что среди тупиковых ДНФ всегда содержатся и минимальные, но не все.

1°. Для функции $f(x_1, ..., x_n)$ выбирается в качестве исходной какая-нибудь ДНФ, например, совершенная ДНФ, так как существует простой способ ее построения.

2°. Осуществляется упорядочивание в исходной ДНФ слагаемых и в каждом слагаемом – множителей. Это упорядочение можно задать записью ДНФ.

3°. Производится просмотр записи ДНФ слева направо. Для очередного члена K_i $(i = 1, ..., s)$ сначала пробуют применить операцию удаления элементарной конъюнкции K_i. Если это невозможно, то просматривают члены $x_{i_\nu}^{\sigma_\nu}$ конъюнкции $K_i = x_{i_1}^{\sigma_1} \cdot ... \cdot x_{i_r}^{\sigma_r}$ слева направо $(\nu = 1, ..., r)$ и применяют операцию удаления множителя $x_{i_\nu}^{\sigma_\nu}$ до тех пор, пока это удается.

Затем переходят к следующей элементарной конъюнкции. После обработки последней элементарной конъюнкции еще раз просматривают полученную ДНФ слева направо и пробуют применить операцию удаления элементарной конъюнкции. В результате получаем ТДНФ.

Пример 5.1. Рассмотрим функцию $f(x_1, x_2, x_3) = (11011011)$. В табличной форме:

Таблица 5.2

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

В качестве исходной ДНФ выберем совершенную ДНФ

$R' = \overline{x}_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \vee \overline{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee x_1 \cdot x_2 \cdot \overline{x}_3 \vee x_1 \cdot x_2 \cdot x_3$,

рассмотрим ее упорядочение и проследим работу алгоритма.

1°. Конъюнкция $\overline{x}_1 \cdot \overline{x}_2 \cdot \overline{x}_3$ не может быть удалена, но можно удалить множитель $\overline{x}_1$:

$$\overline{x}_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 = \overline{x}_2 \cdot \overline{x}_3 .$$

2°. Конъюнкция $\overline{x}_1 \cdot \overline{x}_2 \cdot x_3$ не может быть удалена, но можно удалить множитель $\overline{x}_2$:

$$\overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 = \overline{x}_1 \cdot x_3 .$$

3°. Конъюнкция $\overline{x}_1 \cdot x_2 \cdot x_3$, может быть удалена так как

$$\overline{x}_1 \cdot x_2 \cdot x_3 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 = \overline{x}_1 \cdot x_3 .$$

4°. Конъюнкция $x_1 \cdot \overline{x}_2 \cdot \overline{x}_3$ также может быть удалена.

5°. Конъюнкцию $x_1 \cdot x_2 \cdot \overline{x}_3$ удалить нельзя, но можно удалить множитель x_2:

$$x_1 \cdot x_2 \cdot \overline{x}_3 \vee x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 = x_1 \cdot \overline{x}_3 .$$

6°. Конъюнкцию $x_1 \cdot x_2 \cdot x_3$ удалить нельзя, но можно удалить множитель x_1:

$$x_1 \cdot x_2 \cdot x_3 \vee \overline{x}_1 \cdot x_2 \cdot x_3 = x_2 \cdot x_3 .$$

В результате получаем тупиковую ДНФ:

$$R_1 = \overline{x}_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot x_3 \vee x_1 \cdot \overline{x}_3 \vee x_2 \cdot x_3 .$$

Если для той же функции взять другую упорядоченность ее совершенной ДНФ, например,

$R'' = \overline{x}_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee x_3 \cdot \overline{x}_1 \cdot \overline{x}_2 \vee x_2 \cdot \overline{x}_1 \cdot x_3 \vee x_1 \cdot x_2 \cdot x_3 \vee \overline{x}_3 \cdot x_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2 \cdot \overline{x}_3$,

то в результате работы алгоритма упрощения получим другую тупиковую ДНФ:

$$R_2 = \overline{x}_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot x_3 \vee x_1 \cdot x_2 .$$

Из приведенного примера следует, что результат применения алгоритма упрощения зависит от выбора упорядочения ис-

ходной ДНФ, причем получаемые тупиковые ДНФ имеют различную сложность. В данном случае $L_Б(R_1) = 8$, $L_Б(R_2) = 6$.

Таким образом, тупиковые ДНФ могут имеют различную сложность и, в частности, могут отличаться от минимальных. В связи с этим возникает вопрос: возможно ли для любой функции $f(x_1, ..., x_n)$, исходя из некоторого упорядочения, получить, применяя алгоритм упрощения, минимальную ДНФ? Ответ на этот вопрос дает следующая теорема.

Т е о р е м а 5.1. *Пусть* $f(x_1, ..., x_n)$ *– произвольная булева функция* ($f \neq 0$) *и* $R = \bigvee_{i=1}^{s} K_i$ *– ее произвольная тупиковая ДНФ. Тогда существует такое упорядочение совершенной ДНФ, из которого при помощи алгоритма упрощения получается тупиковая ДНФ* R.

С л е д с т в и е. В силу того, что среди тупиковых ДНФ содержатся обязательно и минимальные, алгоритм упрощения при надлежащем выборе упорядочения совершенной ДНФ позволяет находить и минимальные ДНФ.

З а м е ч а н и е. Из доказательства этой теоремы, которое мы опускаем, следует, что для построения всех тупиковых ДНФ при помощи алгоритма упрощения из совершенной ДНФ достаточно при естественном порядке конъюнкций варьировать только порядок множителей в конъюнкциях.

Таким образом, для построения минимальной ДНФ следует на основе алгоритма упрощения $R' = \bigvee_{i=1}^{s} K_i'$, варьируя только порядком множителей в конъюнкциях, найти все тупиковые ДНФ.

Выполним оценку трудоемкости такой процедуры минимизации.

Однократное применение алгоритма упрощения содержит $L_K(R') \le 2^n$ проверок возможности удаления конъюнкции, $L_Б(K_i') \le n \cdot 2^n$ проверок возможности удаления множителя из K_i'

$(i = 1, \ldots, s)$ и при вторичном просмотре не более $L_K(R') \le 2^n$ проверок возможности удаления конъюнкций, то есть

$$\sum_{i=1}^{s} L_A(K_i') + 2 \cdot L_K(R') \le (n+2) \cdot 2^n .$$

Общее число упорядочения, с учетом замечания, равно $(n!)^s$, причем

$$(n!)^s \le (n!)^{2^n} \le (n^n)^{2^n} = 2^{\log(n^n)^{2^n}} = 2^{2^n \cdot n \cdot \log n}.$$

Это число меньше, чем 2^{3^n}, то есть этот алгоритм лучше, чем алгоритм перебора всех ДНФ, но все же он остается весьма трудоемким.

Определение сокращенной ДНФ и геометрический метод ее построения. Грань N_K, содержащаяся в N_f, называется *максимальной*, если не существует грани $N_{K'}$ такой, что:

1) $N_K \subseteq N_{K'} \subseteq N_f$;

2) $\dim(N_{K'}) > \dim(N_K)$.

Пример 5.2. Пусть $f(x_1, x_2, x_3) = (11011011)$ (см. пример 1). Рассмотрим конъюнкции $\mathrm{K}_1(x_1, x_2, x_3) = \bar{x}_2 \cdot \bar{x}_3$, $\mathrm{K}_2(x_1, x_2, x_3) = x_1 \cdot \bar{x}_2$, $\mathrm{K}_3(x_1, x_2, x_3) = x_1$, которым соответствуют грани

$$N_{K_1} = \{(0, 0, 0), (1, 0, 0)\},$$

$$N_{K_2} = \{(1, 0, 0), (1, 0, 1)\},$$

$$N_{K_3} = \{(1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)\}.$$

Эти грани имеют соответственно ранги $r_1 = 2$, $r_2 = 2$, $r_3 = 1$, первые две из которых представляют одномерные грани (ребра), а последняя – двумерную грань (рис. 5.3).

Грани N_{K_1} и N_{K_3} – максимальные, а грань N_{K_2} не максимальная для N_f, так как $N_{K_2} \subset N_{K_3}$ и размерность N_{K_3} больше

размерности N_{K_2}.

Конъюнкция K, соответствующая максимальной грани N_K множества N_f, называется *простой импликантой функции* f. Из нее нельзя удалить ни одного множителя, иначе мы получим конъюнкцию K', грань которой $N_{K'}$ не содержится в N_f.

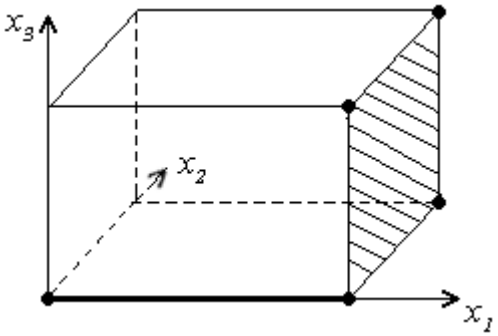


Рис. 5.3.

Из определения следует, что любую ДНФ, в которой хотя бы один из членов не является простой импликантой, можно упростить. Отсюда следует следующее утверждение.

Теорема 5.2. *Минимальная ДНФ функции* f *состоит из простых импликант.*

ДНФ, являющаяся дизъюнкцией всех простых импликант, называется *сокращенной*.

Пусть $N_{K^0} = \{N_{K_1^0}, ..., N_{K_m^0}\}$ множество всех максимальных граней множества N_f. Тогда

$$N_f = \bigcup_{i=1}^{m} N_{K_i^0},$$

так как $N_{K_i^0} \subseteq N_f$ и каждая точка из N_f принадлежит некоторой максимальной грани. Последнее равенство эквивалентно

$$f = \bigvee_{i=1}^{m} K_i^0.$$

Так как сокращенная ДНФ реализует функцию f, то она имеет вид

$$R_C = \bigvee_{i=1}^{m} K_i^0.$$

Пример 5.3. Пусть функция f задана таблицей 5.3:

Таблица 5.3

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

Этой функции соответствует множество

$$N_f = \{(0,\ 0,\ 0),\ (1,\ 0,\ 0),\ (1,\ 0,\ 1),\ (1,\ 1,\ 0),\ (1,\ 1,\ 1)\}.$$

Имеем две максимальные грани:

$$N_{K_1} = \{(0,\ 0,\ 0),\ (1,\ 0,\ 0)\},$$

$$N_{K_3} = \{(1,\ 0,\ 0),\ (1,\ 0,\ 1),\ (1,\ 1,\ 0),\ (1,\ 1,\ 1)\}.$$

Тогда покрытие для функции f имеет вид:

$$N_f = N_{K_1} \bigcup N_{K_3}.$$

Ему соответствует сокращенная ДНФ

$$R_C = \overline{x}_2 \cdot \overline{x}_3 \vee x_1.$$

Аналитические методы построения сокращенной ДНФ

Построение сокращенной ДНФ по совершенной ДНФ. Определим следующие операции над ДНФ:

1°. Склеивание: $\overline{x} \cdot K \vee x \cdot K = K$.

2°. Неполное склеивание: $\overline{x} \cdot K \vee x \cdot K = K \vee \overline{x} \cdot K \vee x \cdot K$.

3°. Поглощение: $K_1 \vee K_1 \cdot K_2 = K_1$.

Алгоритм построения сокращенной ДНФ:

1. Представить заданную функцию совершенной ДНФ.

2. Проводить операции неполного склеивания до тех пор, пока это возможно.

3. Выполнить все поглощения.

Алгоритм Нельсона построения сокращенной ДНФ. Представим заданную функцию любой КНФ, например, совершенной КНФ. Затем проведем раскрытие скобок и выполним все поглощения.

Пример 5.4. Рассмотрим функцию, заданную табл. 5.1. Построим совершенную КНФ:

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

Проведем раскрытие скобок и выполним все поглощения. В результате получим сокращенную ДНФ в виде

$$R_C = \overline{x}_1 \cdot \overline{x}_2 \vee \overline{x}_1 \cdot x_3 \vee x_2 \cdot x_3 \vee x_1 \cdot x_2 \vee x_1 \cdot \overline{x}_3 \vee \overline{x}_2 \cdot \overline{x}_3 .$$

Отметим, что сокращенная ДНФ может иметь большее число членов, чем совершенная ДНФ.

Построение тупиковых ДНФ на основе геометрических представлени. Покрытие множества N_f, состоящее из максимальных граней, называется *неприводимым*, если совокупность граней, получающаяся из исходной путем выбрасывания любой грани, не будет покрытием.

ДНФ, соответствующая неприводимому покрытию множества N_f, называется *тупиковой* в геометрическом смысле.

Будем исходить из покрытия множества N_f системой всех его максимальных граней $N_{K_1^0},\ ...,\ N_{K_m^0}$.

Пусть $N_f = \{P_1,\ ...,\ P_\lambda\}$. Составим таблицу 5.4, в которой

Таблица 5.4

	P_1	⋯	P_j	⋯	P_λ
$N_{K_1^0}$	σ_{11}	⋯	σ_{1j}	⋯	$\sigma_{1\lambda}$
…	…	…	…	…	
$N_{K_i^0}$	σ_{i1}	⋯	σ_{ij}	⋯	$\sigma_{i\lambda}$
…	…	…		…	
$N_{K_m^0}$	σ_{m1}	⋯	σ_{mj}	⋯	$\sigma_{m\lambda}$

$$\sigma_{ij} = \begin{cases} 0, \text{ если } P_j \notin N_i^0, & (i = 1, \ ..., \ m); \\ 0, \text{ если } P_j \notin N_i^0, & (i = 1, \ ..., \ m). \end{cases}$$

Очевидно, что в каждом столбце содержится хотя бы одна единица. Для каждого j найдем множество E_j всех номеров строк, в которых столбец P_j содержит 1. Пусть

$$E_j = \{e_{j_1}, \ ..., \ e_{j_{\mu(j)}}\}.$$

Составим конъюнкцию, рассматривая номера строк как булевы переменные:

$$\overset{\lambda}{\underset{j=1}{\&}}(e_{j_1} \vee \ ... \ \vee e_{j_{\mu(j)}}).$$

Произведем преобразование $\& \vee \to \vee \&$ и далее ликвидируем поглощаемые или дублирующие члены. В результате получим выражение $\vee \&'$, являющееся частью выражения $\vee \&$. Каждое слагаемое в $\vee \&'$ будет определять неприводимое покрытие, соответствующее тупиковой ДНФ

Пример 5.5. Для функции $f(x_1, x_2, x_3) = (11011011)$ множество N_f состоит из 6 вершин, которые занумеруем числами I, II, …, VI. Максимальными гранями являются ребра, которые занумеруем арабскими цифрами (рис. 5.4).

