Белорусский государственный университет
Информатики и радиоэлектроники

Дискретная математика

Минск 2015

Вводятся основные понятия теории множеств и отношений, излагаются основы теории графов, абстрактной булевой алгебры с различными интерпретациями. Рассматриваются задачи комбинаторной оптимизации, возникающие при логическом проектировании. Рассматриваются различные методы минимизации булевых функций и систем в классе ДНФ. Описываются формальные методы проектирования комбинационных логических схем на основе теории булевых функций. Излагаются методы логического проектирования, использующие модель конечного автомата в его синхронной и асинхронной реализации.

Предназначается для студентов и аспирантов, специализирующихся в области проектирования дискретных устройств.

Автор – доцент кафедры ЭВМ Белорусского госуниверситета информатики и радиоэлектроники Поттосин Ю.В.

О Г Л А В Л Е Н И Е

Предисловие

В предлагаемом учебном пособии описываются задачи дискретной математики, возникающие при проектировании цифровых устройств и систем. Основное внимание уделяется задачам логического проектирования, т. е. проектирования логики устройств, поскольку данный этап проектирования охватывает наиболее широкий спектр задач, поддающихся строгой математической формулировке.

Излагаются основные понятия теории множеств и отношений. Описывается аппарат булевых векторов и матриц для представления теоретико-множественных объектов и операций над ними. Довольно подробно рассматриваются основы теории графов, абстрактной булевой алгебры с различными интерпретациями, теории булевых функций. Особое внимание уделяется комбинаторным задачам оптимизации: раскраске графов, кратчайшему покрытию множеств и др.

Описаны классические методы минимизации булевых функций и систем булевых функций в терминах булевых и троичных векторов и матриц. Большое внимание уделено задаче минимизации частичных булевых функций, особенно слабо определенных. Рассмотрены также задачи декомпозиции булевых функций.

Изложены методы проектирования цифровых устройств, использующие классические модели конечного автомата. Детально рассматривается задача минимизации числа состояний полных и частичных автоматов. Описаны методы кодирования состояний автомата при синхронной и асинхронной реализации.

Материал подготовлен на основе курса лекций, который читался в течение ряда лет в Белорусском государственном университете информатики и радиоэлектроники. При этом использовались материалы, изложенные в монографиях, список которых приведен в конце данного текста.

Г л а в а 1

Основные понятия теории множеств

1.1. Определения

Под *множеством* обычно понимается совокупность или набор каких-то объектов, имеющих что-то общее, и при этом каждый из них чем-то отличается от другого. Например, множество людей, присутствующих на каком-то мероприятии, множество домов некоторого района города и т. п. Понятие множества является одним из основных понятий математики. Таким же является понятие *элемента множества*. Это исходные понятия, и поэтому точного определения для них нет. Принадлежность элемента a множеству M обозначается как $a \in M$. Если же некоторый элемент a не принадлежит множеству M, то это обозначается как $a \notin M$ или $a \bar{\in} M$.

Любое множество может быть элементом другого множества, которое также может быть элементом некоторого множества, и т. д. (множество множеств, множество множеств множеств и т. д.). Иногда для большего благозвучия вместо словосочетания «множество множеств» употребляют «совокупность множеств» или «семейство множеств».

Множество A является *подмножеством* множества B, если всякий элемент из A принадлежит множеству B. Этот факт обозначается $A \subseteq B$ ($\subseteq$ – знак включения). При этом говорят, что множество B *содержит*, или *покрывает*, множество A. Множества A и B равны ($A = B$), если $A \subseteq B$ и $B \subseteq A$. Множество, не имеющее ни одного элемента, называется *пустым* и обозначается $\varnothing$. Оно является подмножеством любого множества, т. е. $\varnothing \subseteq M$ для любого M. Пустое множество, а также само M являются *несобственными подмножествами* множества M.

Если $A \subseteq B$ и $A \neq B$ для некоторого непустого множества A, то A является *собственным подмножеством* множества B, и это обозначается как $A \subset B$ ($\subset$ – знак строгого включения).

Не следует путать знаки $\subset$ и $\in$, когда рассматриваются множества множеств. Например, зрительный зал можно рассматривать как множество рядов M, каждый из которых, M_i, представляется как множество кресел. Тогда $M_i \in M$ и для отдельного кресла m_j можно записать $m_j \in M_i$. Тот же зрительный зал можно представить как множество M' всех находящихся в нем кресел. Тогда для того же M_i имеет место $M_i \subset M'$.

Множество всех подмножеств некоторого множества M называется *булеаном*. Булеан обозначается символом 2^M. Среди его элементов находятся само множество M, а также пустое множество $\varnothing$.

Множества бывают *конечными* (содержащими конечное число элементов) и *бесконечными*. Параметром, характеризующим размер множества, является

мощность множества. Для конечного множества M мощностью является число элементов, которое обозначается символом $|M|$. Мощность бесконечного множества – более сложное понятие. Оно выражается через соответствие.

Мощность булеана множества M равна $2^{|M|}$. Действительно, $2^{\varnothing} = \{\varnothing\}$, т. е. число элементов булеана пустого множества есть $2^0 = 1$, а добавление к M одного нового элемента каждый раз увеличивает мощность его булеана вдвое (прежние элементы булеана при этом сохраняются, а новые получаются из прежних добавлением к ним данного нового элемента).

Если множества A и B *равномощны*, т. е. $|A| = |B|$, то между ними можно установить *взаимно однозначное соответствие*. Каждому элементу из A ставится в соответствие элемент из B, и наоборот. Для бесконечных множеств отношение равномощности устанавливается путем нахождения взаимно однозначного соответствия между их элементами.

Примерами бесконечных множеств служат: $\mathsf{N} = \{1, 2, \dots\}$ – множество натуральных чисел, $\mathsf{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$ – множество целых чисел, R – множество действительных чисел (рациональные и иррациональные числа).

Множества, равномощные с множеством N, называются *счетными*. Для того чтобы выяснить, является ли некоторое множество M счетным, надо найти способ установить взаимно однозначное соответствие между M и N, т. е. пронумеровать элементы множества M.

У т в е р ж д е н и е 1.1. Любое бесконечное подмножество N множества N счетно.

Действительно, пусть $N \subset \mathsf{N}$. Выберем в N наименьший элемент и обозначим его n_1. Удалим из N элемент n_1 и из оставшихся элементов выберем снова наименьший, который обозначим n_2, и т. д. Таким образом, можно себе представить, что все элементы бесконечного множества N окажутся пронумерованными.

У т в е р ж д е н и е 1.2. Множество P положительных рациональных чисел счетно.

Любое рациональное число можно представить в виде правильной или неправильной дроби $\frac{a}{b}$, где a и b – натуральные числа. Образуем внутри множества P классы $P_1 = \{\frac{1}{1}\}$, $P_2 = \{\frac{1}{2}, \frac{2}{1}\}$, $P_3 = \{\frac{1}{3}, \frac{2}{2}, \frac{3}{1}\}$, … . Здесь в i-м классе ($i = 1, 2, \dots$) собраны все $\frac{a}{b}$, для которых $a + b = i + 1$. Выстроим последовательность из дробей, принадлежащих классам P_i, сохраняя порядок нумерации этих классов. Дроби, принадлежащие одному и тому же классу, упорядочиваются по возрастанию числителя a. В полученной последовательности любая дробь $\frac{a}{b}$ снабжается номером $1 + 2 + \dots + (i - 1) + a$. Следовательно, множество P счетно.

Примером несчетного множества является множество всех действительных чисел отрезка [0, 1]. Такое множество имеет название *континуум*. Булеан бесконечного счетного множества также не является счетным множеством.

1.2. Способы задания множеств

Перечисление элементов. Это простейший способ задания конечного множества. Например, если множество A состоит из элементов $a_1, a_2, \ldots, a_n$, то можно записать $A = \{a_1, a_2, \ldots, a_n\}$.

Указание свойств элементов. При таком способе задается одно или несколько свойств, по которым определяется принадлежность элементов к данному множеству. Если $P(x)$ означает, что x обладает свойством P, то $A = \{x / P(x)\}$ есть множество всех тех и только тех элементов, которые обладают свойством P. Например, $M = \{x / x = 2^k, k \in \mathbb{N}\}$ – множество всех чисел, каждое из которых представляет собой число 2 в натуральной степени.

Индуктивный способ. Задается некоторая порождающая процедура, которая определяет способ получения элементов множества из уже полученных элементов. Например, для бесконечного множества $M = \{1, 2, 4, 8, 16, \ldots\}$ такой определяющей процедурой является следующая: 1) $1 \in M$; 2) если $m \in M$, то $2m \in M$.

Алгебраический способ. При этом способе дается формула, по которой можно получить множество из других множеств с помощью алгебраических операций над ними.

Визуальное представление множеств. Множества изображаются на плоскости в виде фигур, называемых *диаграммами Эйлера –Венна*. Этот способ используется обычно для наглядной демонстрации операций над множествами или отношений между множествами. Пример использования данного способа будет приведен при описании операций над множествами.

Булевы векторы. При рассмотрении конечных множеств вводится универсальное множество (*универсум*), обозначаемое обычно U, и всякое множество, подлежащее рассмотрению, считается подмножеством множества U. Тогда любое множество M представляется вектором с $|U|$ компонентами, которые соответствуют элементам множества U. Компонента этого вектора равна 1, если соответствующий элемент принадлежит множеству M, и 0 – в противном случае. Пусть $U = \{a, b, c, d, e\}$ и $M = \{a, b, d\}$. Тогда M представится вектором 11010. Векторы 00000 и 11111 задают соответственно пустое множество $\varnothing$ и универсальное множество U.

1.3. Операции над множествами

Как было сказано выше, множество можно представить в виде результата операций над другими множествами.

Объединение множеств A и B представляет собой множество, содержащее те и только те элементы, которые принадлежат A или B (хотя бы одному из этих множеств):

$$A \cup B = \{x / x \in A \text{ или } x \in B\}.$$

Пересечением множеств A и B является множество, содержащее те и только те элементы, каждый из которых принадлежит как A, так и B:

$$A \cap B = \{x / x \in A \text{ и } x \in B\}.$$

Разность множеств A и B состоит из элементов множества A, которые не принадлежат множеству B:

$$A \setminus B = \{x / x \in A \text{ и } x \notin B\}.$$

Сумма множеств A и B (ее называют еще *симметрической разностью* множеств A и B) содержит все элементы из A, не принадлежащие B, и все элементы из B, не принадлежащие A:

$$A + B = \{x / (x \in A \text{ и } x \notin B) \text{ или } (x \in B \text{ и } x \notin A)\}.$$

Дополнение множества A состоит из элементов универсального множества U, не принадлежащих A:

$$\bar{A} = \{x / x \in U \text{ и } x \notin A\}.$$

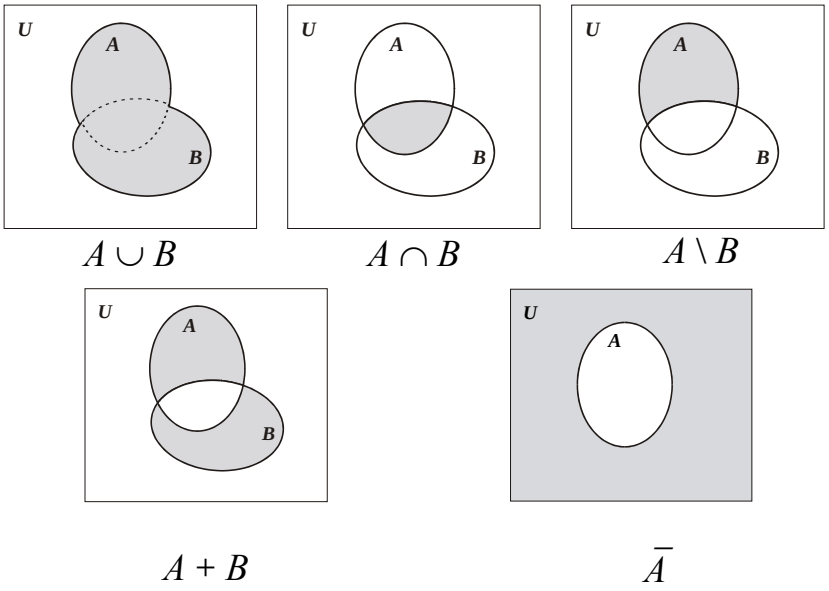
На рис. 1.1 затемненными областями на диаграммах Эйлера –Венна показаны результаты перечисленных операций.

Таким образом, формула, в которой присутствуют символы операций над множествами, есть способ задания множества. Две формулы *равносильны*, если они представляют одно и то же множество. Некоторые операции можно выразить через другие. Так, например, имеем

$$A + B = (A \cap \bar{B}) \cup (\bar{A} \cap B) = (A \cup B) \setminus (A \cap B);$$
$$\bar{A} = U \setminus A;$$
$$A \setminus B = A \cap \bar{B}.$$

Три из перечисленных операций, дополнение, пересечение и объединение, составляют *булеву алгебру множеств*. Перечислим основные законы этой алгебры, используя общепринятое правило: если в формуле отсутствуют скобки, устанавливающие порядок выполнения операций, то сначала выполняется дополнение, потом пересечение и затем объединение. Для

повышения компактности формулы знак пересечения множеств, подобно знаку арифметического умножения, будем опускать.



$A \cup B$ $A \cap B$ $A \setminus B$

$A + B$ $\overline{A}$

Рис. 1.1. Операции над множествами

Коммутативность:
$A \cup B = B \cup A$; $A\ B = B\ A$.

Ассоциативность:
$A \cup (B \cup C) = (A \cup B) \cup C$; $A\ (B\ C) = (A\ B)\ C$.

Дистрибутивность:
$A\ (B \cup C) = A\ B \cup A\ C$; $A \cup B\ C = (A \cup B)\ (A \cup C)$.

Идемпотентность:
$A \cup A = A$; $A\ A = A$.

Законы де Моргана:
$\overline{A \cup B} = \overline{A}\ \overline{B}$; $\overline{AB} = \overline{A} \cup \overline{B}$.

Законы операций с константами (пустым и универсальным множествами):
$A \cup \varnothing = A$; $A\ U = A$;
$A \cup U = U$; $A\ \varnothing = \varnothing$;
$A \cup \overline{A} = U$; $A\ \overline{A} = \varnothing$.

Закон двойного дополнения:
$\overline{\overline{A}} = A$.

Заметим, что для каждой пары формул, представляющих тот или иной закон, справедливо следующее: одна из формул получается из другой взаимной заменой всех операций пересечения на операции объединения и всех символов $\varnothing$ на символы U. При этом должен быть сохранен порядок действий. Этот факт известен под названием *принципа двойственности*.

Любое равенство из булевой алгебры множеств можно вывести путем равносильных преобразований, используя формулы из приведенного списка.

Например, известная как *закон поглощения* формула $A \cup A\,B = A$, которой нет в приведенном списке, выводится следующим образом:

$$A \cup A\,B = A\,U \cup A\,B = A\,(U \cup B) = A\,U = A.$$

Используя принцип двойственности, получим

$$A\,(A \cup B) = A.$$

Список формул, приведенный выше, является достаточным, но для вывода любого равенства из данной алгебры можно воспользоваться меньшим списком, т. е. некоторые формулы этого списка можно вывести из других. Например, формулу

$$A \cup B\,C = (A \cup B)\,(A \cup C)$$

(дистрибутивность объединения относительно пересечения) можно получить следующим образом. Ее правую часть, используя дистрибутивность пересечения, представим как

$$(A \cup B)\,A \cup (A \cup B)\,C.$$

Раскрыв скобки (по закону ассоциативности), получим

$$A\,A \cup B\,A \cup A\,C \cup B\,C.$$

Применим закон идемпотентности и используем константу U ($A\,A = A = A\,U$), в результате чего после применения закона коммутативности пересечения правая часть примет вид $A\,U \cup A\,B \cup A\,C \cup B\,C$. После вынесения за скобки A получим $A\,(U \cup B \cup C) \cup B\,C$, что равно левой части исходного выражения согласно свойству константы U.