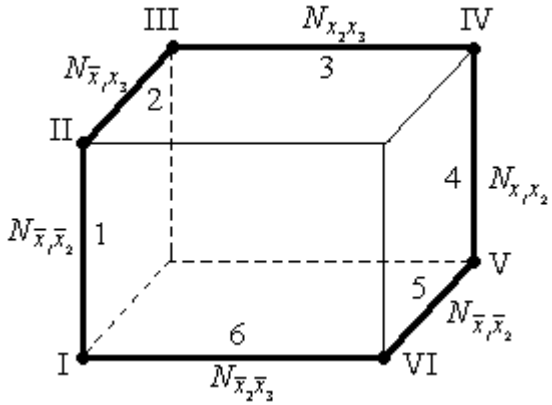


Рис. 5.4

Составим таблицу 5.5 для множеств E_j ($j = 1, \ ..., 6$):

$$E_{\text{I}} = \{1, 6\}, \ E_{\text{II}} = \{1, 2\}, E_{\text{III}} = \{2, 3\},$$

$$E_{\text{IV}} = \{3, 4\}, E_V = \{4, 5\}, E_{\text{VI}} = \{5, 6\}.$$

Таблица 5.5

	I	II	III	IV	V	VI
1	1	1	0	0	0	0
2	0	1	1	0	0	0
3	0	0	1	1	0	0
4	0	0	0	1	1	0
5	0	0	0	0	1	1
6	1	0	0	0	0	1

Тогда

$$\vee\& = (1 \vee 6) \cdot (1 \vee 2) \cdot (2 \vee 3) \cdot (3 \vee 4) \cdot (4 \vee 5) \cdot (5 \vee 6) =$$

$$= (1 \vee 2 \cdot 6) \cdot (3 \vee 2 \cdot 4) \cdot (5 \vee 4 \cdot 6) =$$

$$= (1 \cdot 3 \vee 2 \cdot 3 \cdot 6 \vee 1 \cdot 2 \cdot 4 \vee 2 \cdot 4 \cdot 6) \cdot (5 \vee 4 \cdot 6) =$$

$$= 1 \cdot 3 \cdot 5 \vee 2 \cdot 3 \cdot 5 \cdot 6 \vee 1 \cdot 2 \cdot 4 \cdot 5 \vee 2 \cdot 4 \cdot 5 \cdot 6 \vee 1 \cdot 3 \cdot 4 \cdot 6 \vee$$

$$\vee 2 \cdot 3 \cdot 4 \cdot 6 \vee 1 \cdot 2 \cdot 4 \cdot 6 \vee 2 \cdot 4 \cdot 6 =$$

$$= 1 \cdot 3 \cdot 5 \vee 2 \cdot 3 \cdot 5 \cdot 6 \vee 1 \cdot 2 \cdot 4 \cdot 5 \vee 1 \cdot 3 \cdot 4 \cdot 6 \vee 2 \cdot 4 \cdot 6.$$

Получили пять неприводимых покрытий или пять тупиковых ДНФ:

$$R_1 = \overline{x}_2 \cdot \overline{x}_3 \vee x_2 \cdot x_3 \vee x_1 \cdot \overline{x}_3,$$

$$R_2 = \overline{x}_1 \cdot x_3 \vee x_2 \cdot x_3 \vee x_1 \cdot \overline{x}_3 \vee \overline{x}_2 \cdot \overline{x}_3,$$

$$R_3 = \overline{x}_1 \cdot \overline{x}_2 \vee \overline{x}_1 \cdot x_3 \vee x_1 \cdot x_2 \vee x_1 \cdot \overline{x}_3,$$

$$R_4 = \overline{x}_1 \cdot \overline{x}_2 \vee x_2 \cdot x_3 \vee x_1 \cdot x_2 \vee \overline{x}_2 \cdot \overline{x}_3,$$

$$R_5 = \overline{x}_1 \cdot x_3 \vee x_1 \cdot x_2 \vee \overline{x}_2 \cdot \overline{x}_3,$$

из которых R_1 и R_5 являются минимальными.

5.1.3. Методы минимизации булевых функций

Минимизация булевых функцийна основе построения тупиковых ДНФ. Сокращенная, тупиковая и минимальная ДНФ находятся в следующем соотношении: ту-

пиковая ДНФ получается из сокращенной путем удаления некоторых членов; минимальная ДНФ является тупиковой; среди тупиковых ДНФ найдется минимальная.

Отсюда процесс построения минимальных ДНФ, если исходить из совершенной ДНФ, можно представить следующим образом. Сначала получают сокращенную ДНФ При этом на данном шаге возможно усложнение ДНФ Далее однозначный процесс переходит в ветвящийся – процесс построения всех тупиковых ДНФ. Наконец, из тупиковых ДНФ выделяются минимальные.

Минимизация булевых функций методом карт Карно. При использовании этого метода производится покрытие функций алгебры логики (ФАЛ) с помощью правильных конфигураций, содержащих нули или единицы. Правильными конфигурациями на карте Карно для ФАЛ от n переменных являются все прямоугольники (горизонтальные, вертикальные, квадраты), имеющие площадь 2^{n-i} $(i = 0, 1, ..., n)$. При этом стремятся, чтобы число покрытий ФАЛ на карте было минимально, а площадь, покрываемая каждой конфигурацией, – максимальна. Конфигурации могут перекрываться. Принцип минимизации заключается в объединении соседних полей карты в пределах правильной конфигурации.

При нахождении минимальной формы ФАЛ выписываются переменные, не изменяющие своего значения в пределах правильной конфигурации. При объединении полей, в которых записаны единицы, ФАЛ записывается в виде ДНФ, а при объединении полей, содержащих нули, – в виде КНФ.

Например, функция от четырех переменных $f_1(\tilde{x}^4)$ компактно записывается в форме матрицы размера $[4 \times 4]$, как это показано на рис. 5.5.

Каждой функции сопоставляется подмножество клеток, в которых эта функция равна единице. При этом элементарным конъюнкциям соответствуют некоторые правильно расположен-

ные конфигурации клеток. Для функции n переменных конъюнкции ранга r соответствует 2^{n-r} клеток.

x_1x_2	x_3x_4			
	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	0	1	1	0
10	1	1	1	1

Рис. 5.5. Карта Карно

Для функции от четырех переменных имеем.

1) Конъюнкции ранга 4 соответствует одна клетка:

x_1x_2	x_3x_4			
	00	01	11	10
00			■	
01				
11				
10				

$\overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \cdot x_4$

x_1x_2	x_3x_4			
	00	01	11	10
00				
01				
11		■		
10				

$x_1 \cdot x_2 \cdot \overline{x}_3 \cdot x_4$

2) Конъюнкции ранга 3 соответствует две соседние клетки:

x_1x_2	x_3x_4			
	00	01	11	10
00				
01		■	■	
11				
10				

$\overline{x}_1 \cdot x_2 \cdot x_4$

x_1x_2	x_3x_4			
	00	01	11	10
00			■	
01				
11				
10			■	

$\overline{x}_2 \cdot x_3 \cdot x_4$

3) Конъюнкции ранга 2 соответствуют 4 клетки, образующие горизонтальный или вертикальный ряд, либо подматрицу размера $[2\times 2]$:

x_1x_2	x_3x_4			
	00	01	11	10
00				
01	■	■	■	■
11				
10				

$\overline{x}_1 \cdot x_2$

x_1x_2	x_3x_4			
	00	01	11	10
00			■	
01			■	
11			■	
10			■	

$x_3 \cdot x_4$

x_1x_2	x_3x_4			
	00	01	11	10
00				
01		■	■	
11		■	■	
10				

$x_2 \cdot x_4$

x_1x_2	x_3x_4			
	00	01	11	10
00	■			■
01				
11				
10	■			■

$\overline{x}_2 \cdot \overline{x}_4$

4) Конъюнкции ранга 1 соответствуют 8 клеток из двух соседних горизонтальных или вертикальных рядов:

x_1x_2	x_3x_4			
	00	01	11	10
00	■	■		
01	■	■		
11	■	■		
10	■	■		

$\overline{x}_3$

x_1x_2	x_3x_4			
	00	01	11	10
00				
01	■	■	■	■
11	■	■	■	■
10				

x_2

Пример 5.6. Для функции $f_1(\tilde{x}^4)$ (см. рис. 5.5) разобьем множество единичных клеток на следующие подмножества:

x_1x_2	x_3x_4			
	00	01	11	10
00	1			1
01	1			1
11				
10				

$\overline{x}_1 \cdot \overline{x}_4$

x_1x_2	x_3x_4			
	00	01	11	10
00		1		
01		1		
11		1		
10		1		

$\overline{x}_3 \cdot x_4$

x_1x_2	x_3x_4			
	00	01	11	10
00				
01				
11		1	1	
10		1	1	

$x_1 \cdot x_4$

x_1x_2	x_3x_4			
	00	01	11	10
00	1			1
01				
11				
10	1			1

$\overline{x}_2 \cdot \overline{x}_4$

Объединение этих подмножеств дает все единичные клетки функции $f_1(\tilde{x}^4)$. Поэтому

$$f_1(\tilde{x}^4) = \overline{x}_1 \cdot \overline{x}_4 \vee \overline{x}_3 \cdot x_4 \vee x_1 \cdot x_4 \vee \overline{x}_2 \cdot \overline{x}_4 .$$

Минимизация булевых функций методом Квайна-Мак-Класски. Данный метод основывается на представлении элементарных конъюнкций, входящих в совершенную ДНФ данной функции, в виде двоичных чисел, называемых номерами соответствующих наборов. Кроме номера, каждой элементарной конъюнкции присваивается определенный индекс, равный числу единиц в двоичном представлении данного набора. Например:

элементарная конъюнкция $\overline{x}_1 \cdot x_2 \cdot \overline{x}_3$: номер 010 (2); *индекс* 1;

элементарная конъюнкция $x_1 \cdot x_2 \cdot \overline{x}_3$: номер 110 (6); *индекс* 2.

В результате реализации данного метода функция разлагается на простые импликанты. Алгоритм Квайна-Мак-Класски формулируется следующим образом: для того, чтобы два числа m и n являлись номерами двух склеивающихся между собой наборов, необходимо и достаточно, чтобы индексы данных чи-

сел отличались на единицу, сами числа отличались на степень числа 2 и число с большим индексом было больше числа с меньшим индексом.

Пример 5.6. Реализацию алгоритма рассмотрим на примере минимизации функции

$$f(\tilde{x}^4) = \overline{x}_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \cdot x_4 \vee \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \cdot x_4 \vee x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \cdot x_4 \vee$$
$$\vee \overline{x}_1 \cdot x_2 \cdot x_3 \cdot x_4 \vee x_1 \cdot \overline{x}_2 \cdot x_3 \cdot x_4 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \cdot x_4.$$

На первом этапе минимизации определяем номера и индексы каждого набора, записывая функцию в виде

$$f_2(\tilde{x}^4) = \underset{1(\mathrm{I})}{0001} \vee \underset{5(\mathrm{II})}{0101} \vee \underset{9(\mathrm{II})}{1001} \vee \underset{7(\mathrm{III})}{0111} \vee \underset{11(\mathrm{III})}{1011} \vee \underset{3(\mathrm{II})}{0011}.$$

На втором этапе группируем наборы, располагая их в порядке возрастания индексов (табл. 5.6).

Таблица 5.6. Минимизация ФАЛ методом Квайна-Мак-Класски

Индекс	Номер	Результаты склеивания	
I	0001(1)	00-1 0-01 -001	0--1 -0-1 0--1
II	0011(3) 0101(5) 1001(9)	0-11 -011 01-1	-0-1
III	0111(7) 1011(11)	10-1	

На третьем этапе производим склеивание различных наборов, руководствуясь приведенной выше формулировкой алгоритма. При склеивании не совпадающие в числах разряды отмечаются прочерками. Например, склеивание чисел 0001 и 0011 дает число 00-1. Результат склеивания записывается в следующий столбец таблицы 5.6, также разделяемый на строки с индексами, отличающимися на единицу. После склеивания всех групп первого столбца таблицы переходят ко второму, записывая результат склеивания в третий столбец. При объединении второго и последующих столбцов возможно склеивать только числа, содержащие прочерки в одноименных разрядах. Склеи-

вание продолжается до тех пор, пока образование нового столбца станет невозможно.

На четвертом этапе после окончания склеивания приступают к построению импликантной таблицы (табл. 5.7), записывая в нее в качестве простых импликант наборы, содержащиеся в последнем столбце таблицы 5.6. В таблицу 5.7 также вписываются в качестве простых импликант наборы из других столбцов таблицы 5.6, не принимавшие участия в склеивании. Если импликанта, содержащаяся в i-ой строке таблицы, составляет часть конституенты j-го столбца, то на пересечении i-ой строки и j-го столбца ставится символ *. Для получения минимальной формы ФАЛ из таблицы 5.6 необходимо выбрать минимальное число строк, чтобы для каждого столбца среди выбранных строк нашлась хотя бы одна, содержащая в этом столбце символ *.

Таблица 5.7. Импликантная таблица минимизируемой функции

Импликанты	Наборы					
	0001	0101	1001	0111	1011	0011
$\overline{x}_1 \cdot x_4$	*	*		*		*
$\overline{x}_2 \cdot x_4$	*		*		*	*

В результате минимизации функция $f(\tilde{x}^4)$ представится в виде

$$f(\tilde{x}^4) = \overline{x}_1 \cdot x_4 \vee \overline{x}_2 \cdot x_4 .$$

5.2. Схемы из функциональных элементов

5.2.1. Понятие схемы из функциональных элементов

В современной технике управляющих и вычислительных устройств важное место занимают *дискретные преобразователи*, т. е. устройства, которые обладают некоторым числом входов и выходов. Наборы сигналов, поступающие на входы и возникающие на выходах, принадлежат известным конечным мно-

жествам. Устройства осуществляют преобразования входных наборов сигналов в выходные. Математической моделью таких устройств являются так называемые *схемы из функциональных элементов* (СФЭ).

В качестве примера рассмотрим электрическую схему из трех диодов и сопротивления, показанную на рис. 5.5.

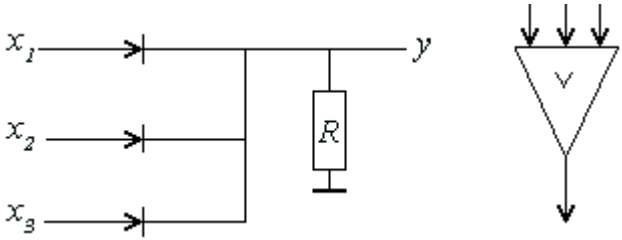


Рис. 5.5. Электрическая схема и ее условное обозначение

В точках схемы, изображенных кружком, в различные моменты времени возможно появление либо высокого уровня, приблизительно равного 5 В, либо низкого уровня, приблизительно равного нулю. В точке схемы, отмеченной черточкой, поддерживается постоянно низкий уровень напряжения. Точки, отмеченные x_1, x_2, x_3, будем интерпретировать как входы, а точку y – как выход. Работу схемы можно описать следующим образом: если на всех входах низкий уровень напряжения, то на выходе тоже низкий, если хотя бы на одном из входов высокий уровень напряжения, то на выходе – высокий. Если обозначить состояние с высоким уровнем напряжения единицей, а с низким – нулем, то зависимость выхода от входов можно задать при помощи булевой функции $y = x_1 \vee x_2 \vee x_3$. На основании этого приведенную схему называют логическим элементом «ИЛИ».

Подобные схемы можно построить из электронных ламп, электромеханических переключателей, пневмоэлементов и др. Зависимость выхода от входов может описываться не только как дизъюнкция, но также при помощи конъюнкции, отрицания и более сложных булевых функций.

Будем рассматривать логические элементы с различной зависимостью выхода от входов. Эти элементы можно соединять друг с другом, подавая выходы некоторых элементов на входы других. В результате получаем СФЭ.

Определение понятия СФЭ можно разбить на два этапа. На первом этапе раскрывается структурная часть этого понятия, на втором – функциональная.

I э т а п . Разобьем этот этап на ряд пунктов.

1°. Имеется конечное множество F объектов F_i ($i = 1, ..., r$), называемых *логическими элементами.* Каждый элемент имеет n_i входов и один выход. Элемент F_i графически изображается так, как указано на рис. 5.6.

2°. По индукции определяем понятие *логической сети* Σ как объекта, в котором имеется некоторое число n входов и некоторое число p выходов (рис. 5.7).

а) Б а з и с и н д у к ц и и . Изолированная вершина называется тривиальной логической сетью. По определению, она является одновременно входом и выходом (рис. 5.8).

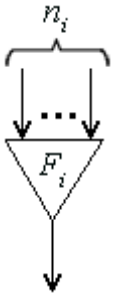


Рис. 5.6

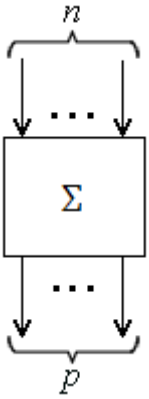


Рис. 5.7

Рис. 5.8

б) И н д у к т и в н ы й п е р е х о д . Эта часть основана на использовании трех операций.

I°. О п е р а ц и я о б ъ е д и н е н и я н е п е р е с е к а ю щ и х с я с е т е й . Пусть Σ' и Σ'' – две непересекающиеся сети

(без общих элементов, входов и выходов), имеющие соответственно n и m входов и p и q выходов. Теоретико-множественное объединение сетей Σ' и Σ'' есть логическая сеть Σ_1, которая имеет $n+m$ входов и $p+q$ выходов.

II°. Операция присоединения элемента F_i. Пусть сеть Σ' и элемент F_i таковы, что $n_i \le p$ и в Σ' выбрано n_i различных выходов с номерами $j_1, ..., j_{n_i}$. Тогда фигура Σ_2 называется логической сетью, являющейся результатом подключения элемента F_i к сети Σ'. Входами Σ_2 являются все входы Σ', выходами – все выходы сети Σ', кроме выходов с номерами $j_1, ..., j_{n_i}$, а также выход элемента F_i. Сеть Σ_2 имеет n входов и $p-n_i+1$ выходов (рис. 5.9).

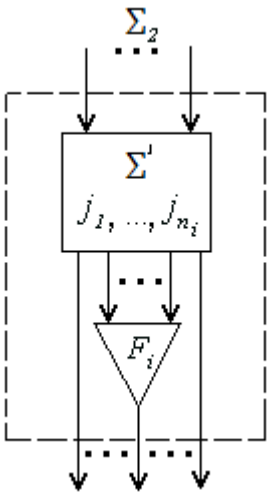


Рис. 5.9

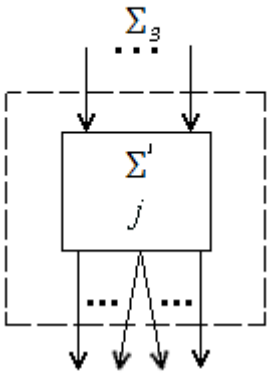


Рис. 5.10

III°. Операция расщепления выхода. Пусть в сети Σ' выделен выход с номером j. Тогда фигура Σ_3 называется логической сетью, полученной путем расщепления выхода j. Входами Σ_3 являются все входы Σ', выходами – все выходы

сети Σ' с номерами 1, …, $j-1$, $j+1$, …, p и еще два выхода, возникших из выхода с номером j сети Σ' (рис. 5.10). Следовательно, Σ_3 имеет n входов и $p+1$ выходов.

3°. Пусть заданы алфавиты $X=\{x_1, ..., x_n\}$ и $Y=\{y_1, ..., y_p\}$.

Схемой из функциональных элементов называется логическая сеть с входами $x_{i_1}, ..., x_{i_n}$ из алфавита X и выходами $y_{j_1}, ..., y_{j_p}$ из алфавита Y, которая обозначается

$$\Sigma(x_{i_1}, ..., x_{i_n}; y_{j_1}, ..., y_{j_p}). \qquad (5.1)$$

Приведем примеры схем.

1. Пусть множество F состоит из трех элементов И (конъюнктора), ИЛИ (дизъюнктора) и НЕ (инвертора).

Тогда фигура Σ_1 (рис. 5.11) будет схемой, так как она может быть построена с использованием операций I° – III°.

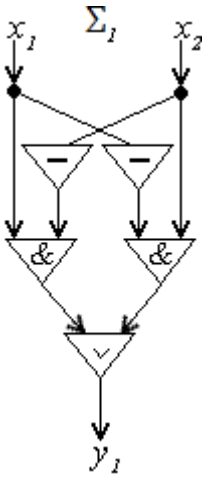


Рис. 5.11

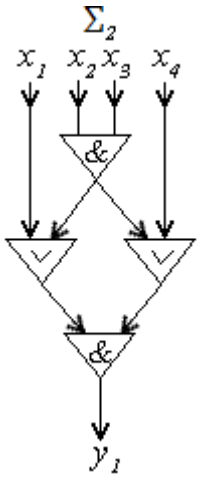


Рис. 5.12

2. Фигура, изображенная на рис. 5.12, будет также схемой.

II э т а п . Определение функционирования схемы.

4°. Сопоставим СФЭ (1) систему функций алгебры логики

$$\begin{cases} y_1 = f_1(x_1, \ ..., x_n), \\ ..... \\ y_p = f_p(x_1, \ ..., x_n), \end{cases} \qquad (5.2)$$

называемую также *проводимостью данной схемы*.

Пример 5.7. а) Для схемы Σ_1 имеем систему, состоящую из одного уравнения

$$y_1 = x_1 \cdot \overline{x}_2 \vee \overline{x}_1 \cdot x_2 \text{ или } y_1 = x_1 \oplus x_2 .$$

б) Для схемы Σ_2 аналогично получаем

$$y_1 = x_1 \cdot x_4 \vee x_2 \cdot x_3 .$$

5.2.2. Реализация булевых функций схемами из ФЭ. Задачи анализа и синтеза

Задача анализа: для данной СФЭ (5.1) получить систему булевых уравнений (5.2).

Алгоритм решения задачи: следуя операциям построения сети I – III, последовательно вычисляем функции на выходах элементов сети.

Задача синтеза: для данного базиса F функциональных элементов и произвольной системы булевых уравнений (5.2) построить схему (5.1) из заданных ФЭ, реализующую эту систему уравнений.

Существование решения задачи синтеза определяется теоремой Поста, согласно которой система функций $(f_1, \ ..., f_r)$, реализующих базисные ФЭ, должна быть полна. Функции $y_i \ (i = 1, \ ..., p)$ можно представить в виде суперпозиции функций $(f_1, \ ..., f_r)$, а каждому шагу суперпозиции соответствует определенное соединение элементов.

Пример 5.8. Для функции

$$f(x_1, \ x_2, \ x_3) = f_1(f_1(x_1, \ x_2, \ x_3), \ f_2(x_2, \ x_3), \ f_3(x_3), x_3) \qquad (5.3)$$

схема, соответствующая суперпозиции в правой части формулы (5.3), показана на рис. 5.13.

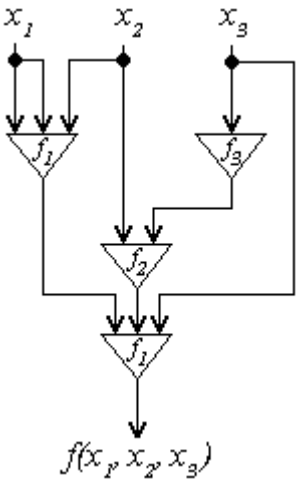


Рис. 5.13

Проблема синтеза заключается в том, что для данной системы булевых уравнений можно построить много схем из ФЭ, которые реализуют эту систему. В связи с этим возникает задача оптимального синтеза: из всевозможных схем, реализующих данную функцию, выбрать наилучшую по тому или иному признаку, например, с наименьшим числом элементов. Такие схемы будем называть *минимальными*.

Справедливо следующее утверждение.

Теорема 5.3. *Существует алгоритм, который для каждой системы булевых функций строит минимальную схему* Σ.

Данный алгоритм построения минимальных схем относится к классу алгоритмов типа «полного перебора», так как он основан на просмотре всех схем до определенной сложности. Алгоритмы полного перебора, как правило, обладают большой трудоемкостью и непригодны для практических целей. Поэтому рассмотрим далее более простую задачу, для которой исходная система уравнений содержит одно уравнение $y = f(x_1, ..., x_n)$, и, следовательно, искомая схема имеет один выход.

Сложность минимальной схемы обозначим через $L(f)$. Будем рассматривать задачу синтеза не для отдельной функции

f, а для всего класса P_2 функций от n переменных. Качество алгоритмов синтеза сравнивается путем сопоставления так называемых функций Шеннона. Пусть

$$L(n) = \max_{f \in P_2} L(f),\ L_A(f)\ L_A(n) = \max_{f \in P_2} L_A(f),$$

где $L_A(f)$ минимальная сложность схем, реализующих f, которые получаются при помощи алгоритма A.

Функции $L(n)$, $L_A(n)$ называются *функциями Шеннона*, причем очевидно, что

$$L_A(n) \geq L(n).$$

Задача синтеза состоит в том, чтобы найти алгоритм A, для которого $L_A(n)$ была бы воз можно ближе к $L(n)$, и чтобы трудоемкость алгоритма A была бы существенно меньше, чем трудоемкость алгоритма полного перебора. При такой постановке задачи не требуется, чтобы алгоритм A для каждой функции f находил минимальную схему, необходимо только, чтобы простейшая схема, получаемая при помощи A, имела сложность $L_A(f)$, сильно не превышающую $L(n)$.

5.2.3. Элементарные методы синтеза. Синтез дешифратора и сумматора

Рассмотрим несколько алгоритмов синтеза, использующих классический базис, состоящий из инвертора, дизъюнктора и конъюнктора.

1°. Метод синтеза, основанный на совершенной ДНФ.

Рассмотрим разложение функции $f(x_1, ..., x_n) \neq \text{const}$ в виде совершенной ДНФ:

$$f(x_1, ..., x_n) = \bigvee_{\substack{(\sigma_1,...,\sigma_n) \\ f(\sigma_1,...,\sigma_n)=1}} x_1^{\sigma_1} \& ... \& x_n^{\sigma_n} = \bigvee_{i=1}^{s} K_i.$$

Введем вспомогательный элемент (рис. 5.14), с помощью которого построим схему (рис. 5.15) Σ_K, реализующую конъюнкцию $K = x_1^{\sigma_1} \& ... \& x_n^{\sigma_n}$.

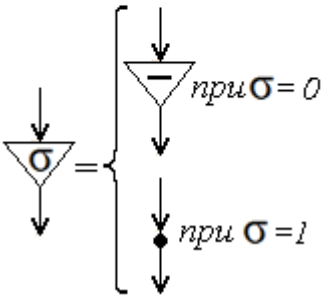


Рис. 5.14

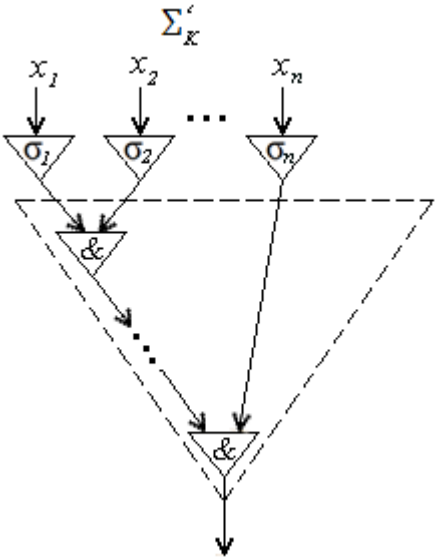


Рис. 5.15

Очевидно, $L(\Sigma_K) \le n + (n-1)$, и Σ_K содержит подсхему Σ'_K, одинаковую для всех конъюнкций и имеющую сложность $n-1$. Если «склеить» схемы Σ_{K_1}, ..., Σ_{K_s}, начиная от входов x_1, ..., x_n вплоть до вспомогательных элементов, то получим схему Σ, для которой $L(\Sigma_K) \le n + s \cdot (n-1)$. Подключая выходы схемы Σ к схеме из дизъюнкторов, мы осуществим синтез схемы для $f(x_1, ..., x_n)$ (рис. 5.16) по совершенной ДНФ (алгоритм A_1).

Сложность этого алгоритма

$$L_{A_1}(n) \le n + s \cdot (n-1) + s - 1 < n + n \cdot s = n \cdot (s+1).$$

Поскольку $f \ne 1$, то $s \le 2^n - 1$ и $L_{A_1}(n) \le n \cdot 2^n$.

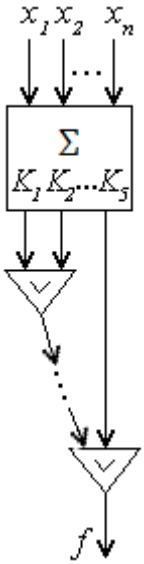


Рис. 5.16

Пример 5.9. Построить схему, реализующую функцию $f(\tilde{x}^3) = (01010100)$.

Представим данную функцию формулой в базисе $\sigma = \{\vee, \&, \neg\}$, используя, например, совершенную ДНФ:

$$f(\tilde{x}^3) = \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \vee \bar{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \cdot x_3 . \qquad (5.4)$$

Для каждой логической операции в этой формуле возьмем соответствующие функциональные элементы и произведем их соединение так, как этого требует формула. В результате получим схему, показанную на рис. 5.17. Эта схема использует 10 элементов. Предварительное упрощение формулы (5.4)

$$f(\tilde{x}^3) = \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \vee \bar{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \cdot x_3 = (\bar{x}_1 \vee x_1) \cdot \bar{x}_2 \cdot x_3 \vee$$

$$\vee \bar{x}_1 \cdot (\bar{x}_2 \vee x_2) \cdot x_3 = \bar{x}_2 \cdot x_3 \vee \bar{x}_1 \cdot x_3 = x_3 \cdot (\bar{x}_1 \vee \bar{x}_2) = \overline{x_1 \cdot x_2} \cdot x_3$$

позволяет для той же функции построить более простую схему (рис. 5.18).

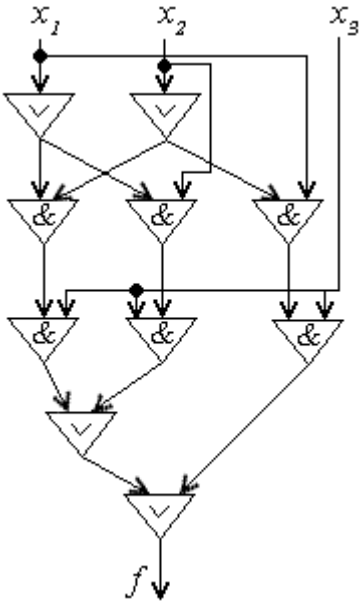


Рис. 5.17

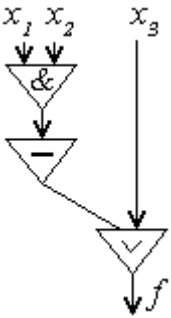


Рис. 5.18

2°. Метод синтеза, основанный на более компактной реализации множества всех конъюнкций $\{x_1^{\sigma_1} \& \ldots \& x_n^{\sigma_n}\}$. На рис. 5.19 представлено индуктивное построение многополюсника Σ^n ($n = 1, 2, \ldots$), реализующего множество всех конъюнкций $\{x_1^{\sigma_1} \& \ldots \& x_n^{\sigma_n}\}$. Такая схема, имеющая n входов $x_1, \ldots, x_n$ и 2^n выходов, на которых реализуются всевозможные элементарные конъюнкции ранга n, называется *дешифратором*. При подаче на входы дешифратора какой-либо комбинации нулей и единиц единичный сигнал появляется только на одном из выходов, остальные выходы находятся в нулевом состоянии. В ЭВМ дешифратор применяется для записи или считывания информации из памяти: на вход подается двоичный адрес определенной ячейки памяти, это вызывает появление единичного сигнала ровно на одном из выходов, который связан с соответствующей ячейкой, что приводит к операции считывания-записи именно для этой ячейки.

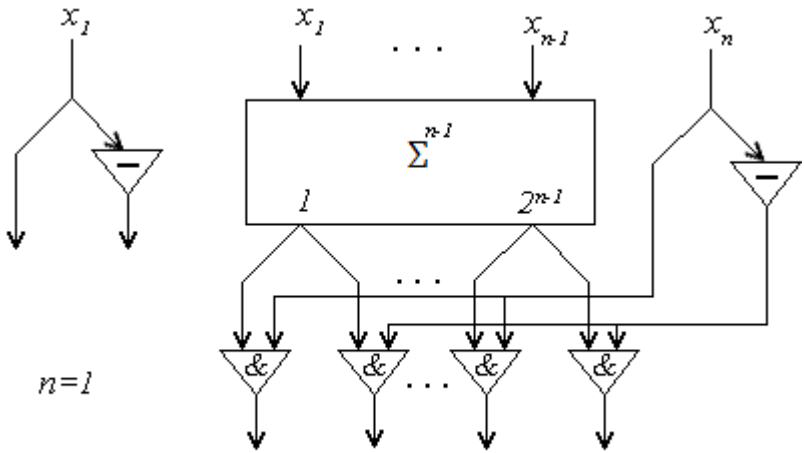


Рис. 5.19

Имеем

$$L(\Sigma^1) = 1,$$

$$L(\Sigma^n) = L(\Sigma^{n-1}) + 1 + 2^n,$$

$$L(\Sigma^n) = 2^n + ... + 2^n + n = 2 \cdot 2^n + n - 4.$$

Для построения схемы, реализующей функцию $f(x_1, ..., x_n)$, нужно в многополюснике Σ^n отобрать выходы, соответствующие членам ее совершенной ДНФ $K_1, ..., K_s$, подключить их к схеме (см. рис. 5.16), осуществляющей логическое сложение, и удалить лишние элементы. Это потребует не более $s \le 2^n - 1$ элементов $\vee$. Таким образом, этот метод (алгоритм A_2) дает

$$L_{A_2}(n) \le 3 \cdot 2^n + n - 5.$$

3°. Метод синтеза, основанный на разложении функции $f(x_1, ..., x_n)$ по переменной x_n.