Выведем теперь *закон простого склеивания* $A\,B \cup \bar{A}\,B = B$:

$$A\,B \cup \bar{A}\,B = B\,(A \cup \bar{A}) = B\,U = B.$$

Формулу $A\,B \cup \bar{A}\,C = A\,B \cup \bar{A}\,C \cup B\,C$ (*обобщенное склеивание*) выведем следующим образом:

$$A\,B \cup \bar{A}\,C \cup B\,C = A\,B \cup \bar{A}\,C \cup B\,C\,(A \cup \bar{A}) =$$
$$= A\,B\,(U \cup C) \cup \bar{A}\,C\,(U \cup B) = A\,B \cup \bar{A}\,C.$$

Используя только что выведенную формулу и закон поглощения, докажем $A \cup \bar{A}\,B = A \cup B$:

$$A \cup \bar{A}\,B = A\,U \cup \bar{A}\,B = A\,U \cup \bar{A}\,B \cup U\,B =$$
$$= A \cup \bar{A}\,B \cup B = A \cup B.$$

Г л а в а 2

Отношения бинарные и *n*-арные

2.1. Декартово произведение

Декартовым, или *прямым*, *произведением* двух множеств A и B (обозначается $A \times B$) называется множество всех таких упорядоченных пар (a, b), что $a \in A$ и $b \in B$. Пусть, например, $A = \{a, b, c\}$ и $B = \{l, m\}$. Тогда $A \times B = \{(a, l), (b, l), (c, l), (a, m), (b, m), (c, m)\}$. Это понятие распространяется на случай с более чем одним сомножителем. Декартово произведение множеств $A_1, A_2, \ldots, A_n$ (обозначается $A_1 \times A_2 \times \ldots \times A_n$) есть множество всех векторов $(a_1, a_2, \ldots, a_n)$ размерности n, таких, что $a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n$.

Декартово произведение n одинаковых сомножителей $A \times A \times \ldots \times A$ обозначается символом A^n и называется n-й *степенью* множества A. При этом $A^1 = A$. Примером декартова произведения является $\mathsf{R} \times \mathsf{R} = \mathsf{R}^2$ – множество точек на плоскости. Здесь элементы $x \in \mathsf{R}$ и $y \in \mathsf{R}$ служат координатами некоторой точки на плоскости. Другим примером является множество R^3 точек в трехмерном евклидовом пространстве. Обобщением этих понятий является n-мерное пространство.

Любое подмножество $R \subseteq A_1 \times A_2 \times \ldots \times A_n$ декартова произведения n множеств называется *n-арным отношением*. При $n = 1, 2, 3$ имеем *унарное*, *бинарное*, *тернарное* отношения соответственно. Унарное отношение на множестве A представляет собой подмножество множества A.

2.2. Бинарные отношения (соответствия)

Бинарным отношением, или *соответствием* между элементами множеств A и B, называется любое подмножество $R \subseteq A \times B$ декартова произведения этих множеств. Тот факт, что некоторые $a \in A$ и $b \in B$ находятся в отношении R, иногда выражают как $a\,R\,b$. В качестве примера бинарного отношения рассмотрим отношение R между элементами множеств $A = \{1, 2, 3\}$ и $B = \{1, 2, 3, 4, 5, 6\}$, которое можно выразить словами так: элемент $x \in A$ есть делитель элемента $y \in B$. Тогда имеем $R = \{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 2), (2, 4), (2, 6), (3, 3), (3, 6)\}$.

Бинарное отношение удобно представлять в виде двоичной (*булевой*) матрицы. При этом элементы множеств A и B должны быть пронумерованы, и если i-й элемент множества A соответствует j-му элементу множества B, то элемент матрицы, расположенный на пересечении i-й строки и j-го столбца, имеет значение 1, в противном случае он имеет значение 0. Например, рассмотренное выше отношение R будет представлено следующей матрицей:

$$\begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \end{matrix}.$$

Проекция элемента (a, b) множества $A \times B$ на множество A есть элемент a. Аналогично элемент b является проекцией элемента (a, b) множества $A \times B$ на множество B.

Проекцией множества $E \subset A \times B$ на A называется множество всех тех элементов из A, которые являются проекциями элементов из E на множество A. Для множеств A и B, рассмотренных выше, проекцией элемента $(2, 4)$ на множество A является элемент 2, а проекцией множества $\{(1, 2), (2, 2), (2, 4)\}$ – множество $\{1, 2\}$.

Сечением множества $R \subset A \times B$ по a, обозначаемым $R(a)$, называется множество всех тех элементов $y \in B$, для которых $(a, y) \in R$.

Сечением $R(X)$ множества R по $X \subset A$ является объединение сечений для всех элементов из X. Пусть $R = \{(1, 1), (1, 3)\ (1, 5),\ (1, 6),\ (2, 2),\ (2, 4),\ (3, 3),\ (3, 6)\}$. Тогда $R(2) = \{2, 4\}$, а если $X = \{2, 3\}$, то $R(X) = \{2, 3, 4, 6\}$.

Бинарное отношение можно задавать с помощью сечений. Например, отношение, представленное матрицей

$$\begin{array}{c} \begin{matrix} b_1 & b_2 & b_3 & b_4 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{matrix},$$

можно задать следующим образом: $R(a_1) = \{b_1, b_3\}$, $R(a_2) = \{b_1, b_3, b_4\}$, $R(a_3) = \{b_1, b_4\}$, $R(a_4) = \varnothing$, $R(a_5) = \{b_4\}$. Множество сечений для всех $a \in A$ называется *фактор-множеством*.

Областью определения отношения $R \subseteq A \times B$ является проекция множества R на A. Для рассматриваемого выше отношения такой областью является $\{a_1, a_2, a_3, a_5\}$. *Областью значений* отношения $R \subseteq A \times B$ является сечение множества R по A. Областью значений рассматриваемого отношения R является $\{b_1, b_3, b_4\}$.

Образом множества $X \subseteq A$ относительно R называется множество $\{b \,/\, b \in B,\ x \in X,\ (x, b) \in R\}$. *Прообразом* множества $Y \subseteq B$ относительно R называется множество $\{a \,/\, a \in A,\ y \in Y,\ (a, y) \in R\}$. В нашем последнем

примере образом множества $\{a_1, a_3\}$ относительно R является $\{b_1, b_3, b_4\}$, а прообразом множества $\{b_3, b_4\}$ является $\{a_1, a_2, a_3, a_5\}$.

Обратным отношением R^{-1} для некоторого отношения $R \subseteq A \times B$ является множество, образованное теми парами $(b, a) \in B \times A$, для которых $(a, b) \in R$. Матрица, представляющая отношение R^{-1}, получается транспонированием матрицы, представляющей R, т. е. заменой строк столбцами и наоборот.

Например, рассмотренному выше отношению R будет соответствовать обратное отношение R^{-1}, представляемое матрицей

$$\begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{matrix}.$$

2.3. Операции над бинарными отношениями

Поскольку всякое отношение есть некоторое множество пар, над отношениями применимы все стандартные операции над множествами, т. е. объединение, пересечение, дополнение. Универсальным множеством для операции дополнения при этом является $A \times B$.

Рассмотрим операцию *композиции* отношений. Заданы множества A, B, C и отношения $R \subseteq A \times B$ и $S \subseteq B \times C$. Композиция отношений S и R (обозначается SR, не путать с пересечением множеств S и R!) – это такое отношение между элементами множеств A и C, что для всех $a \in A$ сечение множества SR по a совпадает с сечением множества S по подмножеству $R(a) \subseteq B$. Это записывается в виде $(SR)(a) \subseteq S(R(a))$. Например, пусть отношения R и S заданы соответственно следующими матрицами:

$$\boldsymbol{R} = \begin{array}{c} \begin{matrix} b_1 & b_2 & b_3 & b_4 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{matrix}, \quad \boldsymbol{S} = \begin{array}{c} \begin{matrix} c_1 & c_2 & c_3 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{matrix}.$$

Тогда композиция SR этих отношений представится матрицей

$$\boldsymbol{SR} = \begin{array}{c} \begin{matrix} c_1 & c_2 & c_3 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{matrix}.$$

2.4. Функциональные отношения

Отношение $R \subseteq A \times B$ называется *функциональным*, если для каждого $a \in A$ сечение множества R по a содержит не более одного элемента, т.е. для каждого a справедливо $|\{b / (a, b) \in R, b \in B\}| \leq 1$. В функциональном отношении не существует пар с одинаковым левым элементом и различными правыми элементами, т. е. если $(a, b) \in R$ и R – функциональное отношение, то в R не может быть пары вида (a, c), где $b \neq c$. Матрица, представляющая функциональное отношение, в каждой строке имеет не более одной единицы. Примером может служить следующая матрица:

$$\begin{array}{c} \begin{matrix} b & d & e \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix}.$$

Если сечение функционального отношения R по любому элементу a из множества A содержит один и только один элемент, то отношение R называется *всюду определенным*.

Если отношение R^{-1}, обратное для функционального отношения R, также является функциональным, то отношение R называется *взаимно однозначным*.

Для всякого функционального отношения $R \subseteq A \times B$ можно определить *функцию*, связанную с этим отношением. Для обозначения функции используется запись $f : A \to B$. Если $(x, y) \in R$, то это можно выразить как $y = f(x)$, где x является *аргументом*, а y – *значением функции* f.

Множество $\{x / (x, y) \in R\}$ называется *областью определения* функции f. Если это множество совпадает с A, то функция f является *всюду определенной*. Такая функция называется *отображением* множества A в B. В противном случае функцию называют *частичной*.

Множество $\{y / (x, y) \in R\}$ называется *областью значений* функции f. Если область значений функции f совпадает с множеством B, то f называют отображением A на B, *сюръективным* отображением, или *сюръекцией*. Обязательным условием существования отображения A на B является $|A| \geq |B|$.

Если функциональное отношение $R \subseteq A \times B$, определяющее функцию f, является взаимно однозначным, то функцию f называют *инъективным* отображением, или *инъекцией*. В этом случае существует функция f^{-1}, которая является *обратной* к функции f. При этом если $y = f(x)$, то $x = f^{-1}(y)$, а мощность области определения функции f не должна превышать $|B|$.

Функция f называется *биективным* отображением, или *биекцией*, если она является как сюръективным, так и инъективным отображением. Такое отображение называется еще *1-1 соответствием*.

Если R – взаимно однозначное отношение между элементами одного и того же множества, т. е. $R \subseteq A \times A = A^2$, и, кроме того, R и R^{-1} всюду определены, то отображение, связанное с R, называется *подстановкой*.

На рис. 2.1 изображены схемы рассмотренных видов отображений.

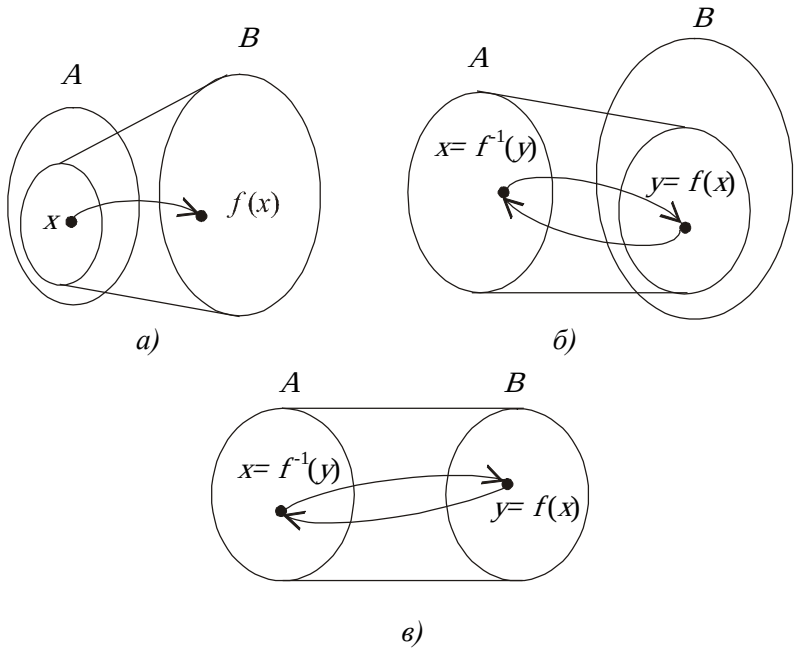


Рис. 2.1. Схемы функциональных отображений: *а)* сюръекция; *б)* инъекция; *в)* биекция

Функция, определенная на множестве натуральных чисел, называется *последовательностью*, а каждое ее значение – *членом* последовательности.

Отображение f произвольного множества в множество действительных чисел называется *функционалом*. Примером функционала может служить определенный интеграл.

Отображение $f: A \to B$, где A и B – некоторые множества функций, называется *оператором*. Оператор преобразует одну функцию в другую. Примером оператора является оператор суперпозиции функций, где

аргументами некоторых функций служат другие функции. Это понятие будет использовано далее.

2.5. Бинарные отношения на множестве

Пусть $R \subseteq A \times A$. Определим некоторые свойства, которыми может обладать или не обладать такое отношение:

рефлексивность: если $a = b$, то $a\ R\ b$;

иррефлексивность: если $a\ R\ b$, то $a \neq b$;

симметричность: если $a\ R\ b$, то $b\ R\ a$;

антисимметричность: если $a\ R\ b$ и $b\ R\ a$, то $a = b$;

транзитивность: если $a\ R\ b$ и $b\ R\ c$, то $a\ R\ c$;

дихотомия: если $a \neq b$, то либо $a\ R\ b$, либо $b\ R\ a$.

Следует выделить некоторые типы бинарных отношений, характеризуемые определенным набором свойств.

Отношение *эквивалентности* рефлексивно, симметрично и транзитивно. Примерами отношения эквивалентности являются равносильность формул, подобие геометрических фигур, принадлежность студентов к одной группе, принадлежность населенных пунктов к одному району и т. п.

Отношение эквивалентности делит множество на непересекающиеся подмножества – *классы эквивалентности*. С другой стороны, всякое разбиение множества M на непересекающиеся подмножества задает отношение эквивалентности на множестве M: любые два элемента, принадлежащие одному и тому же классу разбиения, эквивалентны, а элементы, принадлежащие различным классам, не являются эквивалентными. Множество всех классов эквивалентности образует *фактор-множество* множества M по R (обозначается M / R).

Отношение *совместимости* рефлексивно и симметрично. Примерами отношения совместимости являются близость чисел, знакомство людей и т. п.

Отношение *нестрогого порядка* рефлексивно, антисимметрично и транзитивно. Отношения $\leq$ (меньше или равно) и $\geq$ (больше или равно) для действительных чисел так же, как $\subseteq$ и $\supseteq$ для множеств являются отношениями нестрогого порядка.

Отношение *строгого порядка* иррефлексивно, антисимметрично и транзитивно. Отношениями строгого порядка являются $<$ (меньше) и $>$ (больше) для действительных чисел, а также $\subset$ и $\supset$ для множеств.

Множество M, на котором задано отношение порядка R (строгого или нестрогого), может быть *полностью* упорядоченным, если любые два элемента a и b из M находятся в отношении R, т. е. $a\ R\ b$ или $b\ R\ a$. При этом говорят, что a и b *сравнимы*. Если M содержит хотя бы одну пару элементов c и d, для которых не имеет место ни $c\ R\ d$, ни $d\ R\ c$, то множество M является *частично* упорядоченным, а указанные элементы c и d *несравнимы*. Отношение *полного порядка* обладает свойствами иррефлексивности, антисимметричности и дихотомии. Полный порядок называют еще *линейным* или *совершенным*.

Для множества действительных чисел R отношения $\leq$ и $<$ являются отношениями полного порядка. Для семейства подмножеств некоторого множества M отношение $\subseteq$ является отношением частичного порядка. Например, $\{a_1, a_3\} \subseteq \{a_1, a_2, a_3\}$, а подмножества $\{a_1, a_3\}$ и $\{a_1, a_2, a_4\}$ несравнимы.

Порядок букв в алфавите и естественный порядок цифр являются полными порядками. На основе порядка букв строится *лексикографический* порядок слов, используемый в словарях и определяемый следующим образом.

Обозначим это отношение порядка символом $\prec$. Пусть имеются слова $w_1 = a_{11}a_{12} \ldots a_{1m}$ и $w_2 = a_{21}a_{22} \ldots a_{2n}$. Тогда $w_1 \prec w_2$, если и только если либо $w_1 = pa_iq$, $w_2 = pa_jr$ и $a_i \prec a_j$, где p, q и r – некоторые слова, возможно, пустые, а a_i и a_j – буквы, либо $w_2 = w_1p$, где p – непустое слово.

Например, учебник $\prec$ ученик и мор $\prec$ море. В первом случае p = уче, a_i = б, a_j = н, q = ник, r = ик, и в алфавите буква «н» стоит дальше буквы «б». Потому в словаре слово «ученик» следует искать после слова «учебник». Во втором случае w_1 = мор и p = е. Согласно лексикографическому порядку слово «море» должно быть помещено в словаре после слова «мор».