Возьмем разложение

$$f(x_1, ..., x_{n-1}, x_n) = x_n \,\&\, f(x_1, ..., x_{n-1}, 1) \vee \overline{x}_n \,\&\, f(x_1, ..., x_{n-1}, 0)$$

и для краткости положим

$$f' = f(x_1, \ldots, x_{n-1}, 1), \; f'' = f(x_1, \ldots, x_{n-1}, 0).$$

На рис. 5.20 представлена индуктивная процедура построения схемы для f.

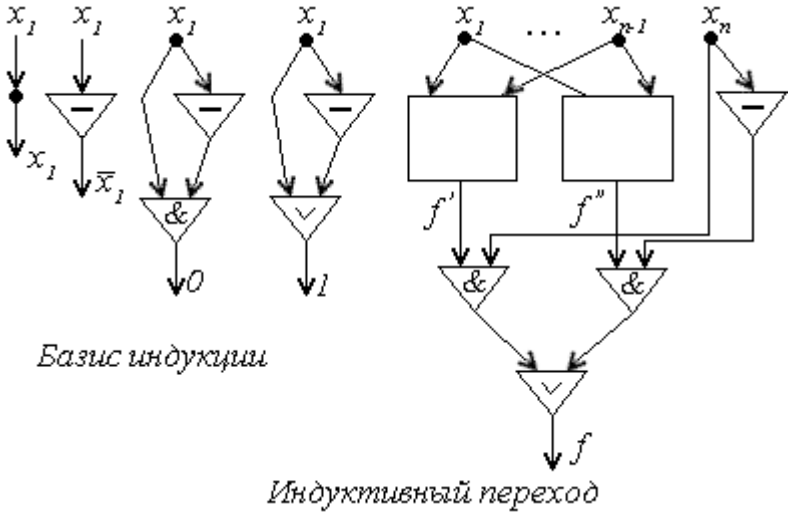


Рис. 5.20

На основании этого метода имеем алгоритм A_3:

$$L_{A_3}(1) = 2, \; L_{A_3}(n) = 2 \cdot L_{A_3}(n-1) + 4.$$

Окончательно получим

$$L_{A_3}(n) \le 3 \cdot 2^n - 4.$$

Построенные алгоритмы A_1, A_2 и A_3 дают возможность получить все более компактные реализации для функций и, в конечном счете, все более хорошие оценки для функций Шеннона. С другой стороны, получение более хороших результатов синтеза достигается за счет некоторого усложнения алгоритма.

Общая теория синтеза СФЭ приводит к выводу о том, что большинство булевых функций $f(\tilde{x}^n)$ при больших значениях n имеет сложные минимальные схемы. Это означает, что практическую ценность с точки зрения синтеза представляет весьма узкий класс булевых функций. Поэтому наряду с универсальными методами синтеза необходимо иметь методы синтеза, при-

способленные к отдельным классам булевых функций, полнее учитывающие свойства отдельных функций.

Рассмотрим схему двоичного сумматора, имеющуюся в каждом компьютере.

Двоичный сумматор – это схема, реализующая сложение двух целых чисел, заданных в двоичной системе счисления:

$$x = x_n x_{n-1} ... x_1 , \; y = y_n y_{n-1} ... y_1 .$$

Условное обозначение схемы сумматора показано на рис. 5.21.

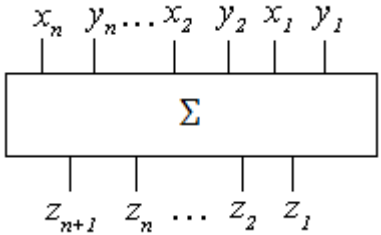


Рис. 5.21

Рассмотрим хорошо известный алгоритм сложения чисел x и y «столбиком»:

$$\begin{array}{r} (q_{n+1} q_n q_{n-1} ... q_1) \\ x_n x_{n-1} ... x_1 \\ + \\ y_n y_{n-1} ... y_1 \\ \hline z_{n+1} z_n z_{n-1} ... z_1 . \end{array}$$

Здесь числа q_{n+1}, q_n, ..., q_1 обозначают результаты переносов из предыдущих разрядов $(q_1 = 0)$. Очевидно, $z_i = x_i \oplus y_i \oplus q_i$, $q_{i+1} = x_i \cdot y_i \vee x_i \cdot q_i \vee y_i \cdot q_i$. Основываясь на тождестве

$$x_i \oplus y_i \oplus q_i = \overline{x_i \cdot y_i \vee x_i \cdot q_i \vee y_i \cdot q_i} \cdot (x_i \vee y_i \vee q_i) \vee x_i \cdot y_i \cdot q_i ,$$

получаем схему, реализующую соответствующее преобразование величин x_i, y_i, q_i в z_i, q_{i+1} (рис. 5.22). Обозначим эту схему через B_i $(i < i \le n)$. Тогда искомая схема Σ_n получается путем последовательного соединения блоков B_i (рис. 5.23). Здесь $z_{n+1} = q_{n+1}$, и блок B_1 осуществляет преобразование

$$z_1 = x_1 \oplus y_1 = \overline{x_1 \cdot y_1} \cdot (x_1 \vee y_1),\ q_2 = x_1 \cdot y_1 .$$

Очевидно, $L(B_1) = 4$ и $L(B_i) = 9$ при $i < i \le n$. Тогда

$$L(\Sigma_n) \le 9 \cdot (n-1) + 4 = 9 \cdot n - 5 < 9 \cdot n .$$

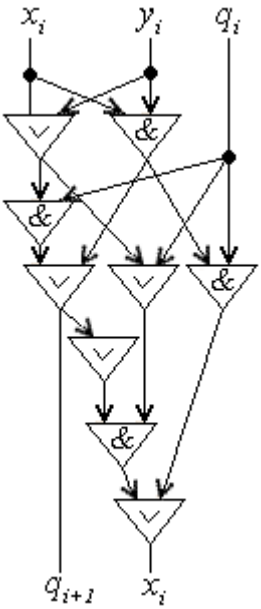


Рис. 5.22

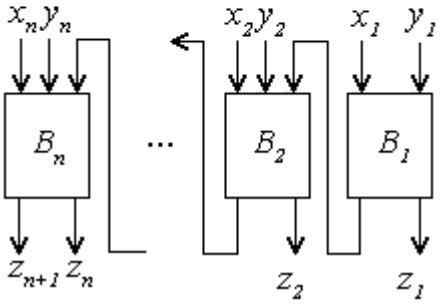


Рис.5.23

6. Конечные автоматы

6.1. Определение и способы задания конечного автомата. Задача синтеза автоматов

СФЭ не учитывают тот факт, что реальные устройства работают во времени. По сравнению со СФЭ конечный автомат является более точной моделью дискретного преобразователя информации. При этом понятие конечного автомата, как и любая модель, связано с рядом упрощающих предположений.

Во-первых, предполагается, что вход и выход автомата в каждый момент времени может находиться только в одном из конечного числа различных состояний. Если реальный преобразователь имеет непрерывный входной сигнал, то для его описания с помощью конечного автомата необходимо провести квантование этого сигнала.

Во-вторых, предполагается, что время изменяется дискретно. Состояния входа и выхода соответствуют дискретной временной последовательности $t_1, t_2, ..., t_n, ...$ Поскольку момент времени однозначно определяется его индексом, то с целью упрощения будем считать, что время t принимает значения 1, 2, …, n, … Временной промежуток $[n, n+1]$ называется *тактом*.

Работа автомата представляется следующим образом.

На вход автомата поступают сигналы $x(1), ..., x(n), ...$ из входного алфавита X, что приводит к появлению сигналов на выходе $y(1), ..., y(n), ...$ из выходного алфавита Y. Зависимость выходной последовательности от входной зависит от внутреннего устройства автомата. Заметим, что в отличие от СФЭ, которые не обладают памятью, автомат представляет собой устройство с памятью, т. е. выход автомата $y(t)$ определяется не только входом $x(t)$, но и предысторией $H(t) = \{x(t-1), ..., x(1)\}$. Учет предыстории осуществляется зависимостью выходного сигнала не только от входа, но и от текущего состояния, которое обозначим $q(t)$.

Дадим формальное определение автомата.

Конечным автоматом (в дальнейшем – просто автоматом) называется система $A=(X,\ Y,\ Q,\ \delta,\ \lambda)$, в которой $X=\{x_1,\ ...,\ x_n\}$ – конечное множество, называемое *входным алфавитом*; $Y=\{y_1,\ ...,\ y_m\}$ – конечное множество, называемое *выходным алфавитом*; $Q=\{q_1,\ ...,\ q_r\}$ – конечное множество, называемое *алфавитом внутренних состояний*; δ – *функция переходов* автомата: $\delta\colon X\times Q\to Q$; эта функция каждой паре «вход-состояние» ставит в соответствие состояние; λ – *функция выходов* автомата: $\lambda\colon X\times Q\to Y$; эта функция каждой паре «вход-состояние» ставит в соответствие значение выхода.

Закон функционирования автомата: автомат изменяет свои состояния в соответствии с функцией δ и вырабатывает выходные сигналы в соответствии с функцией λ:

$$q(t+1)=\delta(x(t),\ q(t)),\ y(t)=\lambda(x(t),\ q(t)),\ t=1,\ 2,\ ...$$

Рассмотрим способы задания конечного автомата.

1°. Табличный способ задания. Поскольку для функций δ и λ области определения и значений принадлежат конечному множеству, то их задают при помощи таблиц.

Пример 6.1. Зададим автомат следующим образом: $X=\{x_1,\ x_2\ ,x_3\}$, $Y=\{y_1,\ y_2\}$, $Q=\{q_1,\ q_1,\ q_3\}$. Функцию δ определим с помощью *таблицы переходов* (табл. 6.1), а функцию λ – с помощью *таблицы выходов* (табл. 6.2).

Таблица 6.1. Таблица переходов

Вход	Состояние		
	q_1	q_2	q_3
x_1	q_2	q_2	q_3
x_2	q_1	q_3	q_3
x_3	q_1	q_2	q_1

Таблица 6. 2. Таблица выходов

Вход	Состояние		
	q_1	q_2	q_3
x_1	y_1	y_2	y_2
x_2	y_2	y_1	y_2
x_3	y_2	y_2	y_1

Если известна последовательность сигналов на входе автомата, то таблицами переходов и выходов однозначно определяется выходная последовательность.

2°. Графический способ задания. Используется *диаграмма переходов-выходов.* Она представляет собой ориентированный мультиграф, в котором каждому внутреннему состоянию автомата соответствует вершина. Переходы автомата из состояния в состояние изображаются стрелками, на каждой из которых пишутся входной символ, вызывающий данный переход, и выходной символ, вырабатываемый автоматом.

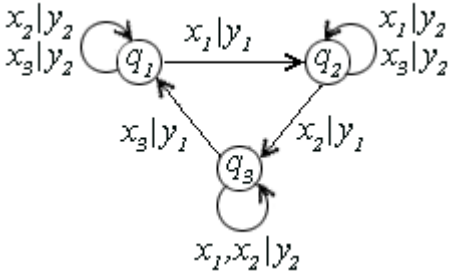


Рис. 6.1. Диаграмма переходов-выходов

Пример 6.2. Требуется построить автомат, который работал бы следующим образом: в каждый такт на вход автомата поступают очередные двоичные разряды слагаемых, автомат вырабатывает соответствующий двоичный разряд их суммы. Для двухразрядных слагаемых имеем: $X = \{00,\ 01,\ 10,\ 11\}$, $Y = \{0,\ 1\}$, $Q = \{0,\ 1\}$.

Автомат находится в состоянии 1, если при сложении предыдущих разрядов возникает перенос, и в состоянии 0 – в противном случае. Диаграмма переходов-выходов показана на рис. 6.2.

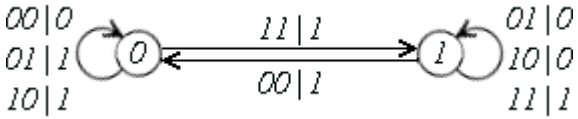


Рис. 6.2

По аналогии с задачей синтеза СФЭ можно поставить задачу синтеза для автоматов. Имеется неограниченный набор базисных автоматов $\{A_1, ..., A_r\}$. Требуется собрать автомат A с наперед заданным функционированием. При этом задача синтеза сталкивается с определенными проблемами. Допустим, что нужно присоединить выход автомата $A_1 = (X_1, Y_1, Q_1, \delta_1, \lambda_1)$ к входу автомата $A_2 = (X_2, Y_2, Q_2, \delta_2, \lambda_2)$. Это возможно при условии $Y_1 \subseteq X_2$, так как иначе второй автомат не поймет сигналы, поступающие с первого. Это приводит к запутанной ситуации, когда некоторые соединения невозможны. Чтобы преодолеть это препятствие, вводится понятие структурного автомата, в котором все алфавиты (входной, выходной и внутренних состояний) кодируются двоичными словами.

Пусть $M = \{a_1, ..., a_p\}$ – конечное множество из p элементов, а $\{0, 1\}^\pi$ – множество двоичных слов длины π, где $2^\pi \ge p$. Произвольное инъективное отображение $\phi: X \to \{0, 1\}^\pi$ будем называть *кодированием множества X двоичными словами.* Произведем кодирование алфавитов для произвольного автомата $A = (X, Y, Q, \delta, \lambda)$:

X	Y	Q
$x_1 \to \xi_{11}...\xi_{1\nu} = \tilde{\xi}_1$	$y_1 \to \eta_{11}...\eta_{1\mu} = \tilde{\eta}_1$	$q_1 \to \sigma_{11}...\sigma_{1\rho} = \tilde{\sigma}_1$
...	...	...
$x_n \to \xi_{n1}...\xi_{n\nu} = \tilde{\xi}_n$	$y_m \to \eta_{m1}...\xi_{m\mu} = \tilde{\eta}_\mu$	$q_r \to \sigma_{11}...\sigma_{r\rho} = \tilde{\sigma}_r$
$(2^\nu \ge n)$	$(2^\mu \ge m)$	$(2^\rho \ge r)$

Обозначим закодированные вход, выход и состояние автомата в момент времени t соответственно $\tilde{\xi}(t)$, $\tilde{\eta}(t)$, $\tilde{\sigma}(t)$. Тогда закон функционирования представится в виде

$$\begin{aligned} \tilde{\sigma}(t+1) &= \Delta(\tilde{\xi}(t),\ \tilde{\sigma}(t)), \\ \tilde{\eta}(t) &= \Lambda(\tilde{\xi}(t),\ \tilde{\sigma}(t)). \end{aligned} \qquad (6.1)$$

Полученный после кодирования автомат называют *структурным*. Будем считать, что структурный автомат имеет ν двоичных входов, μ двоичных выходов, а внутреннее состояние

автомата задается двоичным словом длины ρ. На рис. 6.3 показан *абстрактный* автомат A и соответствующий ему структурный автомат.