Г л а в а 3

Основные понятия теории графов

3.1. Абстрактный граф

Граф можно определить как совокупность двух множеств: $G = (V, E)$, где V – непустое множество, элементы которого называются *вершинами*, и E – произвольное множество пар (v_i, v_j) элементов из множества V, т. е. $v_i \in V$, $v_j \in V$, $E \subseteq V^2$. Элементы множества E называются *ребрами*.

Само понятие графа подразумевает графическое представление данного объекта. Вершины изображаются точками, а ребра – линиями, соединяющими эти точки. Если ребра представляют упорядоченные пары вершин, соответствующие линии изображаются стрелками (рис. 3.1). Такие ребра называют *ориентированными ребрами* или, чаще, *дугами*. В этом случае имеем дело с *ориентированным* графом в отличие от *неориентированного* графа, на ребрах которого порядок вершин не задан.

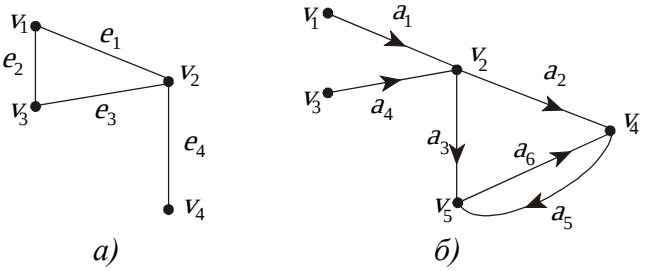


Рис. 3.1. Примеры графов: *а)* неориентированный;
б) ориентированный

Вершины неориентированного графа, связываемые ребром, считаются *концами* этого ребра. Например, концами ребра e_2 графа на рис. 3.1, *а* являются вершины v_1 и v_3. Принято обозначать ребра также парами их концов, например $e_2 = v_1v_3$. Всякая упорядоченная пара вершин (v_i, v_j), представляющая дугу в ориентированном графе, имеет *начало* v_i и *конец* v_j. Говорят, что дуга *выходит* из начала и *входит* в конец. В ориентированном графе на рис. 3.1, *б* началом дуги a_4 является вершина v_3 и концом – вершина v_2. Это можно представить как $a_4 = (v_3, v_2)$.

Между вершинами и ребрами неориентированного графа так же, как между вершинами и дугами ориентированного графа, существует отношение *инцидентности*. При этом в неориентированном графе $G = (V, E)$ вершина $v \in V$ и ребро $e \in E$ *инцидентны*, если v является одним из концов ребра e. В ориентированном графе $G = (V, A)$ вершина $v \in V$ и дуга $a \in A$ *инцидентны*, если v является началом либо концом дуги a. Две вершины

неориентированного графа *смежны*, если они инцидентны одному и тому же ребру.

Граф может содержать *петли*, т. е. ребра, концы которых совпадают, или дуги, у которых начало совпадает с концом. Очевидно, ориентация петли несущественна.

Множество всех вершин графа G, смежных с вершиной v, называется *окрестностью* вершины v и обозначается символом $N(v)$. Мощность множества $N(v)$, обозначаемая $d(v)$, называется *степенью* вершины v. В ориентированном графе с некоторой вершиной v подобным образом связаны два множества: *полуокрестность исхода* $N^+(v)$ – множество вершин, в которые входят дуги, исходящие из вершины v, и *полуокрестность захода* $N^-(v)$ – множество вершин, из которых исходят дуги, заходящие в v. Соответственно мощность множества $N^+(v)$ называется *полустепенью исхода* и обозначается $d^+(v)$, а мощность множества $N^-(v)$ – *полустепенью захода* и обозначается $d^-(v)$. Можно говорить об окрестности $N(v)$ и степени $d(v)$ вершины v ориентированного графа. При этом

$$N(v) = N^+(v) \cup N^-(v) \quad \text{и} \quad d(v) = d^+(v) + d^-(v).$$

Для неориентированного графа с множеством ребер E очевидно следующее соотношение:

$$\sum_{v \in V} d(v) = 2|E|,$$

откуда следует, что в любом неориентированном графе число вершин с нечетной степенью всегда четно.

Для ориентированного графа с множеством дуг A имеем

$$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v) = |A|.$$

В практических приложениях граф (ориентированный или неориентированный), как правило, является *конечным*, т. е. его множество вершин конечно. Специальный раздел теории графов изучает также *бесконечные графы*, у которых множество вершин бесконечно.

Граф $G = (V, E)$, у которого множество ребер пусто, т. е. $E = \varnothing$, называется *пустым* графом. Неориентированный граф называется *полным*, если любые две его вершины смежны. Полный граф, число вершин которого n, обозначается символом K_n.

Обозначим множество ребер полного графа символом U. *Дополнением* графа $G = (V, E)$ является граф $\bar{G} = (V, \bar{E})$, у которого $\bar{E} = U \setminus E$. Очевидно, что

всякий полный граф является дополнением некоторого пустого графа и, наоборот, всякий пустой граф является дополнением некоторого полного графа.

Граф называется *двудольным*, если множество его вершин V разбито на два непересекающихся подмножества V' и V'', а концы любого его ребра находятся в различных подмножествах. Такой граф задается как $G = (V', V'', E)$ или как $G = (V', V'', A)$. В *полном двудольном* графе (V', V'', E) каждая вершина из V' связана ребром с каждой вершиной из V''. Полный двудольный граф, у которого $|V'| = p$ и $|V''| = q$, обозначается символом $K_{p, q}$.

3.2. Графическое представление бинарного отношения

Наглядными примерами графов служат схемы железных дорог, помещаемые на стенах больших вокзалов, и схемы авиалиний в аэропортах. Характерным для таких схем является несоблюдение масштаба, несмотря на то, что они изображаются на фоне очертания страны или контуров материков земного шара. Тем самым подчеркивается, что здесь важна связь (бинарное отношение «есть линия») между населенными пунктами, но не расстояние.

Граф в том виде, как он определен выше, является, по сути дела, графическим представлением бинарного отношения. Пусть задано бинарное отношение $R \subseteq A \times B$. Если $A \cap B = \varnothing$, то данное отношение можно представить двудольным ориентированным графом $G = (A, B, R)$, где каждая пара $(a, b) \in R$ представляется дугой, исходящей из вершины a и заходящей в вершину b. На рис. 3.2 представлено отношение R между элементами множеств A и B, где $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3, b_4, b_5\}$, $R = \{(a_1, b_1), (a_1, b_2), (a_1, b_3), (a_1, b_5), (a_2, b_2), (a_2, b_4), (a_3, b_3)\}$.

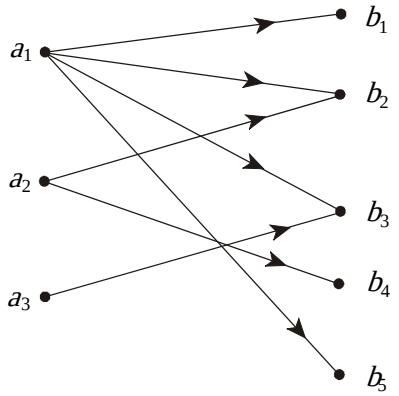


Рис. 3.2. Графическое представление отношения между элементами множеств A и B

Операция композиции отношений, рассмотренная в предыдущей главе, проиллюстрирована на рис. 3.3, где отношение R между элементами множеств A и B и отношение S между элементами множеств B и C показаны совместно

(рис. 3.3, *а*). В представлении отношения *SR* на рис. 3.3, *б* видно, что вершина $a \in A$ соединена с вершиной $c \in C$ дугой тогда и только тогда, когда существует вершина $b \in B$, которая в графе на рис. 3.3, *а* является концом дуги, исходящей из *a*, и началом дуги, заходящей в *c*.

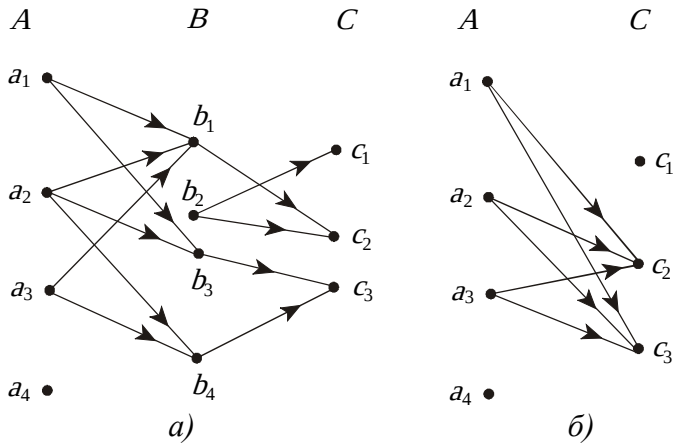


Рис. 3.3. Представление композиции отношений: *а)* отношения *R* и *S*; *б)* отношение *SR*

В графическом представлении функционального отношения $R = \{(a, b), (c, b), (b, d), (e, d), (d, d)\}$ между элементами множеств $A = \{a, b, c, d, e\}$ и $B = \{b, d, e\}$, рассмотренного в предыдущей главе, из каждой вершины выходит только одна дуга, включая петли (рис. 3.4).

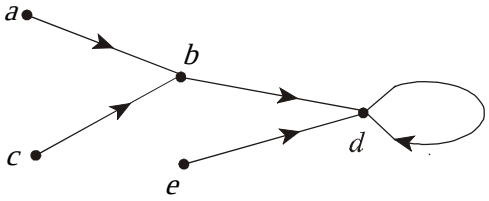


Рис. 3.4. Представление функционального отношения

3.3. Матричные представления графа

Поскольку граф можно рассматривать как графическое представление некоторого бинарного отношения, его можно задать той же булевой матрицей, которая задает данное отношение и описана в предыдущей главе. Эта матрица называется *матрицей смежности* графа. Строки и столбцы матрицы смежности соответствуют вершинам графа, а элемент ее на пересечении строки v_i и столбца v_j имеет значение 1 тогда и только тогда, когда вершины v_i и v_j смежны. В матрице смежности ориентированного графа этот элемент имеет значение 1, если и только если в данном графе имеется дуга с началом в

вершине v_i и концом в вершине v_j. Графы, показанные на рис. 3.1, имеют следующие матрицы смежности:

$$\begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix}, \quad \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix}.$$

Нетрудно видеть, что матрица смежности неориентированного графа обладает симметрией относительно главной диагонали.

Ориентированный граф можно задать также *матрицей инцидентности*, которая определяется следующим образом. Ее строки соответствуют вершинам графа, столбцы – дугам. Элемент на пересечении строки v и столбца a имеет значение 1, если вершина v является началом дуги a, и значение –1, если v является концом дуги a. Если вершина v и дуга a не инцидентны, то указанный элемент имеет значение 0. Матрица инцидентности неориентированного графа имеет тот же вид, только в ней все значения –1 заменяются на 1. Матрицы инцидентности графов на рис. 3.1 будут иметь следующий вид:

$$\begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix}, \quad \begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 0 & -1 & 1 \end{bmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix}.$$

Заметим, что при матричном представлении графа его вершины, а также ребра или дуги оказываются упорядоченными. Любая строка матрицы смежности является векторным представлением окрестности соответствующей вершины. Любой столбец матрицы инцидентности неориентированного графа содержит ровно две единицы. Сумма значений элементов любого столбца матрицы инцидентности ориентированного графа равна нулю.

3.4. Части графа

Граф $H = (W, F)$ называется *подграфом* графа $G = (V, E)$, если $W \subseteq V$, $F \subseteq E$ и обе вершины, инцидентные любому ребру из F, принадлежат W. Подграф H графа G называется его *остовным подграфом*, если $W = V$. Если F является

множеством всех ребер графа G, все концы которых содержатся в множестве W, то подграф $H = (W, F)$ называется *подграфом, порожденным множеством* W.

Любая последовательность вида $v_1, e_1, v_2, e_2, \ldots, e_k, v_{k+1}$, где $v_1, v_2, \ldots, v_{k+1}$ – вершины некоторого графа, а $e_1, e_2, \ldots, e_k$ – его ребра, причем $e_i = v_i v_{i+1}$ ($i = 1, 2, \ldots, k$), называется *маршрутом*. Маршрут может быть конечным либо бесконечным. Одно и то же ребро может встречаться в маршруте не один раз. *Длиной маршрута* называется количество входящих в него ребер, причем каждое ребро считается столько раз, сколько оно встречается в данном маршруте.

Маршрут, все ребра которого различны, называется *цепью*. Цепь, все вершины которой различны, называется *простой цепью*. С понятием длины цепи связано понятие *расстояния* в графе. Под расстоянием между двумя вершинами понимается длина кратчайшей цепи, связывающей данные вершины.

Маршрут $v_1, e_1, v_2, e_2, \ldots, e_k, v_1$ называется *циклическим*. Циклическая цепь называется *циклом*. *Простой цикл* – это циклическая простая цепь.

Любую цепь и любой цикл графа можно рассматривать как его подграф.

Граф является *связным*, если между любыми двумя его вершинами имеется цепь. Связный подграф некоторого графа, не содержащийся ни в каком другом его связном подграфе, называется *компонентой связности* или просто *компонентой* данного графа.

В ориентированном графе *маршрутом* называется последовательность вида $v_1, a_1, v_2, a_2, \ldots, a_k, v_{k+1}$, где для всякой дуги a_i вершина v_i является началом, а v_{i+1} – концом. Вершина v_1 является началом маршрута, а вершина v_{k+1} – его концом. Маршрут, в котором все вершины, кроме, возможно, начальной и конечной, различны, называется *путем*. Путь вида $v_1, a_1, v_2, a_2, \ldots, a_k, v_1$ называется *контуром*.

Вершина v_j в ориентированном графе является *достижимой* из вершины v_j, если в этом графе имеется путь с началом в v_i и концом в v_j. Ориентированный граф является *сильно связным*, если любая его вершина достижима из любой вершины.

Ориентированный граф называется *транзитивным*, если из существования дуг $a_p = (v_i, v_j)$ и $a_q = (v_j, v_k)$ следует существование дуги $a_r = (v_i, v_k)$. *Транзитивным замыканием* ориентированного графа $G = (V, A)$ называется граф $G^* = (V, A^*)$, где A^* получено из A добавлением минимально возможного количества дуг, необходимого для того, чтобы граф G^* был транзитивным.

3.5. Обобщения графов

Существуют различные обобщения понятия графа. Одним из таких обобщений является *мультиграф*. Это граф, в котором любые две вершины могут быть связаны любым количеством ребер, т. е. мультиграф допускает *кратные ребра*.

В некоторых задачах используются графы, на множествах вершин или ребер которых заданы функции, принимающие значения из множеств действительных, целых или натуральных чисел. Эти значения называются *весами*. Тогда речь идет о *взвешенных графах*, о графах со *взвешенными вершинами*, со *взвешенными ребрами* или со *взвешенными дугами*. Графы со взвешенными ребрами используются в транспортных задачах и в задачах о потоках в сетях. Мультиграф можно рассматривать как граф, ребра которого взвешены натуральными числами, представляющими кратности ребер.

Иногда рассматриваются *смешанные графы*, у которых наряду с элементами ориентированного графа (дугами) имеются элементы неориентированного графа (ребра). Ребром может быть заменена пара противоположно направленных дуг в ориентированном графе, соединяющих одни и те же вершины. Смешанные графы используются при решении задач, связанных с установлением схемы выполнения операций в технологическом процессе.

Еще одним обобщением понятия графа является *гиперграф*, который также представляет собой два множества – множество вершин и множество ребер, однако если ребром графа является пара вершин, то ребром гиперграфа может быть любое непустое подмножество множества вершин.

Гиперграф может служить моделью принципиальной электрической схемы. При этом полюса элементов данной схемы соответствуют вершинам гиперграфа, а электрические цепи – ребрам. Электрическая цепь здесь рассматривается как множество выводов, соединенных между собой проводниками. Многие понятия, связанные с графами, распространяются на случай гиперграфа, однако графически изобразить гиперграф гораздо труднее, чем граф. Вместе с тем от гиперграфа можно перейти к двудольному графу, долями которого являются множество вершин и множество ребер гиперграфа, а ребра показывают принадлежность вершин гиперграфа его ребрам.

Г л а в а 4

Изоморфизм графов

4.1. Отношение изоморфизма

Два графа $G = (V, E)$ и $H = (W, F)$ *изоморфны*, если между их множествами вершин имеется взаимно однозначное соответствие, сохраняющее отношение смежности. Другими словами, имеют место $\varphi : V \to W$, $\psi : E \to F$, и если $\varphi(v_i) = w_k$ и $\varphi(v_j) = w_l$, то $\psi(v_i v_j) = w_k w_l$. Графы, изображенные на рис. 4.1, являются изоморфными, причем $\varphi(v_1) = w_1$, $\varphi(v_2) = w_3$, $\varphi(v_3) = w_6$, $\varphi(v_4) = w_4$, $\varphi(v_5) = w_5$, $\varphi(v_6) = w_2$. Соответствие ψ определяется однозначно по φ: $\psi(v_1v_4) = w_1w_4$, $\psi(v_1v_5) = w_1w_5$, $\psi(v_1v_6) = w_1w_2$, $\psi(v_2v_4) = w_3w_4$, $\psi(v_2v_5) = w_3w_5$, $\psi(v_2v_6) = w_2w_3$, $\psi(v_3v_4) = w_4w_6$, $\psi(v_3v_5) = w_5w_6$, $\psi(v_3v_6) = w_2w_6$.

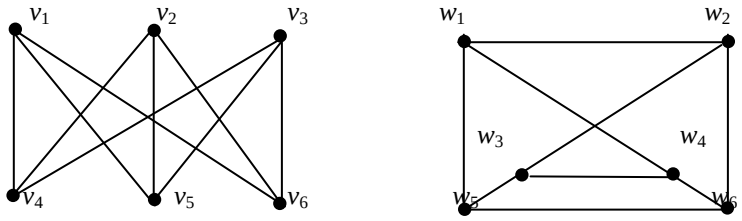


Рис. 4.1. Изоморфные графы

Задача установления изоморфизма формулируется следующим образом. Заданы два графа $G = (V, E)$ и $H = (W, F)$. Требуется установить, изоморфны они или нет, и если изоморфны, то определить соответствие φ между вершинами (как было указано выше, соответствие ψ определяется однозначно по φ). Данная задача может иметь приложение при контроле больших интегральных схем.

Очевидно, необходимым условием изоморфизма двух графов является равенство чисел их вершин и равенство чисел ребер, т. е. графы не могут быть изоморфными, если хотя бы одно из этих равенств не выполняется. Тривиальным способом установления изоморфизма графов является следующий.

Пусть оба графа представлены матрицами смежности – это означает, что вершины графов пронумерованы в порядке следования строк и столбцов матриц. Изменение нумерации вершин графа выражается в перестановке строк и столбцов его матрицы смежности. Если зафиксировать матрицу смежности одного графа, а в другой последовательно менять порядок строк и, соответственно, столбцов, то в случае изоморфизма на каком-то шаге матрицы совпадут. Например, нетрудно заметить, что следующие матрицы смежности графов из рис. 4.1, где вершины правого графа упорядочены в порядке нумерации вершин левого графа, совпадают:

$$\begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix}, \quad \begin{array}{c} \begin{matrix} w_1 & w_3 & w_6 & w_4 & w_5 & w_2 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} w_1 \\ w_3 \\ w_6 \\ w_4 \\ w_5 \\ w_2 \end{matrix}.$$

Если такого совпадения не произойдет, то графы, вероятно, неизоморфны, но чтобы убедиться в этом, надо выполнить $n!$ перестановок, где n – число вершин графа. Ясно, что даже при сравнительно небольшом n эта задача окажется непосильной для современной вычислительной техники.

Рассмотрим некоторые приемы сокращения перебора при выявлении изоморфизма.

4.2. Инварианты перенумерации вершин графа

Пусть некоторая величина a есть функция, заданная на некотором множестве и сохраняющая свое значение при некотором преобразовании T этого множества. Говорят, что величина a *инвариантна* относительно преобразования T, если она не меняет свое значение при преобразовании T. Величина a называется *инвариантой* относительно T. В нашем случае таким преобразованием является перенумерация вершин.

В связи с установлением изоморфизма рассматриваются инварианты графа и инварианты вершин. Инвариантами графа являются, например, число вершин, число ребер, число компонент связности и т. п. Инвариантами вершины являются степень, полустепени, число вершин, отстоящих от данной вершины на определенном расстоянии.

Совокупность инвариант вершин служит инвариантой графа. В этом случае величиной a является множество, которое должно быть упорядоченным.

Канонизация графа заключается в упорядочении его вершин по значениям инвариант. Пусть для вершин графа имеется система инвариант $\alpha_1, \alpha_2, \ldots, \alpha_p$. Считаем, что задано отношение частичного порядка $\prec$ на множестве вершин графа $V = \{v_1, v_2, \ldots, v_n\}$, такое, что $v_i \prec v_j$, если $\alpha_k(v_i) < \alpha_k(v_j)$ для некоторого $k \in \{1, 2, \ldots, p)$ и $\alpha_l(v_i) = \alpha_l(v_j)$ для всех $l < k$.

Полная канонизация графа достигается, когда порядок $\prec$ оказывается полным и строгим. Матрицы смежности полностью канонизированных изоморфных графов должны совпадать. Даже если невозможно полностью канонизировать анализируемые на изоморфизм графы, но удается разбить множество вершин V канонизируемого графа на подмножества $S_1, S_2, \ldots, S_m$, характеризуемые совпадением систем инвариант входящих в них вершин, упомянутый выше перебор значительно сокращается, так как его следует вести

только внутри данных подмножеств. Действительно, если $|V| = n$ и $|S_i| = n_i$ ($i = 1, 2, \ldots, m$), то $n_1!n_2! \ldots n_m! << n!$.

Рассмотрим один из способов канонизации графа. Разобьем множество V вершин графа G на подмножества $S_1, S_2, \ldots, S_m$, число m которых равно числу различных степеней вершин и в каждом из которых присутствуют вершины с одинаковой степенью. Для каждой вершины $v_i \in V$ образуем вектор размерности m, компоненты которого соответствуют множествам $S_1, S_2, \ldots, S_m$ и значением j-й компоненты является число вершин из множества S_j, смежных с v_i. Очевидно, данный вектор является инвариантой вершины. Если в одном и том же S_k ($k = 1, 2, \ldots, m$) оказались вершины с различными векторами, то разобьем это S_k так, чтобы в каждом из получившихся множеств оставались вершины с одинаковыми векторами, соответственно увеличив размерность векторов и придав их компонентам новые значения. При этом поддерживаем порядок вершин, соответствующий лексикографическому порядку их векторов. Данное преобразование повторяем до тех пор, пока в любом из $S_1, S_2, \ldots, S_m$ не останутся только вершины с одинаковыми векторами.

Проиллюстрируем описанный процесс канонизации на примере графа, изображенного на рис. 4.2. В качестве начальной инварианты вершины возьмем ее степень, т. е. $\alpha(v) = d(v)$. Результаты выполнения данного процесса по шагам выглядят следующим образом:

		α
S_1	v_2	2
	v_6	2
	v_1	3
S_2	v_3	3
	v_4	3
	v_5	3

		α_1	α_2
S_1	v_2	0	2
	v_6	0	2
	v_1	1	2
S_2	v_3	2	1
	v_4	0	3
	v_5	1	2

		α_1	α_2	α_3	α_4
S_1	v_2	0	0	1	1
	v_6	0	0	1	1
S_2	v_4	0	0	2	1
S_3	v_1	1	1	1	0
	v_5	1	1	1	0
S_4	v_3	2	1	0	0

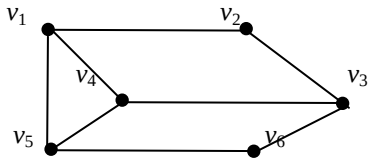


Рис. 4.2. Канонизируемый граф

При установлении изоморфизма между двумя графами следует параллельно канонизировать оба графа. Тогда, если встретится какое-нибудь несовпадение, надо прекращать процесс и выносить решение об отсутствии изоморфизма.

Г л а в а 5

Циклы и разрезы

5.1. Цикломатическое число графа

Для исследования циклов в графах необходимо ввести понятие *дерева*, которое является особым видом графа, часто используемым в различных приложениях. Приведем три эквивалентных определения:

1. Дерево – это связный граф, число ребер которого на единицу меньше числа вершин.

2. Дерево – это связный граф, не имеющий циклов.

3. Дерево – это граф, в котором каждая пара вершин связана одной и только одной цепью.

Граф, каждая компонента связности которого является деревом, называется *лесом*.

Пусть G – неориентированный граф с n вершинами, m ребрами и p компонентами связности. Если G – связный граф ($p = 1$), то он имеет остовный подграф в виде дерева. Такой подграф называется *остовным деревом*. Число ребер в остовном дереве $n - 1$. Число ребер в остовном лесе $n - p$.

Определим $\nu(G) = m - n + p$ и $\rho(G) = n - p$. Число $\nu(G)$ называется *цикломатическим числом*, а $\rho(G)$ – *коцикломатическим числом*.

В теории электрических цепей моделью электрической схемы может служить мультиграф, вершины которого соответствуют узлам схемы, а ребра – электрическим цепям между узлами. Тогда $\nu(G)$ представляет число независимых круговых токов, протекающих в схеме, а $\rho(G)$ – число независимых разностей потенциалов между узлами.

5.2. Базис циклов

Рассмотрим остовное дерево T некоторого связного графа $G = (V, E)$. Добавление одного ребра из множества E к T приводит к появлению точно одного простого цикла, состоящего из добавленного ребра и ребер дерева T, лежащих на единственной цепи, соединяющей в T концы данного ребра. Число получаемых таким образом циклов в графе G есть $m - n + 1$, что совпадает с цикломатическим числом $\nu(G)$ графа G.

Каждый из этих циклов имеет ребро, не принадлежащее никакому другому из них. В этом смысле они независимы. Множество циклов, определяемых таким образом с помощью остовного дерева, называется *базисом циклов* графа G, а сами циклы, принадлежащие базису, – *фундаментальными циклами*. Любой цикл, не принадлежащий базису, может быть выражен в виде линейной комбинации фундаментальных циклов. Это означает следующее.

Всякий цикл графа G представим m-мерным булевым вектором, в котором i-я компонента имеет значение 1 или 0 в зависимости от того, принадлежит или нет i-е ребро данному циклу. Тогда любой цикл можно выразить как покомпонентную сумму по модулю 2 векторов, представляющих фундаментальные циклы (сумма по модулю 2 определяется следующим образом: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$).

Для примера рассмотрим граф, изображенный на рис. 5.1. Выделенные ребра в нем образуют остовное дерево. Оно определяет три фундаментальных цикла – это цикл, проходящий через вершины v_1, v_2, v_3, v_5; цикл, проходящий через вершины v_2, v_3, v_5; и цикл, проходящий через вершины v_3, v_4, v_5. Возьмем два последних цикла. Их линейная комбинация представит цикл, проходящий через вершины v_2, v_3, v_4, v_5 и не являющийся фундаментальным:

$$
\begin{array}{l}
e_1 e_2\, e_3\, e_4\, e_5\, e_6\, e_7 \\
0\,0\;\; 1\;\; 1\;\; 0\;\; 1\;\; 0 \\
0\,0\;\; 0\;\; 0\;\; 1\;\; 1\;\; 1 \\
\hline
0\,0\;\; 1\;\; 1\;\; 1\;\; 0\;\; 1\,.
\end{array}
$$

Заметим, что базис циклов определяется неоднозначно и зависит от выбранного остовного дерева. Заметим также, что цикл, представляющий собой линейную комбинацию фундаментальных циклов, не обязательно простой. Например, линейная комбинация всех трех фундаментальных циклов графа на рис. 5.1 является циклом, который проходит два раза через вершину v_5.

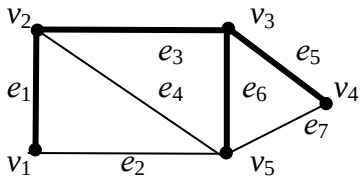


Рис. 5.1. Граф с выделенным остовным деревом

5.3. Базис разрезов

Разрезом графа называется множество его ребер, удаление которых увеличивает число компонент связности графа. Будем вести речь только о минимальных разрезах, т. е. о таких разрезах, каждый из которых перестает быть разрезом при удалении из него любого ребра, и под разрезом будем понимать именно минимальный разрез.

Разрез графа связан с его остовным деревом. Остовное дерево представляет минимальное множество ребер графа, связывающее все его вершины, а разрез – минимальное множество ребер, разделяющее вершины.

Очевидно, любое остовное дерево графа должно иметь хотя бы одно общее ребро с каждым разрезом.

Пусть в связном графе G, число вершин которого n, выделено остовное дерево T. Назовем *фундаментальным разрезом* каждый из $n-1$ разрезов, который содержит одно и только одно ребро, принадлежащее дереву T. Таким образом, каждое ребро e дерева T определяет фундаментальный разрез, который составляют, кроме ребра e, все ребра графа G, не принадлежащие дереву T, но входящие в фундаментальные циклы, содержащие e. Действительно, чтобы разделить вершины, связанные ребром e, надо удалить e и разорвать все цепи, связывающие данные вершины помимо ребра e. Каждая такая цепь вместе с ребром e образует фундаментальный цикл.

Множество фундаментальных разрезов графа G называется *базисом разрезов* графа G. Так же, как любой цикл представляется линейной комбинацией фундаментальных циклов, любой разрез графа G представляется линейной комбинацией его фундаментальных разрезов.

В графе на рис. 5.1 множества ребер $\{e_2, e_3, e_4\}$ и $\{e_2, e_4, e_6, e_7\}$ представляют собой фундаментальные разрезы, а разрез $\{e_3, e_6, e_7\}$ – их линейную комбинацию. В векторном представлении это выглядит следующим образом:

$$
\begin{array}{c}
e_1\ e_2\ e_3\ e_4\ e_5\ e_6\ e_7 \\
0\ 1\ 1\ 1\ 0\ 0\ 0 \\
0\ 1\ 0\ 1\ 0\ 1\ 1 \\
\hline
0\ 0\ 1\ 0\ 0\ 1\ 1\,.
\end{array}
$$

5.4. Матрицы циклов и разрезов

Пусть G – граф с n вершинами и m ребрами, T – остовное дерево графа G. *Матрицей фундаментальных циклов* графа G называется булева матрица, состоящая из $\nu(G)$ строк, соответствующих фундаментальным циклам, и m столбцов, соответствующих ребрам. На пересечении столбца и строки имеется 1, если соответствующее ребро принадлежит соответствующему циклу, и 0 – в противном случае. Таким образом, каждая строка матрицы циклов является векторным представлением фундаментального цикла, описанным выше. При этом удобно упорядочить ребра так, чтобы в начале получаемой последовательности находились ребра, не принадлежащие дереву T, и в том же порядке расположить определяемые ими циклы. Матрица фундаментальных циклов графа, представленного на рис. 5.1, примет тогда следующий вид:

$$\begin{matrix} e_2 & e_4 & e_7 & e_1 & e_3 & e_5 & e_6 \end{matrix}$$
$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Левая часть такой матрицы, состоящая из $\nu(G)$ столбцов, представляет собой единичную матрицу.

Матрица фундаментальных разрезов определяется аналогично. Это матрица с $n-1$ строками и m столбцами, где строки соответствуют фундаментальным разрезам, а столбцы – ребрам. На пересечении столбца и строки имеется 1, если соответствующее ребро принадлежит соответствующему разрезу, и 0 – в противном случае. Порядок ребер при этом точно такой же, как в случае матрицы циклов, а разрезы располагаются в том же порядке, что и определяющие их ребра. Матрица фундаментальных разрезов графа на рис. 5.1 имеет следующий вид:

$$\begin{matrix} e_2 & e_4 & e_7 & e_1 & e_3 & e_5 & e_6 \end{matrix}$$
$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Как видно из данного примера, правая часть матрицы фундаментальных разрезов, состоящая из $\rho(G)$ столбцов, является единичной матрицей, а левая часть совпадает с транспонированной правой частью матрицы фундаментальных разрезов. Таким образом, нахождение базиса циклов и нахождение базиса разрезов являются двойственными по отношению друг к другу задачами. Любая из представляющих эти базисы матриц получается из другой путем транспонирования.