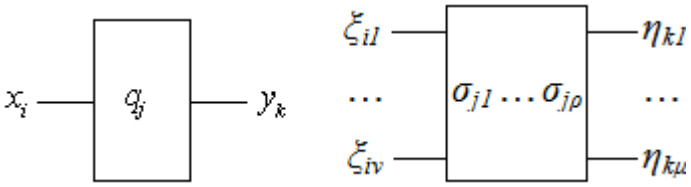


Рис. 6.3

Переход к структурному автомату обеспечивает два важных для синтеза преимущества.

1°. Совместимость входов и выходов, так как через них передается двоичная информация. Общее определение схемы из структурных автоматов аналогично определению СФЭ.

2°. Запишем соотношения (6.1) в «координатах»:

$$\begin{aligned}
&\sigma_1(t+1) = \Delta_1(\xi_1(t),\ \ldots,\ \xi_\nu(t),\ \sigma_1(t),\ \ldots,\ \sigma_\rho(t)),\\
&\qquad\qquad \ldots\\
&\sigma_\rho(t+1) = \Delta_\rho(\xi_1(t),\ \ldots,\ \xi_\nu(t),\ \sigma_1(t),\ \ldots,\ \sigma_\rho(t)),\\
&\eta_1(t) = \Lambda_1(\xi_1(t),\ \ldots,\ \xi_\nu(t),\ \sigma_1(t),\ \ldots,\ \sigma_\rho(t)),
\end{aligned} \qquad (6.2)$$

Из (6.2) следует, что *закон функционирования структурного автомата задается системой булевых функций*.

6.2. Элементарные автоматы. Задача о полноте автоматного базиса. Канонический метод синтеза автомата

Элементарные автоматы. Функциональный элемент, имеющий только одно состояние, можно рассматривать как автомат без памяти. Перейдем к автоматам с двумя состояниями. Пусть автомат с двумя состояниями имеет один двоичный вход ξ и один двоичный выход η, совпадающий с внутренним состоянием: $\eta = \sigma$ (рис. 6.4). Для такого автомата достаточно задать только таблицу переходов, в которой вместо звездочек нужно поставить 0 и 1 (табл. 6.3)

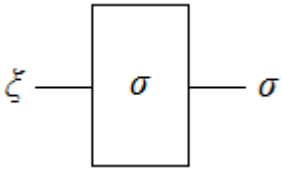


Рис. 6.4

Таблица 6.3

Вход	Состояние	
	0	1
0	*	*
1	*	*

Это можно сделать 16 способами, однако, не все они приемлемы. Для того чтобы получился автомат, не сводящийся к автомату без памяти, надо потребовать, чтобы в каждом столбце таблицы 6.3 встречались ноль и единица. Таких таблиц всего четыре.

Таблица 6.4

Вход	Состояние	
	0	1
0	1	0
1	0	1

Таблица 6.5

Вход	Состояние	
	0	1
0	1	1
1	0	0

Таблица 6.6

Вход	Состояние	
	0	1
0	1	0
1	0	1

Таблица 6.7

Вход	Состояние	
	0	1
0	1	1
1	0	0

Имеем только два простейших автомата, так как таблица 6.7 получается из 6.4, а 6.6 из 6.5 путем инверсии внутренних состояний.

Автомат, задаваемый таблицей 6.4, называется *задержкой* или *D-триггером*:

$$\eta(t+1) = \sigma(t+1) = \xi(t),$$

то есть этот автомат задерживает сигнал на один такт.

Автомат, задаваемый таблицей 6.5, называется *триггером со счетным входом* или *T -триггером*. Состояние автомата меняется на противоположное, если на вход поступает 1, и остается без изменения, если на вход поступает 0:

$$\eta(t+1) = \sigma(t+1) = \sigma(t) \oplus \xi(t).$$

Если в некоторый момент времени T -триггер находится в состоянии 0, то это означает, что на вход автомата поступило четное число единиц, если в состоянии 1, то – нечетное. Таким образом, T -триггер считает количество единиц на входе, но так как он имеет всего два состояния, то и считает до двух.

При физической реализации триггеров используют два выхода: *прямой* и *инверсный* (рис. 6.5). Если поменять их местами, то из D -триггера получится автомат, задаваемый таблицей 6.7, а из T -триггера – автомат, задаваемый таблицей 6.6.

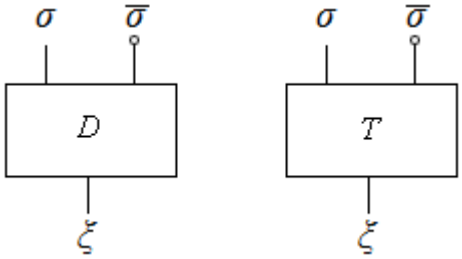


Рис. 6.5

Задача о полноте автоматного базиса. Набор структурных автоматов $\{A_1, ..., A_r\}$ называется *полным* (или *автоматным базисом*), если из них можно построить любой наперед заданный структурный автомат.

Усилия математиков для получения аналога теоремы Поста для автоматов не увенчались успехом. В 1964 г. М.И. Кратко доказал несуществование алгоритма для определения полноты системы $\{A_1, ..., A_r\}$. В этом случае представляют интерес вари-

анты теоремы о полноте с дополнительными предположениями о системе $\{A_1, ..., A_r\}$. Рассмотрим наиболее популярный из них.

Теорема 6.1. *Система автоматов* $\{A_1, ..., A_r\}$, *содержащая полный набор ФЭ и* D*-триггер (или* T*-триггер) является полной.*

Доказательство. Рассмотрим произвольный автомат A, заданный соотношениями (6.1), и опишем его схему из указанных автоматов, называемую *канонической структурой* (рис. 6.6). Схема состоит из двух частей.

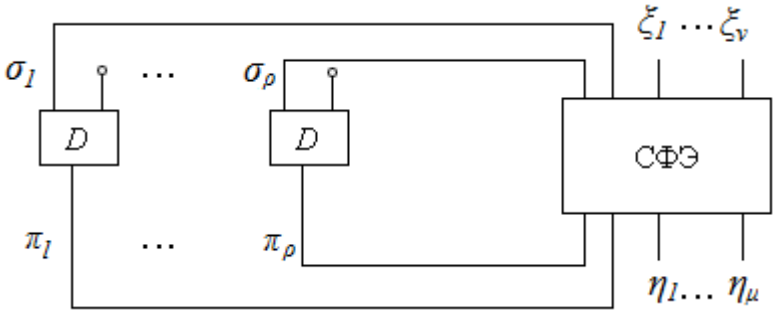


Рис. 6.6

Левая половина называется запоминающей частью. Она состоит из ρ триггеров, набор состояний которых образует состояние автомата: если в момент времени t

$$\sigma_1(t) = \sigma_{i1}, \ \dots, \ \sigma_\rho(t) = \sigma_{i\rho},$$

то это означает, что автомат A находится в состоянии $\tilde{\sigma}_i$.

Правая половина называется комбинационной частью и представляет СФЭ. Входы и выходы этой схемы:

1) двоичное слово $\tilde{\xi} = \xi_1 \dots \xi_\nu$ – вход автомата A;

2) двоичное слово $\tilde{\sigma} = \sigma_1 \dots \sigma_\rho$ – текущее внутреннее состояние автомата A;

3) двоичное слово $\tilde{\eta} = \eta_1 \dots \eta_\mu$ – выход автомата A, который реализуется по формулам (6.2);

4) двоичное слово $\tilde{\pi} = \pi_1 \ldots \pi_\rho$, которое поступает на входы триггеров в запоминающей части и управляет памятью автомата.

Покажем, что сигналы управления памятью являются булевыми функциями от тех же переменных, что и выход автомата, а, значит, они могут быть реализованы полной системой ФЭ. В каждый момент времени t сигналы управления памятью должны переводить автомат из состояния $\tilde{\sigma}(t)$ в состояние $\tilde{\sigma}(t+1)$. Для этого надо изменить состояние каждого триггера

$$\sigma_i(t) \rightarrow \sigma_i(t+1),\ i = 1, \ldots, \rho.$$

Используемые в канонической схеме D-триггеры или T-триггеры обладают следующим свойством: для любой пары состояний $\sigma,\ \sigma'$ существует входной сигнал, переводящий автомат из состояния σ в состояние σ'. Обозначим этот сигнал через $\pi(\sigma,\ \sigma')$. Для D-триггера $\pi_D(\sigma,\ \sigma') = \sigma'$, так как состояние, в которое устанавливается D-триггер, равно входному сигналу. Для T-триггера $\pi_T(\sigma,\ \sigma') = \sigma \oplus \sigma'$: при $\sigma = \sigma'$ на вход надо подать 0, чтобы состояние не изменилось; при $\sigma \neq \sigma'$ – 1, чтобы триггер «перевернулся».

Итак, $\pi_i = \pi(\sigma_i(t),\ \sigma_i(t+1)),\ i = 1, \ldots, \rho$ или в векторной форме $\tilde{\pi} = \tilde{\pi}(\tilde{\sigma}(t),\ \tilde{\sigma}(t+1))$. Выразим $\tilde{\sigma}(t+1)$ из закона функционирования автомата (6.1). Тогда

$$\tilde{\pi} = \tilde{\pi}(\tilde{\sigma}(t),\ \Delta(\tilde{\xi}(t),\ \tilde{\sigma}(t)).\blacksquare$$

Канонический метод синтеза автомата. Рассмотрим этот метод на конкретном примере. Пример. На конвейере, по которому двигаются детали двух типов A и B, установлен автомат, задачей которого является такая сортировка деталей, чтобы после прохождения мимо автомата они образовывали группы ABB. Неподходящую деталь автомат сталкивает с конвейера. Требуется построить схему такого автомата, используя T-триггер и элементы «И», «ИЛИ», «НЕ».

Синтез автомата разбивается на следующие этапы.

1°. Построение абстрактного автомата $A = (X,\ Y,\ Q,\ \delta,\ \lambda)$.

Входной алфавит – $X=\{A, B\}$. Выходной алфавит – $Y=\{C, П\}$, где C – сталкивание детали, $П$ – ее пропуск. Внутренние состояния автомата отражают его память о том, какую часть группы ABB он уже сформировал: $Q=\{q, q_A, q_{AB}\}$. По мере формирования группы автомат циклически перемещается по этим состояниям, не изменяя состояния при поступлении неподходящей детали. Диаграмма переходов-выходов показана на рис. 6.7.

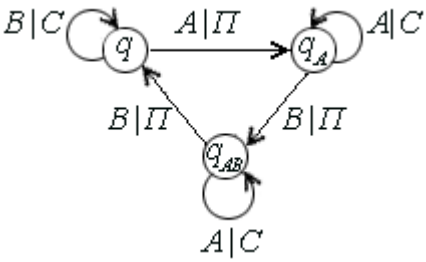


Рис. 6.7

2°. Кодирование алфавитов X, Y, Q.

Один из возможных вариантов кодирования приведен в следующих таблицах.

Вход

X	ξ
A	0
B	1

Выход

Y	η
C	0
$П$	1

Состояние

Q	$\sigma_1\sigma_2$
q	00
q_A	01
q_{AB}	11

3°. Построение канонической структуры автомата.

Каноническая структура разрабатываемого автомата показана на рис. 6.8.

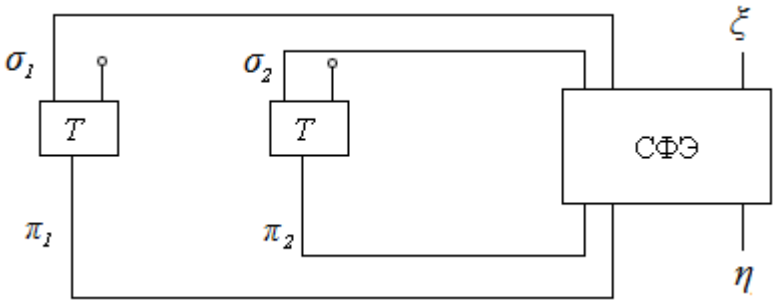


Рис. 6.8

Найдем зависимости выходов СФЭ η, π_1, π_2 от переменных ξ, σ_1, σ_2 сначала в табличном виде (табл. 6.8), по которым далее построим формулы

$$\eta = f_1(\xi,\ \sigma_1,\ \sigma_2),\ \pi_1 = f_2(\xi,\ \sigma_1,\ \sigma_2),\ \pi_2 = f_3(\xi,\ \sigma_1,\ \sigma_2).$$

Таблица 6. 8

ξ	σ_1	σ_2	η	σ_1'	σ_2'	π_1	π_2
0	0	0	1	0	1	0	1
0	0	1	0	0	1	0	0
0	1	0	*	*	*	*	*
0	1	1	0	1	1	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	1	1	0
1	1	0	*	*	*	*	*
1	1	1	1	0	0	1	1

Эти функции называются *частично определенными*, так как они не определены при $\sigma_1\sigma_2 = 10$. Для представления этих функций формулами их доопределяют таким образом, чтобы получить более простой вид формул.

4°. Представление функций выхода автомата и функций управления памятью формулами.

Используя методы минимизации булевых функций, строим по возможности экономное представление функций

$\eta = f_1(\xi,\ \sigma_1,\ \sigma_2)$, $\pi_1 = f_2(\xi,\ \sigma_1,\ \sigma_2)$, $\pi_2 = f_3(\xi,\ \sigma_1,\ \sigma_2)$ формулами в базисе $\{\vee,\ \&,\ -\}$:

$$\eta = \bar{\xi} \cdot \bar{\sigma}_2 \vee \xi \cdot \sigma_2,$$

$$\pi_1 = \xi \cdot \sigma_2,$$

$$\pi_2 = \bar{\xi} \cdot \bar{\sigma}_2 \vee \xi \cdot \sigma_1.$$

5°. Реализация СФЭ и окончательная схема автомата (рис. 6.9).

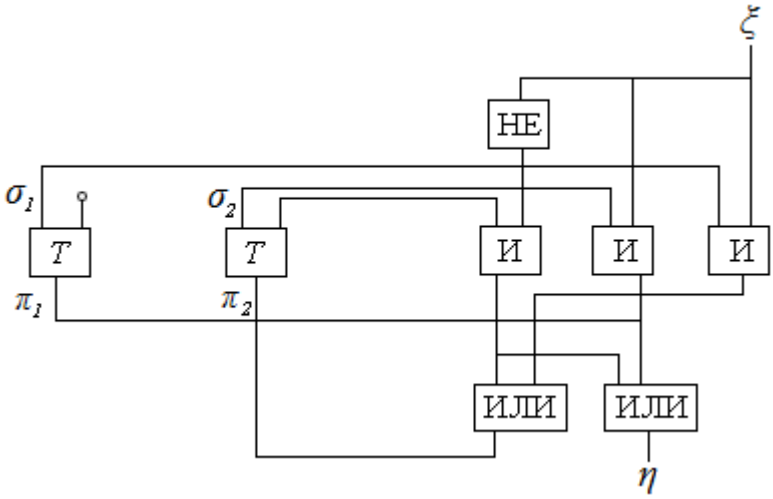


Рис. 6.9

6.3. Функциональное описание и минимизация автомата

Функциональное описание. Рассмотрим задачу анализа для конечного автомата, которая заключается в установлении функции, реализуемой данным автоматом. Далее мы используем «метод черного ящика», то есть будем рассматривать автомат как преобразователь входных слов в выходные и попытаемся установить какие именно преобразования при этом он выполняет.

Словами длины k в алфавите $X = \{x_1,\ \ldots,\ x_n\}$ называют элементы декартового произведения X^k: $\tilde{x} = x_{i_1} \ldots x_{ik}$. При $k = 0$

имеем пустое слово, которое обозначается Λ. Множество всех слов в алфавите X обозначается X^*: $X^* = \Lambda \cup X \cup X^2 \cup ... \cup X^n$. Длину слова $\tilde{x}$ обозначим $l(\tilde{x})$. Например, $l(\Lambda) = 0$, $l(\tilde{x}) = k \Leftrightarrow \tilde{x} \in X^k$.