Интересно заметить, что если приписать строки матрицы фундаментальных разрезов к матрице фундаментальных циклов, то получим квадратную матрицу, симметричную относительно главной диагонали. Эту матрицу можно разделить на такие четыре части, что левая верхняя и правая нижняя части представляют собой единичные матрицы (с единицами на главной диагонали и только на ней), а каждая из остальных двух частей является транспонированным вариантом другой:

$$
\begin{array}{c}
e_2\ e_4\ e_7 \quad e_1\ e_3\ e_5\ e_6 \\
\left[\begin{array}{ccc|cccc}
1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 \\
- & - & - & - & - & - & - \\
1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1
\end{array}\right]
\end{array}.
$$

В приведенной матрице прерывистыми линиями выделены упомянутые части.

Г л а в а 6

Доминирующие и независимые множества

6.1. Доминирующие множества графа

Подмножество S множества вершин V графа G называется *доминирующим множеством* графа G, если выполняется условие $S \cup N(S) = V$, где $N(S) = \bigcup_{v \in S} N(v)$, а $N(v)$ – множество вершин, смежных с вершиной v. Другими словами, множество S является доминирующим, если каждая вершина из множества $V \setminus S$ смежна с некоторой вершиной из S. Если S является доминирующим множеством некоторого графа G, то всякое множество вершин $S' \supseteq S$ этого графа также является доминирующим. Поэтому представляет интерес задача нахождения *минимальных* доминирующих множеств, т. е. таких, у которых ни одно собственное подмножество не является доминирующим. Доминирующее множество, имеющее наименьшую мощность, принято называть *наименьшим*. Эта мощность называется *числом доминирования* графа G и обозначается символом $\beta(G)$.

Наглядным примером задачи о наименьшем доминирующем множестве является одна из задач о ферзях, где надо расставить на шахматной доске наименьшее число ферзей так, чтобы каждая клетка была под ударом хотя бы одного из них. Для этого достаточно найти наименьшее доминирующее множество в графе, вершины которого соответствуют клеткам шахматной доски и две вершины связаны ребром, если и только если они взаимно достижимы ходом ферзя. Найденное множество вершин указывает, куда надо поставить ферзей, а их число, которое в данном случае равно пяти, есть число доминирования данного графа.

Задача о наименьшем доминирующем множестве сводится к известной задаче о *кратчайшем покрытии*, которая подробно будет рассмотрена ниже. В данном случае ее можно поставить следующим образом. Матрицу смежности заданного графа G дополним единицами на главной диагонали. Тогда требуется найти минимальную совокупность строк полученной матрицы, такую, что каждый столбец имел бы единицу по крайней мере в одной из строк найденной совокупности. Говорят, что строка матрицы *покрывает* столбец, если данный столбец имеет единицу в этой строке.

Нетрудно видеть, что наименьшими доминирующими множествами графа G (рис. 6.1) являются $\{v_3, v_7\}$, $\{v_5, v_7\}$ и $\{v_6, v_7\}$. Его матрица смежности, дополненная единицами на главной диагонали, имеет вид

$$
\begin{array}{c}
\begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 1
\end{bmatrix}
\begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{matrix}
\end{array}
$$

Каждое из соответствующих множеств строк рассматриваемой матрицы покрывает все ее столбцы.

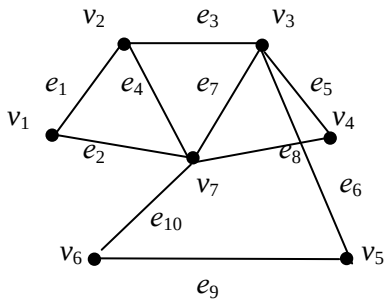


Рис. 6.1. Граф G

6.2. Независимые множества графа

Подмножество S множества вершин V графа G называется *независимым множеством* графа G, если выполняется условие $S \cap N(S) = \varnothing$, т. е. любые две вершины из S не смежны. Если S – независимое множество, то любое его подмножество также является независимым. Поэтому представляет интерес задача нахождения в графе *максимальных независимых множеств*, т. е. таких, которые не являются собственными подмножествами никаких других независимых множеств.

Независимое множество, имеющее наибольшую мощность среди всех независимых множеств графа G, называют *наибольшим независимым множеством*, а его мощность называют *числом независимости* графа G и обозначают символом $\alpha(G)$.

Примером задачи нахождения наибольшего независимого множества является другая задача о ферзях, в которой надо расставить на шахматной доске наибольшее число ферзей так, чтобы ни один из них не находился под ударом другого. Наибольшее независимое множество графа, представляющего шахматную доску, как определено выше, покажет, на какие клетки надо поставить ферзей. Наибольшее число ферзей, расставленных при указанном

условии, которое в данном случае равно восьми, есть число независимости данного графа.

Рассмотрим один из способов нахождения в графе всех максимальных независимых множеств.

Пусть G – заданный граф с произвольно упорядоченным множеством вершин $V = \{v_1, v_2, \dots, v_n\}$. Рассмотрим последовательность подграфов $G_1, G_2, \dots, G_n$, порожденных подмножествами $V_1, V_2, \dots, V_n$, где $V_i = \{v_1, v_2, \dots, v_i\}$ $(i = 1, 2, \dots, n)$. Пусть $S^i = \{S_1^i, S_2^i, \dots, S_{k_i}^i\}$ – совокупность всех максимальных независимых множеств графа G_i. Преобразуем S^i следующим образом. Для каждого S_j^i $(j = 1, 2, \dots, k_i)$ получим множество

$$S' = (S_j^i \setminus N(v_{i+1})) \cup \{v_{i+1}\}.$$

Если в S^i найдется такой элемент S_l^i, что $S_l^i \subset S'$, то S_l^i в S^i заменяем на S'. Если найдется такой элемент S_l^i в множестве S^i, что $S' \subseteq S_l^i$, то S^i не изменяем. В остальных случаях S' добавляем в множество S^i в качестве нового элемента.

Нетрудно убедиться, что в результате таких преобразований множество S^i превращается в S^{i+1} – совокупность всех максимальных независимых множеств графа G_{i+1}. Действительно, все $S_1^i, S_2^i, \dots, S_{k_i}^i$ являются независимыми множествами для G_{i+1}. Множество S' является также независимым. Все поглощаемые множества удаляются, так что остаются только максимальные. То, что S^{i+1} содержит все максимальные независимые множества графа G_{i+1}, легко доказывается от противного. Пусть S'' – максимальное независимое множество графа G_{i+1}, не полученное в результате описанных преобразований. Но тогда $S'' \setminus \{v_{i+1}\}$ является максимальным независимым множеством графа G_i, которого нет в S^i, что противоречит определению множества S^i.

Начиная от $S^1 = \{\{v_1\}\}$, выполним цепочку таких преобразований, в результате чего получим $S^n = S$ – совокупность всех максимальных независимых множеств графа $G_n = G$.

Для графа, изображенного на рис. 6.1, имеем следующую последовательность результатов применения описанного преобразования, где S^7 представляет совокупность всех максимальных независимых множеств:

$S^2 = \{\{v_1\}, \{v_2\}\}$;
$S^3 = \{\{v_1, v_3\}, \{v_2\}\}$;
$S^4 = \{\{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_4\}\}$;
$S^5 = \{\{v_1, v_3\}, \{v_1, v_4, v_5\}, \{v_2, v_4, v_5\}\}$;
$S^6 = \{\{v_1, v_3, v_6\}, \{v_1, v_4, v_5\}, \{v_1, v_4, v_6\}, \{v_2, v_4, v_5\}, \{v_2, v_4, v_6\}\}$;
$S^7 = \{\{v_1, v_3, v_6\}, \{v_1, v_4, v_5\}, \{v_1, v_4, v_6\}, \{v_2, v_4, v_5\}, \{v_2, v_4, v_6\}, \{v_5, v_7\}\}$.

Иногда требуется найти одно, но наибольшее независимое множество. Можно для этого получить все максимальные независимые множества и из них выбрать наибольшее. Тогда интересно знать верхнюю границу числа максимальных независимых множеств в графе с n вершинами. Известно, что она выражается следующими формулами, где k – некоторое положительное целое число:

$2 \cdot 3^{k-1}$, если $n = 3k - 1$;
$3 \cdot 3^{k-1}$, если $n = 3k$;
$4 \cdot 3^{k-1}$, если $n = 3k + 1$.

Экстремальным графом, т. е. графом, в котором достигается указанная граница, для случая $n = 3k$ является показанный на рис. 6.2 несвязный граф, состоящий из k компонент. Для случая $n = 3k - 1$ или $n = 3k + 1$ один из треугольников заменяется соответственно на изолированное ребро или полный четырехвершинный граф.

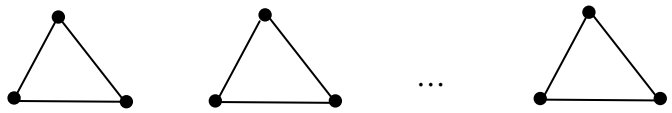


Рис. 6.2. Экстремальный граф для максимальных независимых множеств

Из приведенных формул видно, что верхняя граница числа максимальных независимых множеств растет во много раз быстрее, чем число вершин графа. Поэтому представляет интерес способ получения максимальных независимых множеств, не требующий их одновременного представления. Рассмотрим такой способ, получающий эти множества в лексикографическом порядке.

Пусть $V = \{v_1, v_2, \ldots, v_n\}$ – множество вершин графа G. Весь процесс нахождения максимальных независимых множеств можно разбить на n этапов, каждый из которых связан с определенной вершиной $v_i \in V$. На i-м этапе находятся все максимальные независимые множества, содержащие вершину v_i и не содержащие вершин с меньшими номерами, т. е. таких v_j, для которых $j < i$.

Пусть S_i – одно из независимых множеств графа G, формируемых на i-м этапе. За начальное значение множества S_i принимается множество, состоящее из единственной вершины v_i. Множество S_i расширяется за счет поочередного включения в него элементов $v_j \in V$, удовлетворяющих следующим условиям:

$$i < j \le n \quad \text{и} \quad v_j \in \bigcap_{v \in S_i} \{V \setminus N(v)\}.$$

Каждый раз при соблюдении этих условий выбирается v_j с минимальным j. Это расширение множества S_i продолжается до тех пор, пока множество $\bigcap_{v \in S_i} \{V \setminus N(v)\}$ не станет пустым.

Результат расширения S_i проверяется на максимальность согласно следующему свойству: независимое множество S является максимальным в том и только в том случае, когда $S \cup N(S) = V$.

Действительно, если это равенство не выполняется, то найдется вершина в множестве $V \setminus S$, не смежная ни с одной вершиной из S. Присоединив ее к S, получим независимое множество, содержащее S в качестве собственного подмножества, т. е. S не максимально. Если же это равенство выполняется, то не существует вершины в графе G, которая была бы не смежна ни с одной вершиной из S и ее можно было бы добавить к S, не нарушая независимости S.

Множество, прошедшее такую проверку, включается в решение. Чтобы построить следующее по порядку независимое множество, из полученного S_i (включенного или не включенного в решение) удаляется вершина v_p, присоединенная к S_i последней, и выполняется та же процедура с вершинами v_q, где $q > p$.

За n этапов можно получить все максимальные независимые множества заданного графа. Однако данный процесс можно прекратить на k-м этапе, где $k < n$, если видно, что из оставшихся $n - k$ вершин не получится максимального независимого множества. Для этого надо для каждой вершины v_i сформировать множества $A_i = \{v_i, v_{i+1}, \ldots, v_n\}$ и $B_i = \bigcup_{j=i}^{n} N(v_j)$ и на очередном i-м этапе проверить условие $A_i \cup B_i = V$. Если оно не выполняется, то данный процесс надо прекратить.

Для графа на рис. 6.1 множества S_i ($i = 1, 2, 3, 4, 5, 6$) будут принимать следующие значения (этап, связанный с вершиной v_7, не выполняется, так как $A_7 \cup B_7 = \{v_7\} \cup \{v_1, v_2, v_3, v_4, v_6\} \neq V$):

S_1: $\{v_1\}$, $\{v_1, v_3\}$, $\underline{\{v_1, v_3, v_6\}}$, $\{v_1, v_4\}$, $\underline{\{v_1, v_4, v_5\}}$, $\underline{\{v_1, v_4, v_6\}}$, $\{v_1, v_5\}$, $\{v_1, v_6\}$;
S_2: $\{v_2\}$, $\{v_2, v_4\}$, $\underline{\{v_2, v_4, v_5\}}$, $\underline{\{v_2, v_4, v_6\}}$, $\{v_2, v_5\}$, $\{v_2, v_6\}$;
S_3: $\{v_3\}$, $\{v_3, v_6\}$;
S_4: $\{v_4\}$, $\{v_4, v_5\}$, $\{v_4, v_6\}$;
S_5: $\{v_5\}$, $\underline{\{v_5, v_7\}}$;
S_6: $\{v_6\}$.

Здесь подчеркнуты те значения S_i, которые вошли в решение.

Для получения одного наибольшего независимого множества рассмотренный способ удобнее, чем предыдущий. Хранить все максимальные независимые множества одновременно не надо. Они получаются последовательно, и каждый раз надо оставлять только то из них, которое обладает наибольшей мощностью. Как показано выше, число их может

оказаться слишком большим, и поэтому менее трудоемким представляется следующий способ нахождения одного наибольшего независимого множества, не требующий получения всех максимальных независимых множеств.

Вершинным покрытием графа $G = (V, E)$ называется такое множество $B \subseteq V$, что каждое ребро из E инцидентно хотя бы одной вершине из B. Очевидно, что если B – вершинное покрытие, то $V \setminus B$ – независимое множество. *Вершинное покрытие* наименьшей мощности в графе G является его *наименьшим вершинным покрытием*. Ясно, что если B – наименьшее вершинное покрытие, то $V \setminus B$ – наибольшее независимое множество.

Чтобы найти наименьшее вершинное покрытие в графе G, достаточно покрыть столбцы его матрицы инцидентности минимальным количеством ее строк. Матрица инцидентности графа на рис. 6.1 имеет следующий вид:

$$\begin{array}{cccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & \\ \left[\begin{array}{cccccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right] & \begin{array}{c} v_1 \\ v_1 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \end{array}.$$

Строки v_1, v_3, v_5 и v_7 составляют кратчайшее покрытие этой матрицы. Множество вершин $\{v_1, v_3, v_5, v_7\}$ является наименьшим вершинным покрытием данного графа. Следовательно, $\{v_2, v_4, v_6\}$ – одно из наибольших независимых множеств. Оно присутствует в решении предыдущего примера.

Нахождение наибольшего независимого множества – довольно трудоемкая задача. Поэтому иногда довольствуются получением максимального независимого множества, по мощности близкого к наибольшему. Одним из способов решения такой задачи является чередование следующих двух операций: выбора вершины с наименьшей степенью в качестве элемента искомого множества и удаления выбранной вершины из графа вместе с окрестностью. В результате может получиться наибольшее независимое множество, однако, как видно на примере графа, приведенного на рис. 6.3, это бывает не всегда.

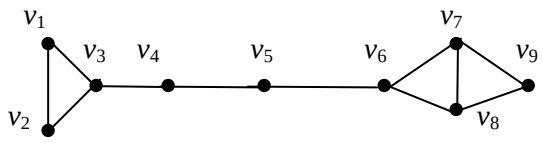


Рис. 6.3. Граф для демонстрации получения независимого множества

Действительно, в качестве первой вершины для включения в решение может быть выбрана вершина v_5. Тогда вслед за ней в решение включаются вершины v_3 и v_7. Мощность полученного таким образом множества на единицу меньше мощности наибольшего независимого множества $\{v_1, v_4, v_6, v_9\}$.

Подграф графа G, порождаемый его независимым множеством, является *пустым подграфом*, т. е. множество его ребер пусто. В графе $\bar{G}$, который является дополнением графа G, это же множество порождает *полный подграф*, т. е. подграф графа $\bar{G}$, в котором все вершины попарно смежны в графе $\bar{G}$. Полный подграф называют еще *кликой*. Таким образом, задача нахождения полных подграфов, или клик, и задача нахождения независимых множеств в некотором графе являются двойственными по отношению друг к другу.

Г л а в а 7

Раскраска графа

7.1. Постановка задачи

Раскраской некоторого графа $G = (V, E)$ называется такое разбиение множества вершин V на непересекающиеся подмножества $V_1, V_2, \ldots, V_k$, что никакие две вершины из одного, любого, из этих подмножеств не смежны. Считается, что вершины, принадлежащие одному и тому же подмножеству V_i, выкрашены при этом в один и тот же цвет i. Задача состоит в том, чтобы раскрасить вершины графа G в минимальное число цветов. Оно называется *хроматическим числом* графа и обозначается $\gamma(G)$.