Пусть $\tilde{x}_1 = x_{i_1} ... x_{ip}$ и $\tilde{x}_2 = x_{j_1} ... x_{jq}$ – произвольные слова из алфавита X. Приписывание слова $\tilde{x}_1$ к слову $\tilde{x}_2$ называется *конкатенацией*: $\tilde{x}_1\tilde{x}_2 = x_{i_1} ... x_{ip} x_{j_1} ... x_{jq}$. Операция конкатенации обладает следующими свойствами:

а) ассоциативность: $\tilde{x}_1(\tilde{x}_2\tilde{x}_3) = (\tilde{x}_1\tilde{x}_2)\tilde{x}_3$;

б) существование нейтрального элемента: $\Lambda\tilde{x} = \tilde{x}\Lambda = \tilde{x}$.

Очевидно, что эта операция некоммутативна.

Пусть X и Y – два алфавита, X^* и Y^* – соответствующие им множества слов. Отображение

$$\varphi: X^* \to Y^*, \tilde{x} \to \tilde{y} = \varphi(\tilde{x})$$

называется *словарным оператором*.

Приведем примеры словарных операторов для двоичных алфавитов $X = Y = \{0, 1\}$.

Пример 6.3. Оператор φ_1 сопоставляет каждому слову его первую букву:

$$\varphi_1(\alpha_1 ... \alpha_n) = \alpha_1.$$

Пример 6.4. Оператор φ_2 производит в каждом слове инверсию нулей единиц:

$$\varphi_2(\alpha_1 ... \alpha_n) = \tilde{\alpha}_1 ... \tilde{\alpha}_n.$$

Пример 6.5. Оператор φ_3 переписывает каждое слово слева направо:

$$\varphi_3(\alpha_1 ... \alpha_n) = \alpha_n ... \alpha_1.$$

Пример 6.6. Оператор φ_4 определяется следующим образом:

$$\varphi_4(\alpha_1\alpha_2 ... \alpha_n) = \beta_1\beta_2 ... \beta_n,$$

где

$$\beta_1 = \alpha_1,$$

$$\beta_2 = \alpha_1 \oplus \alpha_2,$$
$$\dots$$
$$\beta_n = \alpha_1 \oplus \alpha_2 \oplus \dots \oplus \alpha_n.$$

Рассмотрим далее словарные операторы, реализуемые автоматами. По определению автомата $A = (X, Y, Q, \delta, \lambda)$, где $\delta(x, q)$ – состояние автомата, в которое он переходит из состояния q, когда на его вход поступает символ x; $\lambda(x, q)$ символ на выходе автомата, находящегося в состоянии q, в момент, когда на его вход поступает символ x. Функции $\delta(x, q)$ и $\lambda(x, q)$ определены на множестве $X \times Q$. Расширим область определения до $X^* \times Q$, полагая, что на вход поступает не символ x, а слово $\tilde{x}$.

Формальное определение функций $\delta(\tilde{x}, q)$ и $\lambda(\tilde{x}, q)$ можно дать индуктивно по длине слова.

Базис индукции. Полагаем $\delta(\Lambda, q) = q$, $\lambda(\Lambda, q) = \Lambda$.

Индуктивный переход. Предположим, что функции $\delta(\tilde{x}, q)$ и $\lambda(\tilde{x}, q)$ определены для всех слов $\tilde{x}$, длина которых не превосходит $n-1$. Рассмотрим произвольное слово $x\tilde{x}$ длины n с первой буквой x, то есть $l(\tilde{x}) = n-1$. Определим $\delta(x\tilde{x}, q)$ и $\lambda(x\tilde{x}, q)$:

$$\delta(x\tilde{x}, q) = \delta(\tilde{x}, \delta(x, q)), \qquad (6.3)$$
$$\lambda(x\tilde{x}, q) = \lambda(x, q) \cdot \lambda(\tilde{x}, \delta(x, q)). \qquad (6.4)$$

Из формул (6.3) и (6.4) следует, что функции $\delta(\tilde{x}, q)$ и $\lambda(\tilde{x}, q)$ полностью определяются функциями $\delta(x, q)$ и $\lambda(x, q)$, задающими закон функционирования автомата, и являются суперпозициями этих функций. Например, для слова $\tilde{x} = abc$ из формулы (6.3):

$$\delta(\tilde{x}, q) = \delta(abc, q) = \delta(bc, \delta(a, q)) = \delta(c, \delta(b, \delta(a, q))).$$

Из формул (6.3) и (6.4) $\forall \tilde{x}_1, \tilde{x}_2$:

$$\delta(\tilde{x}_1\tilde{x}_2, q) = \delta(\tilde{x}_2, \delta(\tilde{x}_1, q)),$$
$$\lambda(\tilde{x}_1\tilde{x}_2, q) = \lambda(\tilde{x}_1, q) \cdot \lambda(\tilde{x}_2, \delta(\tilde{x}_1, q)).$$

Будем считать, что в момент $t=1$ автомат находится в состоянии $q_1 \in Q$, которое будем называть *начальным*.

Словарным оператором, реализуемым автоматом A, называется отображение

$$\varphi_A : X^* \to Y^*,\ \tilde{x} \to \tilde{y} = \varphi_A(\tilde{x}),$$

определяемое следующим образом:

$$\varphi_A(\tilde{x}) = \lambda(\tilde{x},\ q_1). \qquad (6.5)$$

Поставим вопрос: каждое ли преобразование слов можно реализовать на автомате? Формально нужно проверить выполнение условия: $\forall \varphi \exists A: \varphi_A = \varphi$.

Оказывается, что для выполнения этого условия необходимо, чтобы словарный оператор φ обладал следующими тремя свойствами.

1°. $\forall \tilde{x} \in X:\ l(\varphi(\tilde{x})) = l(\tilde{x})$. Следует из (6.4) и (6.5).

2°. $\forall \tilde{x}_0,\ \tilde{x} \in X:\ \varphi(\tilde{x}_0\tilde{x}) = \tilde{y}_0\tilde{y}$, где $\tilde{y}_0 = \varphi(\tilde{x}_0)$. Словарный оператор φ должен отображать слова с общим началом в слова с общим началом.

Словарный оператор φ, обладающий свойствами 1° и 2°, называется *детерминированным*.

3°. Пусть φ – детерминированный оператор, а $\tilde{x}_0$ – некоторое слово. *Остаточным оператором* оператора φ, порожденным словом $\tilde{x}_0$, называется словарный оператор $\varphi_{\tilde{x}_0} : X^* \to Y^*$, действующий по правилу:

$$\varphi_{\tilde{x}_0} = \tilde{y}, \text{ если } \varphi(\tilde{x}_0\tilde{x}) = \tilde{y}_0\tilde{y}. \qquad (6.6)$$

Корректность этого определения вытекает из свойства 2°. Правило (6.6) можно сформулировать так: чтобы найти образ слова $\tilde{x}$ при отображении $\varphi_{\tilde{x}_0}$, припишем к нему слева слово $\tilde{x}_0$, найдем образ полученного слова при отображении φ и удалим в полученном слове $l(\tilde{x}_0)$ букв. Из этого правила получим соотношение

$$\varphi_{\tilde{x}_1}(\tilde{x}_2\tilde{x}) = \varphi_{\tilde{x}_1}(\tilde{x}_2)\varphi_{\tilde{x}_1\tilde{x}_2}(\tilde{x}). \qquad (6.7)$$

Действительно, из равенства $\varphi(\tilde{x}_1\tilde{x}_2\tilde{x}) = \tilde{y}_1\tilde{y}_2\tilde{y}$ по определению (6.6) следует

$$\varphi_{\tilde{x}_1}(\tilde{x}_2\tilde{x}) = \tilde{y}_2\tilde{y} \text{ и } \varphi_{\tilde{x}_1\tilde{x}_2}(\tilde{x}) = \tilde{y}.$$

При $x = \Lambda$ имеем $\varphi_{\tilde{x}_1}(\tilde{x}_2) = \tilde{y}_2$. Отсюда получаем (6.7).

Рассмотрим остаточный оператор для словарного оператора, реализуемого автоматом. Для множества слов, имеющих общее начало $\{\tilde{x}_0\tilde{x} \mid \tilde{x} \in X^*\}$, имеем

$$\varphi_A(\tilde{x}_0\tilde{x}) = \lambda(\tilde{x}_0\tilde{x},\ q_1) = \lambda(\tilde{x}_0,\ q_1)\cdot\lambda(\tilde{x},\ \delta(\tilde{x}_0,\ q_1)).$$

Отсюда

$$(\varphi_A)_{\tilde{x}_0}(\tilde{x}) = \lambda(\tilde{x},\ \delta(\tilde{x}_0,\ q_1)).$$

Так как $\delta(\tilde{x}_0,\ q_1)$ – одно из состояний автомата q_i, то

$$(\varphi_A)_{\tilde{x}_0}(\tilde{x}) = \lambda(\tilde{x},\ q_i).$$

Таким образом, $\forall\,(\varphi_A)_{\tilde{x}_0}\,\exists q_i : (\varphi_A)_{\tilde{x}_0}(\tilde{x})$ – выход автомата, установленного в состояние q_i, на вход которого подано слово $\tilde{x}$. Так число состояний автомата конечно, то получаем свойство 3°, согласно которому *словарный оператор φ должен иметь конечное число остаточных операторов*.

Словарный оператор, обладающий свойствами 1°–3°, называется *ограниченно-детерминированным.*

Мы доказали следующую теорему.

Теорема 6.2. *Словарный оператор, реализуемый конечным автоматом, является ограниченно-детерминированным.*

Справедлива и обратная теорема.

Теорема 6.3. *Для каждого ограниченно-детерминированного словарного оператора существует реализующий его конечный автомат.*

Доказательство. Пусть $\varphi\colon X^* \to Y^*$ – ограниченно-детерминированный словарный оператор. Обозначим через $r(\varphi)$ число различных остаточных операторов словарного оператора φ. Если $r(\varphi) = r$, то $\{\varphi_i\ i = 1, ..., r\}$ – множество всех остаточных операторов словарного оператора φ $(\varphi_1 = \varphi_\Lambda = \varphi)$. По-

скольку каждый остаточный оператор порождается некоторым словом $\tilde{x}_0 \in X^*$, разобьем множество всех слов на r классов:

$$X^* = \bigcup_{i=1}^{r} C_i , \tag{6.8}$$

где $C_i = \{\tilde{x} \mid \varphi_{\tilde{x}} = \varphi_i\}$. Обозначим класс, в который входит слово $\tilde{x}$ через $C(\tilde{x})$. Имеем

$$C(\tilde{x}_1) = C(\tilde{x}_2) \Leftrightarrow \varphi_{\tilde{x}_1} = \varphi_{\tilde{x}_2} .$$

Разбиение (6.8) обладает следующим свойством

$$\forall x \in X \mid C(\tilde{x}_1) = C(\tilde{x}_2) \Rightarrow C(\tilde{x}_1 x) = C(\tilde{x}_2 x) . \tag{6.9}$$

Построим автомат $A_0 = (X_0,\ Y_0,\ Q_0,\ \delta_0,\ \lambda_0)$, реализующий оператор φ.

Так $\varphi\colon X^* \to Y^*$, то $X_0 = X^*$, $Y_0 = Y^*$. В качестве состояний автомата возьмем классы разбиения (6.8):

$$Q_0 = \{C(\tilde{x}) \mid \tilde{x} \in X^*\} = \{C_1,\ ...,\ C_r\} . \tag{6.10}$$

Функция переходов:

$$\delta_0(x,\ C(\tilde{x})) = C(\tilde{x}x) . \tag{6.11}$$

Соотношение (6.11) можно по индукции распространить на слова:

$$\delta_0(\tilde{x}_1,\ C(\tilde{x})) = C(\tilde{x}\tilde{x}_1) . \tag{6.12}$$

Функция выходов:

$$\lambda_0(x,\ C(\tilde{x})) = \varphi_{\tilde{x}}(x) . \tag{6.13}$$

Покажем индукцией по длине слова, что $\varphi_{A_0} = \varphi$, то есть

$$\forall \tilde{x} \in X^* \mid \varphi_{A_0}(\tilde{x}) = \varphi(x) . \tag{6.14}$$

Если $l(\tilde{x}) = 1$, то $\varphi_{A_0}(x) = \lambda_0(x,\ C_1) = \varphi_\Lambda(x) = \varphi(x)$. Пусть (6.14) справедливо для всех слов длины $n-1$. Рассмотрим произвольное слово длины n с последней буквой x:

$$\varphi_{A_0}(\tilde{x}x) = \lambda_0(\tilde{x}x,\ C_1) = \lambda_0(\tilde{x},\ C_1) \cdot \lambda_0(x,\ \delta(\tilde{x},\ C_1)) .$$

По предположению индукции $\lambda_0(\tilde{x},\ C_1) = \varphi_{A_0}(\tilde{x}) = \varphi(\tilde{x})$. На основании (6.12) имеем $\delta(\tilde{x},\ C_1) = C(\tilde{x})$. Тогда

$$\varphi_{A_0}(\tilde{x}x) = \varphi(\tilde{x}) \cdot \lambda_0(x,\ C(\tilde{x})) = \varphi(\tilde{x}) \cdot \varphi_{\tilde{x}}(x) = \varphi(\tilde{x}x) .\blacksquare$$

Минимизация автомата. Каждый остаточный оператор реализуется в некотором состоянии автомата. Отсюда следует, что у любого автомата A, реализующего ограниченно-детерминированный оператор φ, число состояний $\rho(A)$ не меньше числа различных остаточных операторов $r(\varphi)$ оператора φ:

$$\varphi_A = \varphi \Rightarrow \rho(A) \geq r(\varphi). \qquad (6.15)$$

Автомат с наименьшим числом состояний, реализующий ограниченно-детерминированный оператор φ, называется *минимальным автоматом оператора* φ.

Построенный при доказательстве последней теоремы автомат A_0 – минимальный.

Теорема 6.4. *Минимальный автомат единственен с точностью до обозначения состояний.*

Из теоремы следует признак минимальности автомата: *автомат будет минимальным, если для любых двух его состояний* q *и* q' *реализуемые в этих состояниях остаточные операторы различны.*

Задача минимизации автомата: для данного автомата A построить минимальный автомат, реализующий ограниченно-детерминированный оператор φ.

Алгоритм решения этой задачи основан на разбиении множества Q на непересекающиеся классы:

$$Q = \bigcup_{i=1}^{r} C_i .$$

На очередном шаге алгоритма происходит измельчение предыдущего разбиения до тех пор, пока это возможно. После завершения алгоритма классы разбиения будут состояниями минимального автомата.

1-й шаг. Состояния q и q' отнесем к одному классу, если

$$\forall x \in X: \ \lambda(x, q) = \lambda(x, q').$$

Пусть r_1 – число классов в разбиении $R^{(1)}$.

$i+1$-й шаг. Пусть на i-ом шаге получено разбиение $R^{(i)}$. Состояния q и q' отнесем к одному классу разбиения $R^{(i+1)}$, если выполнены два условия:

1) q и q' входят в один класс разбиения $R^{(i)}$;

2) для каждого символа x состояния $\delta(x, q)$ и $\delta(x, q')$ входят в один класс разбиения $R^{(i)}$.

Обозначим через $C(q)$ класс, в который входит состояние q. Тогда условий 1) и 2):

1) $C(q) = C(q')$;

2) $\forall x(C(\delta(x, q)) = C(\delta(x, q'))$.

Алгоритм заканчивает работу, если на некотором шаге k не произойдет дальнейшего измельчения разбиения: $R^{(k+1)} = R^{(k)}$. Тот факт, что алгоритм закончил работу можно выразить следующим образом:

$$C(q) = C(q') \Rightarrow \forall x(C(\delta(x, q)) = C(\delta(x, q')).$$

Строим автомат $A_0 = (X_0, Y_0, Q_0, \delta_0, \lambda_0)$:

$X_0 = X$, $Y_0 = Y$, $Q_0 = \{C_1, ..., C_r\}$, $\delta_0(x, C(q)) = C(\delta(x, q))$, $\lambda_0(x, C(q)) = \lambda(x, q)$.