Задача раскраски графа имеет много приложений в различных областях человеческой деятельности. К задаче раскраски сводятся составление расписания занятий в учебном заведении, распределение оборудования на предприятии, выбор расцветки проводов в сложных электрических схемах и многие другие практические задачи. Например, при минимизации числа ячеек памяти вычислительной машины для хранения промежуточных результатов в процессе выполнения программы можно построить граф, вершины которого соответствуют промежуточным результатам вычисления и две вершины связаны ребром, если и только если соответствующие величины используются одновременно и не могут храниться в одной и той же ячейке. Хроматическое число данного графа представляет минимум упомянутых ячеек.

Иногда ставится задача раскраски ребер графа $G = (V, E)$, где требуется получить такое разбиение множества ребер E на непересекающиеся подмножества $E_1, E_2, \ldots, E_p$, что ни одна пара ребер из одного и того же E_i ($i = 1, 2, \ldots, p$) не имеет общей инцидентной вершины. Данная задача сводится к раскраске вершин. Для этого надо построить *реберный граф* $L(G)$ графа G. Вершины графа $L(G)$ соответствуют ребрам графа G, и две вершины графа $L(G)$ связаны ребром, если и только если соответствующие ребра графа G имеют общую инцидентную вершину в G. Раскраска ребер графа G соответствует раскраске вершин графа $L(G)$.

7.2. Метод раскраски графа

Очевидно, всякое множество одноцветных вершин графа является независимым множеством. Поэтому получить минимальную раскраску можно следующим образом: найти все максимальные независимые множества; получить кратчайшее покрытие множества вершин графа максимальными независимыми множествами; удалить некоторые вершины из элементов полученного покрытия, добившись того, чтобы каждая вершина входила в одно и только в одно из выделенных независимых множеств. Для графов, число

независимых множеств которых невелико, этот способ является приемлемым. Однако, как показывает оценка, приведенная в предыдущей главе, это число для некоторых графов может оказаться настолько большим, что данный способ вообще не сможет быть реализован. Существует немало методов раскраски, не использующих задачу покрытия и получающих точно минимальное число цветов, но и их применение существенно ограничено размерностью задачи.

Рассмотрим один из методов раскраски графа, который не гарантирует получения минимума цветов, но дает раскраску, близкую к минимальной, а во многих случаях совпадающую с ней.

Процесс раскраски графа $G = (V, E)$ представляет собой последовательность шагов, на каждом из которых выбирается вершина и окрашивается в определенный цвет. Текущая ситуация характеризуется следующими объектами: k – число задействованных цветов; A – множество еще не раскрашенных вершин; $B_1, B_2, \dots, B_k$ – совокупность подмножеств множества вершин V, такая, что B_i ($i = 1, 2, \dots, k$) содержит те и только те вершины из множества A, которые нельзя раскрасить в i-й цвет. Обратим внимание на следующие два случая:

1. Имеется вершина $v \in A$, такая, что $v \in B_i$ для всех $i = 1, 2, \dots, k$.

2. Имеется вершина $v \in A$ и цвет i, такие, что $v \notin B_i$ и $N(v) \cap A \subseteq B_i$.

В первом случае вершину v надо красить в $(k + 1)$-й цвет, удалить ее из множества A и из всех множеств B_i, где она была, сформировать множество B_{k+1} и увеличить k на единицу. Если таких вершин несколько, из них выбирается та, для которой множество B_{k+1} имеет максимальную мощность.

Во втором случае вершину v надо красить в i-й цвет, удалить ее из множества A и из всех множеств B_j, где она была.

Во всех остальных случаях из множества A выбираются вершина v и цвет i такие, что $v \notin B_i$ и приращение $\Delta|B_i|$ мощности множества B_i минимально среди всех пар v, B_i ($v \in A$, $i = 1, 2, \dots, k$). Вершина v удаляется из A и из всех B_j, где она была, и красится в i-й цвет.

Выполнение описанных действий повторяется до тех пор, пока множество A не станет пустым.

В начальной ситуации $A = V$, $k = 0$ и рекомендуется выбирать вершину с максимальной степенью и красить ее в цвет 1, а множество B_1 будет представлять ее окрестность.

Продемонстрируем применение данного метода на примере графа, изображенного на рис. 7.1.

Получим следующие результаты выполнения последовательности шагов.

Шаг 1: $\{v_1\}$; $B_1 = \{v_2, v_6, v_8, v_{10}\}$; $A = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$.

Шаг 2 (1-й случай: $v_2 \in B_1$): $\{v_1\}$, $\{v_2\}$; $B_1 = \{v_6, v_8, v_{10}\}$, $B_2 = \{v_4, v_5\}$; $A = \{v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$.

Шаг 3 (2-й случай: $N(v_8) = \varnothing$): $\{v_1\}$, $\{v_2, v_8\}$; $B_1 = \{v_6, v_{10}\}$, $B_2 = \{v_4, v_5\}$; $A = \{v_3, v_4, v_5, v_6, v_7, v_9, v_{10}\}$.

Шаг 4 (выбор v_7, цвет 1, $\Delta|B_1| = 1$): $\{v_1, v_7\}$, $\{v_2, v_8\}$; $B_1 = \{v_3, v_6, v_{10}\}$, $B_2 = \{v_4, v_5\}$; $A = \{v_3, v_4, v_5, v_6, v_9, v_{10}\}$.

Шаг 5 (выбор v_3, цвет 2, $\Delta|B_2| = 1$): $\{v_1, v_7\}$, $\{v_2, v_3, v_8\}$; $B_1 = \{v_6, v_{10}\}$, $B_2 = \{v_4, v_5, v_6\}$; $A = \{v_4, v_5, v_6, v_9, v_{10}\}$.

Шаг 6 (1-й случай: $v_6 \in B_i$, $i = 1, 2$): $\{v_1, v_7\}$, $\{v_2, v_3, v_8\}$, $\{v_6\}$; $B_1 = \{v_{10}\}$, $B_2 = \{v_4, v_5\}$, $B_3 = \{v_9\}$; $A = \{v_4, v_5, v_9, v_{10}\}$.

Шаг 7 (выбор v_5, цвет 3, $\Delta|B_3| = 1$): $\{v_1, v_7\}$, $\{v_2, v_3, v_8\}$, $\{v_5, v_6\}$; $B_1 = \{v_{10}\}$, $B_2 = \{v_4\}$, $B_3 = \{v_4, v_9\}$; $A = \{v_4, v_9, v_{10}\}$.

Шаг 8 (2-й случай: $N(v_{10}) \cap A \subseteq B_3$): $\{v_1, v_7\}$, $\{v_2, v_3, v_8\}$, $\{v_5, v_6, v_{10}\}$; $B_1 = \varnothing$, $B_2 = \{v_4\}$, $B_3 = \{v_4, v_9\}$; $A = \{v_4, v_9\}$.

Шаг 9 (выбор v_4, цвет 1, $\Delta|B_1| = 1$): $\{v_1, v_4, v_7\}$, $\{v_2, v_3, v_8\}$, $\{v_5, v_6, v_{10}\}$; $B_1 = \{v_9\}$, $B_2 = \varnothing$, $B_3 = \{v_9\}$; $A = \{v_9\}$.

Шаг 10: завершение работы с результатом $\{v_1, v_4, v_7\}$, $\{v_2, v_3, v_8, v_9\}$, $\{v_5, v_6, v_{10}\}$.

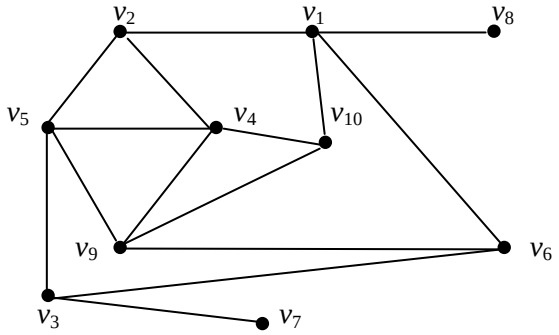


Рис. 7.1. Раскрашиваемый граф

Полученная раскраска для данного графа является минимальной, так как хроматическое число графа $\chi(G)$ не может быть меньше числа вершин его наибольшей клики, а на рис. 7.1 видны клики, состоящие из трех вершин.

Описанный способ дает возможность иногда делать заключение о том, что полученная раскраска является минимальной. Назовем шаги, выполненные в первом и втором случаях, «хорошими», а в остальных случаях – «сомнительными». Если в процессе раскраски выполнялись только хорошие шаги, то, очевидно, полученная раскраска является минимальной. Если приходилось выполнять сомнительные шаги, то полученная раскраска может оказаться не минимальной, но по соотношению между количествами хороших и сомнительных шагов можно судить о близости полученной раскраски к минимальной.

Иногда можно получить раскраску графа, минимальную или близкую к минимальной, с помощью так называемого «*жадного*» алгоритма, где на каждом шаге в текущий цвет раскрашивается как можно больше вершин. Желательно для этого брать наибольшее независимое множество. Раскрашенные вершины удаляются из графа, вводится новый цвет, в него раскрашивается опять как можно больше вершин и так далее до тех пор, пока

множество вершин графа не станет пустым. Однако есть пример графа, для которого число цветов, полученное при такой раскраске, может отличаться от минимального на сколь угодно большую величину [13].

Рассмотрим неограниченную последовательность деревьев $T_1, T_2, \ldots, T_i, \ldots$, начало которой изображено на рис. 7.2. Дерево T_1 состоит из трех вершин и двух ребер. Дерево T_i получается из T_{i-1} присоединением к каждой вершине T_{i-1} двух смежных с ней вершин. Наибольшее независимое множество дерева T_i составляют все вершины, не принадлежащие T_{i-1}. Число цветов, получаемое при раскраске дерева T_i данным способом, равно $i + 1$, хотя ясно, что всякое дерево можно раскрасить в два цвета.

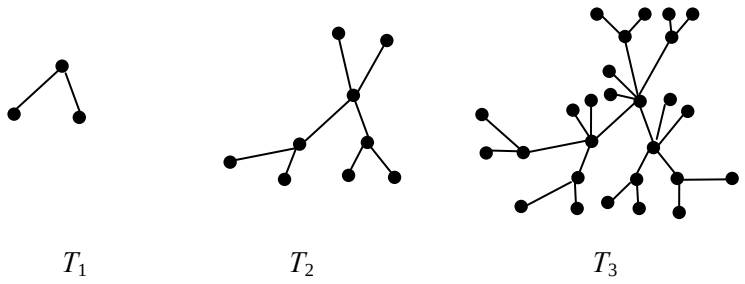


Рис. 7.2. Деревья, раскрашиваемые «жадным» алгоритмом

7.3. Бихроматические графы

Граф G называется *k-хроматическим*, если $\chi(G) = k$. Очевидно, пустые и только пустые графы являются 1-хроматическими. Особый класс составляют *бихроматические графы*, т. е. такие, у которых $\chi(G) = 2$.

Т е о р е м а К ё н и г а. Непустой граф является бихроматическим тогда и только тогда, когда он не содержит циклов нечетной длины.

Допустим, что $G = (V, E)$ – связный бихроматический граф. Это не нарушает общности, так как в случае несвязного графа последующие рассуждения можно провести для каждой его компоненты в отдельности. Пусть V^1 и V^2 – множества вершин графа G, раскрашенных соответственно в цвета 1 и 2. Всякое ребро соединяет вершину из V^1 с вершиной из V^2. Следовательно, всякая цепь, начинающаяся и оканчивающаяся в одном и том же множестве V^i ($i = 1, 2$), имеет четную длину. Пусть теперь G – связный граф, не имеющий циклов нечетной длины. Возьмем любую вершину v из графа G. Сформируем множество V^1 из вершин, отстоящих от v на четном расстоянии в графе G, и множество V^2 из всех остальных вершин. Ни одна пара вершин из V^2 не связана ребром. Действительно, если имелось бы ребро $v_iv_j \in E$, у которого $v_i \in V^2$ и $v_j \in V^2$, то цикл, составленный из цепи, соединяющей v с v_i, ребра v_iv_j и цепи, соединяющей v_j с v, имел бы нечетную длину, что противоречит условию.

Очевидно, всякий двудольный граф, имеющий хотя бы одно ребро, является бихроматическим, так как любая доля двудольного графа представляет собой независимое множество.

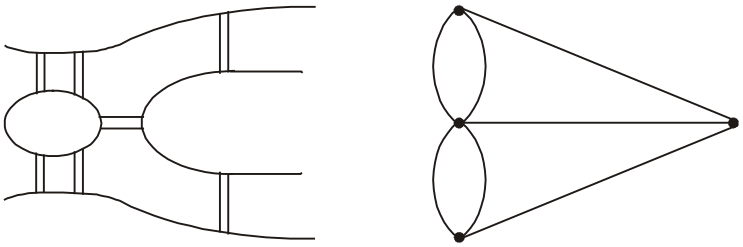
Бихроматичность графа легко установить, используя способ последовательной раскраски. Для этого произвольно выбирается вершина v и красится в цвет 1. Вершины ее окрестности $N(v)$ красятся в цвет 2, неокрашенные вершины из окрестностей вершин, принадлежащих $N(v)$, красятся в цвет 1 и т. д. В результате либо граф раскрашивается в два цвета, либо на каком-то шаге смежные вершины оказываются окрашенными в один и тот же цвет. Это говорит о том, что граф не является бихроматическим.

Г л а в а 8

Обходы графа

8.1. Эйлеровы цепи и циклы

Началом теории графов считается работа Л. Эйлера, опубликованная в 1736 г., в которой он решил *задачу о кёнигсбергских мостах*. На протекающей через город Кёнигсберг реке Прегель расположены два острова, которые были соединены семью мостами между собой и с берегами реки (рис. 8.1, *а*). Эйлер задался целью определить, можно ли, гуляя по городу, обойти все семь мостов, пройдя каждый из них ровно один раз. Для этого он сформулировал следующую задачу: в заданном связном графе (или мультиграфе) выделить цикл, содержащий все ребра данного графа. В исходной задаче острова и берега соответствуют вершинам графа, а проходы по мостам представлены ребрами. Соответствующий мультиграф изображен на рис. 8.1, *б*. Эйлер показал, что в данном случае задача решения не имеет.



а) *б)*

Рис. 8.1. Задача Эйлера: *а)* план кёнигсбергских мостов;
б) соответствующий мультиграф

Цикл, содержащий все ребра графа, носит название *эйлерова цикла*. Цепь, содержащая все ребра графа, называется *эйлеровой цепью*.

Т е о р е м а Э й л е р а. Связный неориентированный граф имеет эйлеров цикл тогда и только тогда, когда степени всех его вершин четны. В связном неориентированном графе существует эйлерова цепь тогда и только тогда, когда он имеет не более двух вершин с нечетной степенью.

Граф, имеющий эйлеров цикл, называется *эйлеровым графом*.

Эйлеров цикл можно найти в эйлеровом графе *алгоритмом Флёри*, который заключается в построении последовательности ребер, определяющей искомый цикл. При этом должны выполняться следующие правила.

1. Идем из некоторой вершины по ребру и удаляем каждое пройденное ребро, помещая его в получаемую последовательность. Начальная вершина выбирается произвольно.

2. Отправляясь из очередной вершины, никогда не идем по ребру, удаление которого делает граф несвязным.

Если граф имеет две вершины с нечетной степенью, то в нем можно найти эйлерову цепь, начиная поиск с одной из этих вершин, и заканчивая другой. При этом следует выполнять те же правила, что и при поиске эйлерова цикла.

Обобщением задачи Эйлера является известная *задача китайского почтальона*, которая имеет приложение, например, при проверке электрических сетей и ставится следующим образом. Каждому ребру e_i графа G приписывается положительный вес $c(e_i)$ (расстояние). Требуется найти маршрут, проходящий через каждое ребро графа G по крайней мере один раз и такой, что сумма величин $n_i c(e_i)$, где n_i – число прохождений ребра e_i, минимальна. Если граф является эйлеровым, то любой такой маршрут представляет собой эйлеров цикл, а данная сумма одинакова для всех эйлеровых циклов и является суммой весов всех ребер.

8.2. Гамильтоновы цепи и циклы

Цикл называется *гамильтоновым*, если он проходит каждую вершину графа ровно один раз. Другими словами, это остовный цикл графа. *Гамильтоновой цепью* называется цепь, проходящая каждую вершину графа ровно один раз. Граф, содержащий гамильтонов цикл, называется *гамильтоновым графом*.