Автомат A_0 реализует тот же словарный оператор, что и автомат A, и является минимальным.

6.4. Регулярные языки. Автоматы Мили и Мура

Регулярные языки. До начала XX века наука о языках – лингвистика – сводилась в основном к изучению естественных языков (русского, английского, латинского и т. д.), их классификации, выяснению сходств и различий между ними.

Возникновение и развитие метаматематики, изучающей язык математики, привели в 30-х годах . XX века к существенно более широкому представлению о языке, при котором под языком понимается всякое средство общения, состоящее из знаковой системы, т. е. множества допустимых последовательностей знаков, множества смыслов этой системы и соответствия между

последовательностями знаков и смыслами, делающего «осмысленными» допустимые последовательности знаков.

Наука об осмысленных знаковых системах называется *семиотикой*. Наиболее продвинутыми являются исследования знаковых систем, в которых знаками являются символы алфавитов, а последовательностями знаков – тексты. К таким знаковым системам относятся естественные языки, языки науки, а также языки программирования. Именно интерес к языкам программирования, совпавший с новыми подходами в структурной лингвистике и необходимостью решать задачу машинного перевода естественных языков, привел в 50-х гг. к возникновению новой науки – математической лингвистики, которая рассматривает языки как произвольные множества осмысленных текстов.

Правила, определяющие множества текстов, образуют *синтаксис* языка; описание множества смыслов и соответствия между текстами и смыслами – *семантику* языка. Наибольших успехов математическая лингвистика достигла в изучении синтаксиса, где в последние время сложился специальный математический аппарат – *теория формальных грамматик*.

С точки зрения синтаксиса язык понимается не как средство общения, а как множество формальных объектов – последовательностей символов алфавита, которые в теории алгоритмов и формальных систем называют словами. Язык, понимаемый как множество слов, будем называть *формальным языком*.

Пусть задан конечный алфавит $X = \{x_1, \ldots, x_n\}$ и тем самым множество всех конечных слов в этом алфавите:

$$X^* = \Lambda \cup X \cup X^2 \cup \ldots \cup X^n.$$

Формальный язык L в алфавите X – это произвольное подмножество $L \subseteq X^*$.

Набор правил образования слов формального языка называют его *грамматикой*. В зависимости от сложности этих правил формальные языки делятся на ряд классов. Далее мы рассмотрим один из наиболее простых классов языков – регулярные языки – и установим их связь с конечными автоматами.

Рассмотрим совокупность языков с одним и тем же алфавитом и введем операции над этими языками.

1°. *Объединение*. Это теоретико-множественная операция, которая, в отличие от пересечения и дополнения, имеет естественную синтаксическую интерпретацию:

$$L = L_1 \bigcup L_2 .$$

2°. *Конкатенация* – это операция, связанная с подстановкой языка в язык:

$$L = L_1 L_2 = \{\tilde{x} = \tilde{x}_1 \tilde{x}_2 \mid \tilde{x}_1 \in L_1,\ \tilde{x}_2 \in L_2\} .$$

3°. *Итерация* L^* языка L определяется равенством

$$L^* = \{\Lambda\} \bigcup L \bigcup L^2 \bigcup ... \bigcup L^n = \{\Lambda\} \bigcup \bigcup_{i=1}^{n} L^i ,$$

где $\{\Lambda\}$ – язык, состоящий из пустого слова, который не надо смешивать с пустым языком $\varnothing$, не содержащим ни одного слова; $L^2 = LL$, $L^3 = L^2 L$, …

Языки $\{\Lambda\}$, $\{x_1\}$, ..., $\{x_n\}$, состоящие из пустого или однобуквенного слова, называются *элементарными*.

Язык называется *регулярным*, если его можно построить с помощью конечного числа операций объединения, конкатенации и итерации.

Автоматы Мили и Мура. Конечный автомат называется *автоматом Мура*, если его функция выходов зависит только от состояния:

$$\lambda(x,\ q) = \mu(q) .$$

Общая модель конечного автомата, которая рассматривалась ранее, называется *автоматом Милли.*

Несмотря на то, что автомат Милли – частный случай автомата Мура, возможности этих двух автоматов совпадают.

Теорема 6.5. *Для любого автомата Милли существует эквивалентный ему автомат Мура.*

Доказательство. Пусть автомат Мили $A = (X,\ Y, Q,\ \delta,\ \lambda)$, имеющий n входных символов и s состояний, реализует ограниченно-детерминированный оператор φ. Построим эквивалентный автомат Мура $M = (X,\ Y,\ Q_M,\ \delta_M,\ \mu)$. В число состояний автомата M включим все состояния автомата A, обозначив их q_{01}, ..., q_{0s}. Кроме того, для каждой пары

$(x_i,\ q_j) \in X \times Q$ включим в Q_M состояние q_{ij}. Автомат M будет находиться в состоянии q_{ij}, когда на автомат A в состоянии q_j поступает символ x_i. Тогда выход автомата M можно определить следующим образом:

$$\begin{aligned} &\mu(q_{ij}) = \lambda(x_i, q_j),\ 1 \le i \le n,\ 1 \le j \le s, \\ &\mu(q_{0j}) \text{ определяем произвольно } 1 \le j \le s. \end{aligned} \tag{6.16}$$

Таким образом, автомат M имеет $s + ns$ состояний. Определим функцию его переходов:

$$\begin{aligned} &\delta_M(x_i,\ q_{0j}) = q_{ij}, \\ &\delta_M(x_i,\ q_{jk}) = q_{il},\ \text{если } \delta(x_i,\ q_k) = q_l, \\ &1 \le i \le n,\ 1 \le j \le n,\ 1 \le k \le s,\ 1 \le l \le s. \end{aligned} \tag{6.17}$$

Связь между δ_M и δ:

$$\forall \tilde{x} x_i :\ \delta(\tilde{x},\ q_k) = q_l \Rightarrow \delta_M(\tilde{x} x_i,\ q_{0k}) = q_{il}.$$

Для того, чтобы убедиться, что автоматы A и M реализуют один и тот же словарный оператор, то есть, что $\varphi_A = \varphi_M$, применим индукцию по длине слов. Для однобуквенных слов

$$\varphi_A(x_i) = \lambda(x_i,\ q_1).$$

Автомат M с начальным состоянием q_{01} под действием входа x_i перейдет в состояние $\delta_M(x_i,\ q_{01})$, равное q_{i1} (см. (6.17), а в следующем такте будет иметь выход $\mu(q_{i1})$, равный $\lambda(x_i,\ q_1)$ (см. (6.2)). Отсюда $\varphi_A(x_i) = \varphi_M(x_i)$.

Пусть φ_A и φ_M совпадают на всех словах, длина которых не превосходит $l-1$. Для слова $\tilde{x} x_i$ дины имеем

$$\varphi_A(\tilde{x} x_i) = \lambda(\tilde{x} x_i,\ q_1) = \lambda(\tilde{x},\ q_1)\lambda(x_i,\ \delta(\tilde{x},\ q_1)). \tag{6.18}$$

По предположению индукции

$$\lambda(\tilde{x},\ q_1) = \varphi_A(\tilde{x}) = \varphi_M(\tilde{x}). \tag{6.19}$$

Обозначим $\delta(\tilde{x},\ q_1) = q_m$. Тогда

$$\lambda(x_i,\ \delta(\tilde{x},\ q_1)) = \lambda(x_i,\ q_m) = \mu(q_{im}). \tag{6.20}$$

На основе связи между δ_M и δ:

$$\delta_M(\tilde{x}x_i,\ q_{01}) = q_{im}. \qquad (6.21)$$

Подставляя (6.19) – (6.21) в (6.18), получим

$$\varphi_A(\tilde{x}x_i) = \lambda(\tilde{x},\ q_1)\lambda(x_i,\ \delta(\tilde{x},\ q_1)) = \varphi_M(\tilde{x})\mu(q_{im}) =$$

$$= \varphi_M(\tilde{x})\varphi_M(\delta_M(\tilde{x}x_i,\ q_{01})) = \varphi_M(\tilde{x}x_i).$$

Распознавание языков автоматами. Рассмотрим автомат Мура с двумя выходными символами 0 и 1. Такой автомат будет для одних входных слов выдавать 1, для других – 0. Будем считать, что в первом случае автомат «распознал» слово, а во втором – нет. Тем самым определяется некоторый язык, состоящий из слов, распознаваемых автоматом.

Разобьем состояния автомата Мура на два класса: класс F – выход равен 1, класс $Q \setminus F$ – выход равен 0. Это позволяет не рассматривать функцию выходов и определить *автомат-распознаватель* как систему $A = (X,\ Q,\ \delta,\ q_1, F)$.

С каждым таким автоматом свяжем распознаваемый им язык $L(A) = \{\tilde{x} \mid \delta(\tilde{x},\ q_1) \in F\}$, то есть язык $L(A)$ состоит из всех слов, которые переводят автомат A из начального состояния q_1 в одно из заключительных.

Пример 6.7. $A = (X,\ Q,\ \delta,\ q_1\ F)$, где $X = \{0,\ 1\}$, $Q = \{1,\ 2\}$, $q_1 = 1$, $F = \{1\}$, а δ задается таблицей

Вход	Состояние	
	1	2
0	1	2
1	2	1

Слова, переводящие автомат в состояние 1 – это слова с четным количеством единиц. Поэтому язык $L(A) = (0 \cup 10^*1)^* = (0 \cup 101 \cup 1011 \cup 10111 \cup \ldots)^*$.

Приведем без доказательства следующее утверждение, называемое *теоремой анализа*.

Теорема 6.6. *Язык, распознаваемый автоматом, является регулярным.*

7. Элементы теории алгоритмов

Теория алгоритмов – это раздел математики, изучающий общие свойства алгоритмов. Различают *качественную* и *метрическую* теорию алгоритмов.

Основной проблемой качественной теории алгоритмов является проблема построения алгоритма, обладающими заданными свойствами. Такую проблему называют алгоритмической.

Метрическая теория алгоритмов исследует алгоритмы с точки зрения их сложности. Этот раздел теории алгоритмов известен также как *алгоритмическая теория сложности*.

Приложения теории алгоритмов имеются практически во всех разделах математики, а также во многих прикладных дисциплинах.

7.1. Понятие алгоритма

С алгоритмами, т. е. эффективными процедурами, однозначно приводящими к результату, математика имела дело с давних времен. При этом понятие алгоритма оставалось описательным и определялось примерно так: *алгоритм* – это четкая система правил оперирования над объектами, приводящая после некоторого числа шагов к достижению поставленной цели.

Слово «алгоритм» произошло от имени узбекского математика и астронома Мухаммеда бен Мусы аль-Хорезми (IX в.), который впервые разработал правила выполнения арифметических операций в позиционной системе счисления. Затем понятие алгоритма стало использоваться в более широком смысле и не только в математике. Приведенное выше определение алгоритма не является строгим с математической точки зрения и называется *интуитивным*.

Рассмотрим основные требования к алгоритмам, которые установились в процессе развития математики.

1°. Дискретность. Любой алгоритм состоит из отдельных шагов, на каждом из которых происходят определенные преобразования некоторой системы величин. В начале алгорит-

ма эту систему образуют исходные данные, в конце – результаты работы алгоритма.

2°. Детерминированность. Система величин, получаемая на каждом шаге алгоритма, однозначно определяется величинами, полученными на предыдущих шагах.

3°. Элементарность шагов. Преобразования системы величин на каждом шаге алгоритма должны быть простыми и число шагов должно быть конечным.

4°. Результативность. Заключается в остановке алгоритма после конечного числа шагов с указанием того, что считать результатом.

5°. Массовость. Заключается в выборе для работы алгоритма исходных данных из потенциально бесконечного множества.

Со временем в математике накопился ряд проблем, для которых не удалось найти алгоритма их решения, и имелись серьезные основания предполагать, что такого алгоритма нет. В связи с этим возникла необходимость формального определения алгоритма как некоторого математического объекта таким образом, чтобы для одних задач такой объект можно было построить, а для других доказать, что такого объекта не существует.

Такие формальные определения алгоритма были сделаны в 30-х годах XX-го века в работах сразу нескольких математиков, которые оказались различными по форме, но эквивалентными по содержанию.

На основе анализа алгоритмов из самых различных областей человеческой деятельности можно постулировать следующее утверждение: *любой алгоритм представляет собой преобразование слов в некотором алфавите в слова в том же алфавите, то есть словарный оператор.*

Таким образом, для определения алгоритма необходимо из множества словарных операторов выделить те, которые можно рассматривать как алгоритмы. Это можно сделать разными способами, из которых основными являются следующие:

1°. Рекурсивные функции.

2°. Машины Тьюринга.

3°. Нормальные алгорифмы А.А. Маркова.

7.2. Машины Тьюринга

Определение и примеры машин Тьюринга. Одно из формальных определений алгоритма связано с использованием математической модели вычислительного устройства, которое носит название *машины Тьюринга*.

Каждая машина Тьюринга имеет *ленту, управляющую головку* (читающую и записывающую) и работает с некоторым конечным *алфавитом* $A=\{a_0, a_1, ..., a_n\}$. Лента бесконечна в обе стороны и разбита на клетки. Машина работает в дискретные моменты времени. Среди букв алфавита A имеется «пустой» символ $a_0 = \Lambda$. В каждый момент времени в каждой клетке записана одна из букв алфавита A.

Управляющая головка представляет собой конечный автомат с множеством внутренних состояний $Q=\{q_0, q_1, ..., q_r\}$. В каждый момент времени она находится в одном из этих состояний и обозревает одну из клеток ленты. В зависимости от своего состояния и буквы на ленте головка записывает новую букву (в частности, ту же самую букву), переходит к следующему моменту времени в новое состояние(в частности, то же самое) и сдвигается по ленте – либо на одну клетку влево (L), либо на одну клетку вправо (R), либо остается на месте (C). Среди внутренних состояний выделено одно, скажем q_0, которое называется *заключительным*. Если после некоторого очередного сдвига головка попадает в заключительное состояние, то машина прекращает свою работу (останавливается).

Чтобы полностью определить работу машины, достаточно указать для начального момента времени ее *конфигурацию*, в которую входят:

1) распределение букв по клеткам ленты;
2) клетка, обозреваемая головкой;
3) состояние головки.

Обычно рассматриваются конфигурации, в которых на ленте записано конечное число непустых символов. В качестве начального состояния будем считать q_1. Далее будем предполагать, что в начальной конфигурации головка обозревает крайнюю правую клетку ленты с непустым символом.

Если машина, работая в соответствие с указанными правилами, в некоторый момент времени придет в заключительное состояние, то она считается *применимой* к данной конфигурации (или к начальному слову, записанному на ленте). Результатом работы машины при этом считается слово, которое окажется записанным на ленте в заключительной конфигурации.

Если же при работе машины ни в какой момент времени ее головка не окажется в заключительном состоянии, то машина считается *неприменимой* к начальной конфигурации (и к начальному слову). Результат ее работы в этом случае не определен.

Поскольку каждая машина Тьюринга имеет конечный алфавит и конечное число состояний, то она полностью определяется *конечным числом команд* вида

$$qa \to q'a'S,$$

где q – состояние головки; a – буква в обозреваемой головкой клетке; q' – состояние головки в следующий момент; a' – буква, записываемая вместо a в обозреваемую клетку; S – сдвиг головки в следующий момент (L, R или C).

Пример 7.1. Машина “+1” имеет следующую систему команд:

$$q_1 0 \to q_0 1C$$
$$q_1 1 \to q_1 0L$$
$$q_1 \Lambda \to q_0 1C$$

Нетрудно убедиться, что если на ленте написать двоичное разложение числа n ($n = 0, 1, 2, \ldots$), установить головку в состояние q_1 на младший разряд и последовательно применять систему команд, то после остановки машины на ленте будет написано число $n+1$.

Пример 7.2. Машина "$\rightarrow$" задается системой команд

$$q_1 0 \to q_1 0R$$
$$q_1 1 \to q_1 1R$$
$$q_1 \Lambda \to q_0 \Lambda L$$

Эта машина, не изменяя написанного на ленте слова, устанавливает головку против крайнего правого символа слова.

Меняя R и L, получим машину "$\leftarrow$", выходящую на крайний левый символ слова.

Пример 7.3. Машина "–1" задается системой команд

$$q_1 0 \to q_1 1L$$
$$q_1 1 \to q_0 0C$$
$$q_1 \Lambda \to q_2 \Lambda R$$
$$q_2 1 \to q_2 \Lambda R$$
$$q_2 \Lambda \to q_{00} \Lambda C$$

Если в начальной конфигурации записано двоичное разложение положительного числа n, то машина преобразует его в двоичное разложение числа $n-1$. Если же в начальной конфигурации записано двоичное разложение нуля, то машина останавливается в заключительном состоянии q_{00}.

Композиция машин Тьюринга. Для построения сложных машин Тьюринга определяют понятие композиции машин Тьюринга.

Композицией машин Тьюринга M_1 и M_2 называется машина Тьюринга M, которая первоначально функционирует как машина M_1, заключительное состояние которой q_0 изменяют на начальное состояние машины M_2.

Композицию машин Тьюринга обозначают: $M = M_1 \circ M_2$. Из определения композиции машин Тьюринга следует, что $M_1 \circ M_2 \neq M_2 \circ M_1$.

1°. Машина последовательной обработки. Пусть M_1 и M_2 – машины Тьюринга в одном алфавите. По-

строим третью машину Тьюринга, работа которой будет равносильна последовательной работе машин M_1 и M_2.

Пусть машина M_1 имеет множество состояний $\{q_{10}, q_{11}, ..., q_{1n}\}$, а M_2 – $\{q_{20}, q_{21}, ..., q_{2m}\}$. Назначим заключительное состояние q_{10} машины M_1 начальным состоянием q_{21} машины M_2. Определим машину M более точно. Пусть она имеет множество состояний $\{q_0, q_1, ..., q_{n+m}\}$, а ее команды получаются из команд машин M_1 и M_2 следующим образом:

– в командах машины M_1 всюду заменим q_{10} на q_{n+1};

– в командах машины M_2 всюду заменим q_i $(i \neq 0)$ на q_{n+i}.

Такую композицию машин M_1 и M_2 называют *машиной последовательной обработки* и обозначают $M_2(M_1)$.

Пример 7.4. Композиция машин “+1” и "$\rightarrow$" дает машину, которая после прибавления 1 возвращает головку на правый символ результата. Программа новой машины:

$$q_1 0 \rightarrow q_2 1C$$
$$q_1 1 \rightarrow q_1 0L$$
$$q_1 \Lambda \rightarrow q_2 1C$$
$$q_2 0 \rightarrow q_2 0R$$
$$q_2 1 \rightarrow q_2 1R$$
$$q_2 \Lambda \rightarrow q_{20} \Lambda L$$

2°. Машина условного перехода. Пусть M, M_1 и M_2 – машины Тьюринга в одном алфавите. Машина M имеет множество внутренних состояний $\{q_{0(1)}, q_{0(2)}, q_1, ..., q_n\}$, среди которых два заключительных. Машина M_1 – $\{q_{10}, q_{11}, ..., q_{1m}\}$, машина M_2 – $\{q_{20}, q_{21}, ..., q_{2p}\}$.

Построим машину, которая работает следующим образом: сначала записанное на ленту слово обрабатывает машина M, которая в зависимости от процесса обработки может остано-

виться либо в состоянии $q_{0(1)}$, либо в состоянии $q_{0(2)}$. В первом случае образовавшееся на ленте слово обрабатывается машиной M_1, во втором – M_2.

Для построения такой машины надо выполнить действия:

1) переобозначить состояния машин M_1 и M_2:

$$q_{10} \to q_{n+1}, \ldots, q_{1m} \to q_{n+m};$$

$$q_{20} \to q_{n+m+1}, \ldots, q_{2p} \to q_{m+n+p};$$

2) заменить в программе машины M $q_{0(1)}$ на q_{n+1}, $q_{0(2)}$ на q_{n+m+1} и добавить к измененному таким образом тексту программы машины M программы машин M_1 и M_2 с переобозначенными состояниями.

Построенная композиция машин M, M_1 и M_2 называется *машиной условного перехода*. Если в качестве одной из машин M_1 или M_2 взять машину M, то получим *циклическую* структуру.

7.4. Вычислимые функции и операции над ними

Определение вычислимой функции. Пусть функция $f(x_1, ..., x_n)$ определена на подмножестве E_f множества всех наборов $(\alpha_1, ..., \alpha_n)$ чисел из расширенного натурального ряда E_0 и принимающая значения также из E_0. Такого рода функции называются *частичными функциями счетнозначной логики*. Обозначим множество таких функций через $P_0^ч$.

Функция $f(x_1, ..., x_n) \in P_0^ч$ называется *вычислимой*, если существует машина Тьюринга M такая, что:

1) при $(\alpha_1, ..., \alpha_n) \in E_f$ машина M, будучи применена к основному коду $(\alpha_1, ..., \alpha_n)$ и находясь в начальном состоянии над его левой единицей, останавливается и в заключительном состоянии на ленте выдает код для $f(\alpha_1, ..., \alpha_n)$;

2) при $(\alpha_1, ..., \alpha_n) \notin E_f$ машина M, будучи применена к основному коду $(\alpha_1, ..., \alpha_n)$ и находясь в начальном состоянии над его левой единицей, либо не останавливается, либо останавливается, но при этом запись на ленте отлична от кода для $f(\alpha_1, ..., \alpha_n)$.

Обозначим через $P_{выч}$ класс всех вычислимых функций. Очевидно, $P_{выч} \subseteq P_0^ч$.

Машина Тьюринга M реализует (вычисляет) функцию $f(x_1, ..., x_n) \in P_{выч}$ *правильным образом*, если:

1) при $(\alpha_1, ..., \alpha_n) \in E_f$ машина M, будучи применена к основному коду $(\alpha_1, ..., \alpha_n)$ и находясь в начальном состоянии над его левой единицей, останавливается и в заключительном состоянии на ленте выдает код для $f(\alpha_1, ..., \alpha_n)$; при этом останов происходит над левой единицей кода для $f(\alpha_1, ..., \alpha_n)$;

2) при $(\alpha_1, ..., \alpha_n) \notin E_f$ машина M, будучи применена к основному коду $(\alpha_1, ..., \alpha_n)$ и находясь в начальном состоянии над его левой единицей, останавливается.

Справедливо следующее утверждение.

Если $f(x_1, ..., x_n)$ – вычислимая функция, то существует машина Тьюринга, которая вычисляет ее правильным образом.

Операции суперпозиции, примитивной рекурсии и минимизации. На множестве $P_0^ч$ определим три операции: C (суперпозиция), Pr (примитивная рекурсия) и μ (минимизация).

Операция *суперпозиции* вводится так же, как и для булевой алгебры.

Операция примитивной рекурсии определяется следующим образом.

Пусть $\varphi(x_1, ..., x_n)$ и $\psi(x_1, ..., x_n, x_{n+1}, x_{n+2})$ – произвольные функции из $P_0^ч$. Построим функцию $f(x_1, ..., x_n, x_{n+1})$, используя «схему» примитивной рекурсии:

$$f(x_1, \ldots, x_n, 0) = \varphi(x_1, \ldots, x_n),$$

$$f(x_1, \ldots, x_n, y+1) = \psi(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y)).$$

Пусть $(\alpha_1, \ldots, \alpha_{n+1})$ – произвольный набор чисел из E_0. Полагаем

$$f(\alpha_1, \ldots, \alpha_n, 0) = \varphi(\alpha_1, \ldots, \alpha_n).$$

Если φ на этом наборе определена, то полагаем

$$f(\alpha_1, \ldots, \alpha_n, 1) = \psi(\alpha_1, \ldots, \alpha_n, 0, f(\alpha_1, \ldots, \alpha_n, 0)).$$

Через конечное число шагов мы либо определим $f(\alpha_1, \ldots, \alpha_{n+1})$, либо установим, что на этом наборе f не определена. Говорят, что функция f получена из функций φ и ψ при помощи операции примитивной рекурсии.

Операция минимизации определяется следующим образом. Пусть $\varphi(x_1, \ldots, x_n)$ – произвольная функция из $P_0^ч$. Построим функцию $f(x_1, \ldots, x_n)$ через оператор минимизации

$$f(x_1, \ldots, x_n) = \mu_y(\varphi(x_1, \ldots, x_{n-1}, y) = x_n),$$

что означает, что для произвольного набора $(\alpha_1, \ldots, \alpha_n)$ составляется уравнение

$$\varphi(\alpha_1, \ldots, \alpha_{n-1}, y) = \alpha_n.$$

Если существует y из E_0, являющееся решением этого уравнения, то берем минимальное из решений, обозначаем его через μ_y и полагаем

$$f(\alpha_1, \ldots, \alpha_n) = \mu_y.$$

В противном случае функция f не определена.

Про функцию f говорят, что она получена из функции φ при помощи операции минимизации.

Данные операции позволяют построить три следующие функциональные системы.

I. Множество $P_{чр}$ всех функций, которые можно получить из системы элементарных рекурсивных функций $\{0, S(x), I_m^n(x_1, \ldots, x_n), 1 \le m \le n, n = 1, 2, \ldots\}$ при помощи опера-

ций C, Pr и μ, называемое *классом частично-рекурсивных функций.* Здесь $S(x)=x+1$ – функция следования, $0=O^n(x_1,\ ...,\ x_n)$ –функция константы, $I_m^n(x_1,\ ...,\ x_n)=x_m$ – тождественная функция.

II. *Класс рекурсивных функций*, т. е. множество P_p всех всюду определенных функций из $P_{чр}$.

III. *Класс примитивно-рекурсивных функций*, т. е. множество $P_{пр}$ всех функций, которые можно получить из системы $\{0,\ S(x),\ I_m^n(x_1,\ ...,\ x_n),\ 1\le m\le n,\ n=1,\ 2,\ ...\}$ при помощи операций C и Pr.

Очевидно, что

$$P_{пр}\subseteq P_p\subseteq P_{чр}\subseteq P_0^{ч}.$$

Пример 7.5. Докажем, что функция $f(x,\ y)=x+y$ является примитивно-рекурсивной.

Так как заданная функция $f(x,\ y)$ есть функция двух аргументов, то для использования операции примитивной рекурсии необходимо определить функцию φ, зависящую от одного аргумента, и функцию ψ, зависящую от трех аргументов.

В функции $f(x,\ y)=x+y$ положим $y=0$. Тогда получим тождественную функцию $f(x,\ 0)=x$. Полагая $y=1$, получим функцию следования $f(x,\ 1)=x+1$.

Таким образом, выбираем следующие элементарные функции: тождественную функцию $\varphi(x)=x$ и функцию следования $\psi(x,\ y,\ z)=z+1$. Заметим, что $\psi(x,\ y,\ z)=I_3^3(x,\ y,\ S(x))$.

Используем схему примитивной рекурсии:

$$f(x,\ 0)=\varphi(x)=x,$$

$$f(x,\ 1)=\psi(x,\ 0,\ f(x,\ 0))=f(x,\ 0)+1=x+1,$$

$$f(x,\ 2)=\psi(x,\ 1,\ f(x,\ 1))=f(x,\ 1)+1=x+2,$$

$$f(x,\ y)=\psi(x,\ y\text{-}1,\ f(x,\ y\text{-}1))=f(x,\ y\text{-}1)+1=x+y-1+1=x+y.$$

7.5. Формальное определение алгоритма и алгоритмически неразрешимые проблемы

Формальное определение алгоритма. Словарный оператор для алфавита A

$$f : A^* \to A^*,\ \tilde{x} \to \tilde{y} = f(\tilde{x})$$

называется *вычислимым по Тьюрингу*, если существует машина Тьюринга с алфавитом ленты $A \cup \{\Lambda\}$, которая работает следующим образом: если на ленту записать произвольное слово , установить головку на правый символ и запустить машину по ее программе, то после остановки машины на ленте будет написано слово $f(\tilde{x})$.

Отсюда следует одно из возможных формальных определений алгоритма: *алгоритм – это словарный оператор, вычислимый по Тьюрингу.*

Алгоритмически неразрешимые проблемы. Из определения алгоритма следует, что если для решения данной задачи не может быть построена машина Тьюринга, то такая задача становится *алгоритмически неразрешимой.*

Были установлены некоторые алгоритмически неразрешимые задачи.

Первый результат об алгоритмической неразрешимости был получен в 1947 г. независимо друг от друга А.А. Марковым и Э.Л. Постом по проблеме равенства в полугруппах, которая заключается в отыскании алгоритма, выясняющего эквивалентность для любой пары слов.

В 1964 г. М.И. Кратко установил алгоритмическую неразрешимость задачи о полноте автоматного базиса. Этот результат означает, что не существует машины Тьюринга, работающей следующим образом. Если на ленту машины поместить описания автоматов и запустить машину в работу, то машина остановится в состоянии q_0, если система полная и в состоянии q_{00}, если эта система не полная.

Следующая задача – проблема остановки машины Тьюринга, которая формулируется следующим образом. Пусть имеется некоторая программа A и некоторые исходные данные

d. Необходимо для *любой* программы A и *любых* исходных данных d определить, остановится программа A или нет. Было доказано, что данная проблема алгоритмически не разрешима.

Имеются также другие алгоритмически не разрешимые проблемы. Например, проблема распознавания выводимости в математической логике, задача о разрешимости диофантова уравнения (10-я проблема Гильберта), задача о разрешимости в радикалах алгебраического уравнения n-й степени при $n > 4$ и другие. При этом надо иметь в виду, что понятие алгоритмической неразрешимости не означает, что определенная проблема не может быть решена в некоторых частных случаях. Неразрешимость обычно свидетельствует о чрезмерно общей постановке проблемы, о невозможности решения всех задач данного класса одним и тем же приемом. Важность доказательства факта, что некоторая задача алгоритмически не разрешима, состоит в том, что исследователь не будет тратить время и силы на разработку алгоритма решения данной задачи в общем виде.

Список литературы

1. Аляев Ю.А. Дискретная математика и математическая логика. – М.: Финансы и статистика, 2006. – 368 с.
2. Андерсон Д. Дискретная математика и комбинаторика. – М.: Изд. дом «Вильямс», 2004. – 960 с.
3. Гаврилов Г.П., Сапоженко А.А. Задачи и упражнения по дискретной математике. – М.: Физматлит, 2005. – 416 с.
4. Глаголев В.В. Методы дискретной математики. – Тула: Изд-во ТулГУ, 2005. – 256 с.
5. Домнин П.Н. Элементы теории графов. – Пенза: Изд-во ПГУ, 2004. –139 с.
6. Ерусалимский Я.М. Дискретная математика: теория, задачи, приложения. – М.: Вузовская книга, 2000. – 280 с.
7. Комбинаторный анализ. Задачи и упражнения / Пол ред. К.А. Рыбникова. – М.: Наука, 1982. – 368 с.
8. Кристофидес Н. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 с.
9. Кузнецов О.П. Дискретная математика для инженера. – СПб.: Лань, 2004. –400 с.
10. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. – М.: Наука, 2001. – 224 с.
11. Плотников А.Д. Дискретная математика. – М.: Новое знание, 2005. – 288 с.
12. Редькин Н.П. Дискретная математика. – М.: Изд-во МГУ, 2007. – 174 с.
13. Соболева Т.С. Дискретная математика. – М.: Академия, 2006. – 256 с.
14. Тишин В.В. Дискретная математика в примерах и задачах. – СПб: БВХ-Петербург, 2008. –352 с.
15. Хаггарти Р. Дискретная математика для программистов. –М.: Техносфера, 2004. – 320 с.
16. Яблонский С.В. Введение в дискретную математику.– М.: Наука, 2008. – 384 с.