На первый взгляд задача нахождения гамильтонова цикла в графе сходна с задачей нахождения эйлерова цикла. На самом же деле эти задачи принципиально различны. Для распознавания гамильтоновости графа не существует такого простого способа, как для распознавания эйлеровости графа. Поиск гамильтонова цикла или гамильтоновой цепи в графе значительно более трудоемок, чем поиск эйлерова цикла или эйлеровой цепи.

Рассмотрим один из способов построения гамильтонова цикла в графе.

Пусть вершины заданного графа $G = (V, E)$ пронумерованы в произвольном порядке: $v_1, v_2, \dots, v_n$. Для каждой вершины v_i сформируем список смежных с ней вершин, расположив вершины в списке в произвольном порядке. Будем представлять цикл в виде последовательности вершин C. В качестве отправной возьмем первую в порядке нумерации вершину v_1 и объявим ее первым элементом получаемой последовательности C. К вершине v_1 припишем вершину v_j, первую в списке v_1, в результате чего получим $C = (v_1, v_j)$. Обратимся к списку v_j, выберем из него первую вершину v_k, не присутствующую в последовательности C, и, присоединив ее к C, получим $C = (v_1, v_j, v_k)$. Затем обратимся к списку v_k, проделаем то же самое и т. д. Пусть получена последовательность $C = (v_1, v_j, v_k, \dots, v_q, v_r)$, не содержащая всех вершин из V, но в списке v_r нет вершин, не содержащихся в C. Тогда делаем шаг назад и в списке v_q выбираем вместо вершины v_r следующую по порядку вершину. Если и там такой вершины нет, делаем еще шаг назад и обращаемся к списку предшествующей вершины и т. д. В результате либо получаем искомую

последовательность, когда все вершины из V вошли в C, либо, возвращаясь к списку v_1, обнаруживаем, что он оказывается исчерпанным. В последнем случае граф не имеет гамильтонова цикла.

Продемонстрируем описанный процесс на примере графа, изображенного на рис. 8.2. Данный граф можно задать перечислением списков окрестностей вершин:

v_1: v_2, v_4, v_5, v_6;
v_2: v_1, v_3, v_4, v_5;
v_3: v_2, v_4, v_5;
v_4: v_1, v_2, v_3, v_6;
v_5: v_1, v_2, v_3;
v_6: v_1, v_4.

Сначала получаем последовательность $C = (v_1, v_2, v_3, v_4, v_6)$ и в списке v_6 нет вершин, не присутствующих в данной последовательности. Шаг назад приводит к последовательности $C = (v_1, v_2, v_3, v_4)$. В списке вершины v_4 также нет вершин, не присутствующих в последовательности C. Возвращаемся к $C = (v_1, v_2, v_3)$ и получаем последовательность $C = (v_1, v_2, v_3, v_5)$. Затем получаем последовательности с тупиковыми вершинами $C = (v_1, v_2, v_4, v_3, v_5)$, $C = (v_1, v_2, v_4, v_6)$ и, наконец, последовательность $C = (v_1, v_2, v_5, v_3, v_4, v_6, v_1)$, которая представляет искомый гамильтонов цикл. На рис. 8.2 выделены ребра, принадлежащие полученному гамильтонову циклу. Таким же способом можно построить гамильтонову цепь.

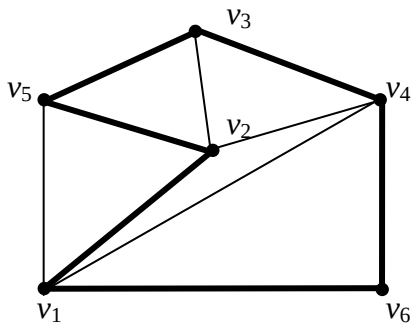


Рис. 8.2. Граф с гамильтоновым циклом

Обобщением задачи поиска гамильтонова цикла является известная *задача коммивояжера*, которая состоит в том, что в заданном графе с взвешенными ребрами надо найти маршрут, проходящий через каждую вершину не менее одного раза и обладающий минимальной суммой весов ребер среди всех таких маршрутов.

8.3. Кратчайшие пути в графе

Задан связный граф $G = (V, E)$ с ребрами, взвешенными действительными положительными числами. В данном случае вес ребра $e = v_i v_j$ будем считать его длиной $l(e) = l(v_i v_j)$. Требуется найти цепь c минимальной длины, соединяющую две заданные вершины в графе G, т. е. такую цепь, для которой величина $\sum_{e \in c} l(e)$ минимальна.

Для решения этой задачи можно применить *алгоритм Форда*, который заключается в следующем.

Пусть в графе G надо найти путь от вершины v_1 к вершине v_n. Каждой вершине $v_i \in V$ припишем индекс $\lambda(v_i)$. При этом положим $\lambda(v_1) = 0$ и $\lambda(v_i) = +\infty$ для $i \neq 1$.

На каждом шаге алгоритма отыскивается ребро $v_i v_j$, для которого $\lambda(v_i) - \lambda(v_j) > l(v_i v_j)$, и индекс $\lambda(v_i)$ заменяется на $\lambda'(v_i) = \lambda(v_j) + l(v_i v_j)$. Повторение таких шагов продолжается, пока находятся ребра, для которых выполняется данное неравенство.

В результате выполнения данной процедуры определяется длина кратчайшего пути, равная $\lambda(v_n)$. Сам путь надо строить, начиная с вершины v_n и двигаясь обратно к вершине v_1. При этом всякий раз надо выбирать такую вершину v_j после вершины v_i, чтобы выполнялось равенство $\lambda(v_i) - \lambda(v_j) = l(v_i v_j)$.

Пусть в графе на рис. 8.3 требуется найти кратчайший путь из вершины v_1 к вершине v_8. Возле каждого ребра дана его длина. Покажем изменение индексов вершин:

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
0	∞	∞	∞	∞	∞	∞	∞
	1	10	6	11	14	9	16
					13		15

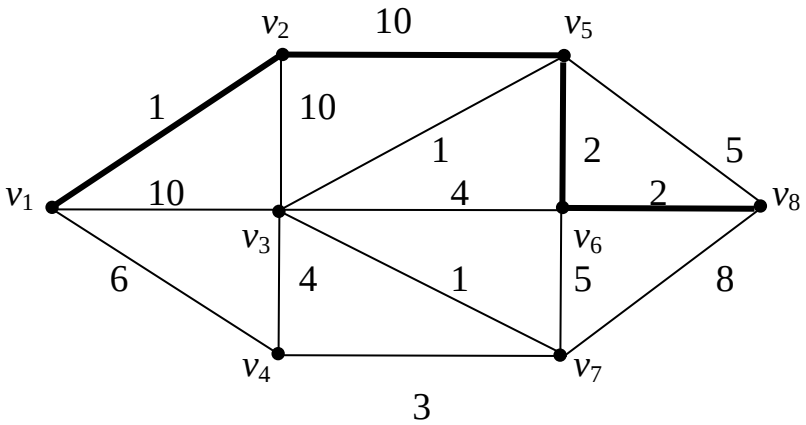


Рис. 8.3. Граф со взвешенными ребрами и выделенным кратчайшим путем

Длина кратчайшей цепи в данном графе равна 15. Двигаясь от вершины v_8 к вершине v_1, подходим сначала к вершине v_6, так как $\lambda(v_8) - \lambda(v_6) = l(v_6v_8) = 2$. Следуя тому же правилу, проходим вершину v_5 и затем вершину v_2. На рис. 8.3 выделены ребра, принадлежащие найденному пути.

Г л а в а 9

Планарные графы

9.1. Определения

Граф *укладывается* на некоторой поверхности, если его можно так нарисовать на этой поверхности, что никакие два ребра не будут иметь общей точки, кроме, возможно, общей вершины. На таком рисунке ребра графа изображаются линиями, а вершины – точками. Граф называется *планарным*, если его можно уложить на плоскости.

Плоский граф – это граф, уложенный на плоскости. Очевидно, каждый планарный граф изоморфен некоторому плоскому графу.

При проектировании печатного монтажа для электронной схемы возникают задачи определения планарности графа, разложения графа на планарные подграфы, а также укладки планарного графа на плоскости.

Гранью плоского графа называется область плоскости, ограниченная ребрами, любые две точки которой могут быть соединены линией, не пересекающей ребра графа. Каждый плоский граф имеет единственную неограниченную грань, которая называется *внешней*. Все остальные его грани являются *внутренними*. Плоский граф, изображенный на рис. 9.1, имеет три грани: f_1 и f_2 – внутренние грани, f_3 – внешнюю.

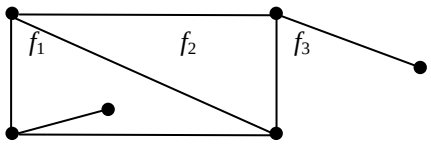


Рис. 9.1. Плоский граф

Т е о р е м а Э й л е р а. Для всякого связного плоского графа, имеющего n вершин, m ребер и f граней, имеет место соотношение $n - m + f = 2$.

Действительно, для дерева имеем $m = n - 1$ и $f = 1$. Следовательно, данная формула для дерева верна. Увеличивая число ребер в графе на некоторую величину при сохранении числа вершин, мы увеличиваем число граней на ту же величину. Следовательно, соотношение сохраняется. Оно называется *формулой Эйлера*.

Максимальным планарным графом называется планарный граф с числом вершин не меньше пяти, который перестает быть планарным при добавлении любого нового ребра. Пример такого графа дан на рис. 9.2

Легко показать, что каждая грань в таком графе ограничена тремя ребрами.

9.2. Простейшие непланарные графы

Полный граф K_5 и полный двудольный граф $K_{3,3}$ (рис. 9.3) являются простейшими непланарными графами. Нетрудно убедиться, что удаление любой вершины или любого ребра превращает их в планарные графы.

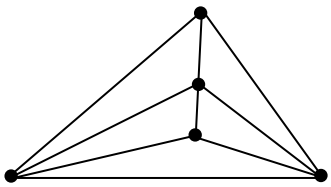


Рис. 9.2. Максимальный планарный граф

Граф $K_{3,3}$ можно использовать для решения *задачи о трех домах и трех колодцах*, которая ставится следующим образом. Имеются три дома и расположенные неподалеку три колодца. Можно ли проложить от каждого дома дорожки во всем трем колодцам так, чтобы эти дорожки не пересекались?

Приводимое ниже утверждение дает отрицательный ответ на этот вопрос.

Т е о р е м а П о н т р я г и н а – К у р а т о в с к о г о. Необходимым и достаточным условием непланарности графа является любое из следующих условий: 1) в графе можно выделить пять вершин, каждая из которых связана цепью с любой другой из них, причем все эти цепи не пересекаются по ребрам; 2) в графе можно выделить два множества, состоящие из трех вершин каждое, так, что каждая вершина одного множества связана цепью со всеми вершинами другого множества, причем все эти цепи не пересекаются по ребрам.

Другими словами, планарный граф не должен иметь подграфов, изображенных на рис. 9.3, где каждое ребро может быть заменено простой цепью.

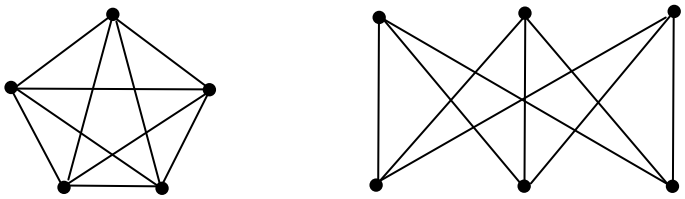


Рис. 9.3. Простейшие непланарные графы

9.3. Раскраска планарных графов

Двойственный граф G^* по отношению к плоскому графу G строится следующим образом. Внутри каждой грани графа G поместим вершину графа G^* и каждому ребру e графа G отнесем ребро графа G^*, которое соединяет его

вершины, помещенные на гранях, имеющих общее ребро *e* в графе *G*. Пример двойственного графа дан на рис. 9.4, где его ребра показаны тонкими линиями.

Раскраска граней плоского графа, при которой соседние грани раскрашиваются в различные цвета, эквивалентна раскраске вершин его двойственного графа. Таким образом, к раскраске вершин графа сводится известная задача о раскраске географической карты, при решении которой территории соседних государств требуется окрашивать различными цветами, минимизируя при этом число последних.

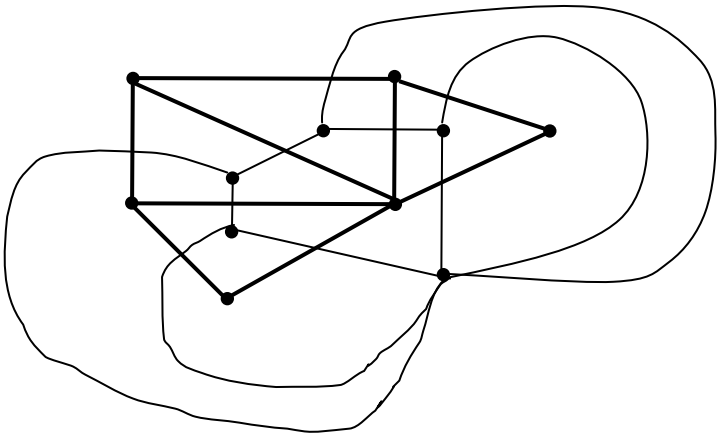


Рис. 9.4. Плоский граф и его двойственный граф

Доказано, что всякий планарный граф можно раскрасить не более чем в пять цветов, и долгое время существовала *гипотеза четырех красок*, утверждавшая, что всякий планарный граф можно раскрасить не более чем в четыре цвета. Было много безуспешных попыток ее доказать. В 1976 г. появилось сообщение о том, что данная гипотеза доказана с помощью вычислительной машины. Для этого ранее были найдены 1 482 конфигурации и дано доказательство утверждения, что если в максимальном планарном графе подграф вида такой конфигурации раскрашивается в четыре цвета, то и сам граф раскрашивается в четыре цвета. С помощью мощной вычислительной машины все графы, изоморфные таким конфигурациям, приблизительно за 2 000 ч машинного времени были раскрашены в четыре цвета. Однако в связи со сложностью проверки упомянутых доказательств не все специалисты с ними согласны.

Г л а в а 10

Комбинаторные задачи и методы комбинаторного поиска

Можно выделить три типа комбинаторных задач: *задачи подсчета* числа конфигураций определенного вида; *перечислительные задачи*, в результате решения которых получаются все конструкции определенного вида (например, получение всех независимых множеств графа); *оптимизационные комбинаторные задачи*, решением любой из которых является конструкция, обладающая оптимальным значением некоторого параметра среди всех конструкций данного вида (например, раскраска графа минимальным количеством цветов).

10.1. Задачи подсчета

Рассмотрим простейшие задачи подсчета.

Число размещений $U(m, n)$ показывает, сколькими способами можно разместить n предметов по m ящикам. Для каждого из n предметов имеется m вариантов размещения. Следовательно,

$$U(m, n) = m^n.$$

Числом перестановок $P(n)$ является число различных последовательностей, которые можно составить из n предметов. В последовательности всего n позиций. Зафиксируем один предмет. Его можно разместить в одну из n позиций, т. е. имеем n вариантов размещения. Для следующего предмета имеется $n-1$ вариантов размещения по незанятым позициям и т. д. Таким образом,

$$P(n) = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1 = n!.$$

Число размещений без повторений $A(m, n)$ представляет собой число способов размещения n предметов по m ящикам не более чем по одному в ящик (при этом считается, что $m \geq n$). Путем рассуждений, подобных предыдущим, получим

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

Число сочетаний $C(m, n)$ показывает, сколькими способами из m предметов можно выбрать n предметов. В данном случае не важно, в каком порядке эти предметы выбираются, поэтому

$$C(m, n) = \frac{A(m,n)}{n!} = \frac{m!}{(m-n)!n!}.$$

10.2. Особенности комбинаторных задач

Примерами оптимизационных комбинаторных задач, решение которых предполагает комбинаторный поиск, являются рассмотренные нами в предыдущих разделах поиск наибольшего независимого и наименьшего доминирующего множеств, раскраска графа.

В отличие от задач традиционной математики, где решение получается с помощью целенаправленной вычислительной процедуры, однозначно ведущей к цели, решение комбинаторной задачи сводится зачастую к *полному перебору* различных вариантов. Перебираются и испытываются конструкции определенного вида, среди которых должно находиться решение задачи. Как только выясняется, что очередная конструкция является решением, процесс поиска решения можно считать завершенным.

В традиционной математике трудоемкость задачи обычно не очень сильно зависит от размера области возможных решений, в то время как для комбинаторных задач эта зависимость весьма велика.

Комбинаторные задачи характерны еще тем, что множество, среди элементов которого отыскивается решение, всегда конечно. Реализовав полный перебор, либо найдем решение, либо убедимся в том, что решения нет. Таким образом, всякая подобная задача может быть решена за конечное время. Однако это не значит, что она может быть решена за практически приемлемое время даже с помощью самой быстродействующей вычислительной машины.

10.3. Вычислительная сложность

Трудоемкость алгоритма, или временна́я сложность, т. е. время, затрачиваемое на выполнение алгоритма, оценивается числом условных элементарных операций, которые необходимо выполнить при решении задачи. Естественно, эта величина зависит от объема исходных данных, который оценивается некоторым параметром. Например, для графа это может быть число вершин или число ребер. Трудоемкость алгоритма, таким образом, можно оценить некоторой функцией $f(n)$, где n – натуральное число, выражающее объем исходных данных.

Принято писать $f(n) = O(g(n))$, где $g(n)$ − некоторая конкретная функция от n, если найдется такая константа c, что $f(n) \le cg(n)$ для любого $n \ge 0$. При этом употребляют такие выражения: «трудоемкость алгоритма есть $O(g(n))$» или «алгоритм решает задачу за время $O(g(n))$». Если трудоемкость не зависит от объема исходных данных, то для ее обозначения используется символ $O(1)$.

Алгоритм трудоемкости $O(n)$ называют *линейным*. Алгоритм трудоемкости $O(n^b)$, где b – константа (возможно, дробная), называется *полиномиальным*. Если $g(n)$ является показательной функцией, например 2^n, то говорят, что алгоритм обладает *неполиномиальной*, или *экспоненциальной*, сложностью.

Оценка трудоемкости алгоритма позволяет судить о том, как влияет быстродействие вычислительной машины на время выполнения алгоритма. Пусть имеется пять алгоритмов, трудоемкость которых соответственно n, $n\log n$, n^2, n^3 и 2^n. Пусть условная элементарная операция, которая является единицей измерения трудоемкости алгоритма, выполняется за одну миллисекунду. В табл. 10.1, заимствованной из работы [3], показано, какого размера задачи могут быть решены каждым из этих алгоритмов за одну секунду, одну минуту и один час. Из этой таблицы видно, например, что за одну минуту алгоритм с трудоемкостью n^2 решает задачу в шесть раз большую, чем алгоритм с трудоемкостью n^3.

Следует, однако, иметь в виду, что трудоемкость, выражаемая большей степенью полинома, может иметь меньший множитель c из приведенного выше неравенства $f(n) \le cg(n)$. Точно так же сложность алгоритма, которая носит экспоненциальный характер, может иметь множитель, меньший, чем у полиномиальной сложности. При разработке компьютерных программ для решения практических задач важно знать, при каких значениях параметра n время выполнения экспоненциального алгоритма оказывается меньше, чем время выполнения полиномиального алгоритма, решающего ту же задачу.

Таблица 10.1

Связь трудоемкости алгоритма с максимальным размером задачи, решаемой за единицу времени

Временна́я сложность	Максимальный размер задачи		
	1 с	1 мин	1 ч
n	1000	6×10^4	$3{,}6 \times 10^6$
$n\log n$	140	4893	$2{,}0 \times 10^5$
n^2	31	244	1897
n^3	10	39	153
2^n	9	15	21

Для очень многих практических комбинаторных задач существуют алгоритмы только экспоненциальной трудоемкости. Может показаться, что с совершенствованием вычислительной техники и ростом быстродействия вычислительных машин проблема трудоемкости ослабевает. Однако данные, приведенные в табл. 10.2 [3], говорят, что это не так. Пусть следующее поколение вычислительных машин будет иметь быстродействие, в десять раз большее, чем у современных вычислительных машин. В табл. 10.2 показано, как благодаря увеличению быстродействия возрастут размеры задач, которые могут быть решены за некоторую фиксированную единицу времени. Задачи достаточно большого размера, решаемые только алгоритмами экспоненциальной трудоемкости, вообще не могут быть решены за практически

приемлемое время, даже если надеяться на существенное увеличение быстродействия вычислительных машин в будущем.

Таблица 10.2

Связь размера задачи, решаемой за заданное время,
с быстродействием вычислительной машины

Временна́я сложность	Максимальный размер задачи	
	до ускорения	после ускорения
n	s_1	$10\ s_1$
$n \log n$	s_2	$\approx 10\ s_2$
n^2	s_3	$3{,}16\ s_3$
n^3	s_4	$2{,}15\ s_4$
2^n	s_5	$s_5 + 3{,}3$

Иногда удается найти способы сокращения перебора благодаря некоторым особенностям конкретных исходных данных.

Другой путь выхода из такого положения – использование приближенных методов. Для практических задач не всегда требуется получать точное решение. Часто достаточно иметь решение, близкое к оптимальному. Пример приближенного метода рассмотрен нами ранее при решении задачи раскраски графа.

10.4. Методы комбинаторного поиска

Один из наиболее общих и плодотворных подходов к решению комбинаторных задач заключается в применении *дерева поиска*. В дереве выделяется вершина, которая называется *корнем дерева* и которая ставится в соответствие исходной ситуации в процессе решения задачи. Остальные вершины сопоставляются с ситуациями, которые можно достичь в данном процессе. Выделение корня придает дереву ориентацию, при которой все пути ведут из корня в остальные вершины. Дуги дерева соответствуют некоторым простым операциям, представляющим шаги процесса решения, и связывают вершины, соответствующие ситуациям, одна из которых преобразуется в другую в результате выполнения шага. Для ситуации характерно разнообразие вариантов выбора очередного шага, представленных дугами, исходящими из соответствующей вершины. Некоторые ситуации соответствуют решениям.

Дерево поиска не задается априори, а строится в процессе поиска: когда возникает некоторая ситуация, тогда и определяются возможные направления процесса, которые представляются исходящими из вершины дугами. Естественным является стремление сокращать число этих дуг, чтобы быстрее найти решение. Способы этого сокращения строятся с учетом особенностей конкретных задач.

Довольно общим средством повышения эффективности процесса решения задачи является *редуцирование*, т. е. упрощение текущей ситуации, сокращающее объем вычислений, проводимых при анализе множества

вытекающих из нее вариантов. Способы редуцирования определяются особенностями конкретной задачи и исходных данных.

Процедуру комбинаторного поиска удобно проследить на примере решения задачи о кратчайшем покрытии, которую рассмотрим в следующем разделе.

Г л а в а 11

Задача о кратчайшем покрытии

11.1. Постановка задачи

Многие комбинаторные оптимизационные задачи сводятся к *задаче о кратчайшем покрытии*, которая ставится следующим образом. Пусть даны некоторое множество $A = \{a_1, a_2, \ldots, a_n\}$ и совокупность его подмножеств $B_1, B_2, \ldots, B_m$, т. е. $B_i \subseteq A$, $i = 1, 2, \ldots, m$, причем $B_1 \cup B_2 \cup \ldots \cup B_m = A$. Требуется среди данных подмножеств выделить такую совокупность $B_{i_1}, B_{i_2}, \ldots, B_{i_k}$ с минимальным k, чтобы каждый элемент из A попал хотя бы в одно из B_{i_j} ($j = 1, 2, \ldots, k$), т. е. $B_{i_1} \cup B_{i_2} \cup \ldots \cup B_{i_k} = A$.

Одной из интерпретаций этой задачи является задача о переводчиках. Из некоторого коллектива переводчиков, число которых m и каждый из которых владеет несколькими определенными языками, требуется скомплектовать минимальную по числу членов группу, такую, чтобы она смогла обеспечить перевод с любого из заданного множества языков, число которых n. Здесь A – множество языков, перевод с которых требуется обеспечить, а B_i – множество языков, которыми владеет i-й переводчик.

Удобно рассматривать матричную формулировку данной задачи, при которой совокупность $B_1, B_2, \ldots, B_m$ задается в виде булевой матрицы, строки которой соответствуют подмножествам из данной совокупности, а столбцы – элементам множества A. Элемент i-й строки и j-го столбца имеет значение 1, если и только если $a_j \in B_i$. В этом случае говорят, что i-я строка *покрывает* j-й столбец. Требуется найти такое множество строк данной матрицы, чтобы каждый ее столбец имел единицу хотя бы в одной строке из этого множества, и при этом мощность выбранного множества должна быть минимальной.

11.2. Приближенные методы решения задачи

Существуют приближенные методы решения данной задачи. Например, ее можно решать с помощью жадного алгоритма, представляющего собой многошаговый процесс, где на каждом шаге выбирается и включается в покрытие строка заданной матрицы, покрывающая наибольшее число из еще не покрытых столбцов. Этот процесс заканчивается, когда все столбцы матрицы оказываются покрытыми. Применение жадного алгоритма иногда дает точное решение, но гарантии этому нет. Например, если задана матрица

$$\begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} B_1 \\ B_2 \\ B_3 \end{matrix},$$

первой для включения в формируемое решение жадный алгоритм выберет строку B_1, после чего для покрытия оставшихся столбцов должны быть включены в решение обе строки B_2 и B_3. Кратчайшее же покрытие данной матрицы составляют только две строки – B_2 и B_3.

Более близкое к кратчайшему покрытие получается чаще всего с помощью «*минимаксного*» алгоритма. Он представляет собой многошаговый процесс, на каждом шаге которого выбирается столбец с минимальным числом единиц и из покрывающих его строк для включения в решение выбирается та, которая покрывает максимальное число непокрытых столбцов. Пусть, например, задана матрица

$$\begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ B_7 \\ B_8 \\ B_9 \end{matrix}.$$

Одним из столбцов с минимальным числом единиц является столбец a_6. Из покрывающих его строк максимальное число столбцов покрывает строка B_6. Включим эту строку в решение и удалим ее и столбцы, которые она покрывает, в результате чего получим

$$\begin{array}{cccccc} a_2 & a_4 & a_5 & a_7 & a_8 & a_{10} & \\ \left[\begin{array}{cccccc} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right] & \begin{array}{c} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_7 \\ B_8 \\ B_9 \end{array} \end{array}.$$

Из оставшихся столбцов минимальное число единиц имеет столбец a_{10}. Покрывающие его строки B_4 и B_9 имеют одинаковое число единиц, т. е. одинаковое число покрываемых ими, но еще не покрытых столбцов. Включаем в решение первую по порядку строку B_4 и получаем матрицу

$$\begin{array}{ccc} a_2 & a_5 & a_7 & \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{array}\right] & \begin{array}{c} B_1 \\ B_2 \\ B_3 \\ B_5 \\ B_7 \\ B_8 \\ B_9 \end{array} \end{array}.$$

В полученной матрице столбцом с минимальным числом единиц является столбец a_2, а из покрывающих его строк строка B_7 имеет максимальное число единиц. Включение этой строки в решение завершает процесс, в результате которого полученным покрытием является $\{B_4, B_6, B_7\}$. Как будет показано ниже, это решение является точным.

11.3. Точный метод

Точный метод нахождения кратчайшего покрытия представляет собой обход дерева поиска. Текущая ситуация, соответствующая некоторой вершине дерева поиска, представляется переменной матрицей $\boldsymbol{X}$, которая показывает, какие столбцы еще не покрыты и какие строки можно использовать для их покрытия. В этой ситуации выбирается первый из столбцов с минимальным числом единиц – так минимизируется число вариантов продолжения поиска. Очередной шаг процесса состоит в выборе покрывающей строки для этого

столбца и пробном включении ее в получаемое решение. Таким образом, вершины дерева поиска соответствуют некоторым столбцам исходной матрицы, а дуги – выбираемым для их покрытия строкам.

Начальное значение матрицы $\boldsymbol{X}$ совпадает с исходной матрицей. Последующие значения получаются удалением строк, включаемых в решение, и столбцов, покрытых этими строками. Кроме того, выполняются следующие *правила редукции*.

1. Если столбец k имеет единицы везде, где имеет единицы столбец l, то столбец k можно удалить. Любая строка, покрывающая столбец l, покрывает также столбец k. Поэтому при поиске покрытия столбец k можно не рассматривать. Достаточно, чтобы в покрытие была включена какая-либо из строк, покрывающих столбец l.

2. Если строка i имеет единицы везде, где имеет единицы строка j, то строку j можно удалить. Действительно, пусть в некотором кратчайшем покрытии имеется строка j. Очевидно, данное покрытие останется кратчайшим, если в нем строку j заменить строкой i.

Продемонстрируем описанный процесс на матрице из предыдущего примера:

$$\begin{array}{cccccccccc} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} \\ \left[\begin{array}{cccccccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right] & \begin{array}{c} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ B_7 \\ B_8 \\ B_9 \end{array} \end{array}.$$

На первом шаге выбираем столбец a_6, содержащий две единицы. Среди покрывающих ее строк выбираем такую, которая покрывает наибольшее число столбцов. Такой строкой является строка B_6. Удалив эту строку и покрываемые ею столбцы, получим следующее значение матрицы $\boldsymbol{X}$:

$$\begin{array}{c} \begin{matrix} a_2 & a_4 & a_5 & a_7 & a_8 & a_{10} \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_7 \\ B_8 \\ B_9 \end{matrix}.$$

После удаления строк B_1, B_2 и B_8 согласно второму правилу редукции матрица $\boldsymbol{X}$ будет иметь следующий вид:

$$\begin{array}{c} \begin{matrix} a_2 & a_4 & a_5 & a_7 & a_8 & a_{10} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} B_3 \\ B_4 \\ B_5 \\ B_7 \\ B_9 \end{matrix}.$$

Одним из столбцов, обладающих минимальным числом единиц, является столбец a_2. Обе покрывающие его строки B_3 и B_7 содержат по три единицы. Выбираем первую по порядку строку B_3 и включаем ее в формируемое покрытие. Теперь имеем множество $\{B_3, B_6\}$. Этот шаг приводит к матрице

$$\begin{array}{c} \begin{matrix} a_4 & a_7 & a_{10} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} B_4 \\ B_5 \\ B_7 \\ B_9 \end{matrix}.$$

После удаления строки B_7 по второму правилу редукции получим матрицу, каждая строка и каждый столбец которой содержат ровно две единицы. Выбрав строку B_4, покрывающую столбец a_4, и проведя аналогичные преобразования, получим матрицу с одним столбцом a_7 и двумя строками B_5 и B_9, любая из которых покрывает оставшийся столбец. Таким образом, получено покрытие $\{B_3, B_4, B_5, B_6\}$, но пройдена пока только одна ветвь дерева поиска, и до

совершения полного обхода дерева неизвестно, является ли это покрытие кратчайшим.

Возвращаемся к ситуации, когда очередным столбцом для покрытия взят a_2. Теперь вместо строки B_3 возьмем для покрытия столбца a_2 строку B_7. Действуя дальше аналогичным образом, получаем очередное покрытие $\{B_4, B_6, B_7\}$, которое вытесняет предыдущее, так как оно оказалось лучше, однако и его пока нельзя назвать кратчайшим.

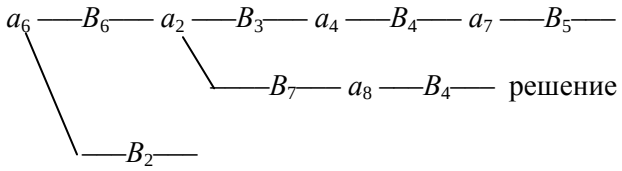


Рис. 11.1. Дерево поиска кратчайшего покрытия

Возвратившись к начальной вершине дерева поиска и следуя по дуге, соответствующей строке B_2, убеждаемся, что длина покрытия не может быть меньше трех. На этом поиск можно закончить и выдать в качестве решения множество $\{B_4, B_6, B_7\}$. На дереве поиска, обход которого совершался в процессе решения данного примера (рис. 11.1), вершинам приписаны столбцы, а дугам – строки.

Г л а в а 12

Булевы функции

12.1. Способы задания булевой функции

Пусть $x_1, x_2, \ldots, x_n$ – некоторые *булевы переменные*, т. е. переменные, принимающие значение из множества $\{0, 1\}$. Упорядоченную совокупность булевых переменных $(x_1, x_2, \ldots, x_n)$ можно рассматривать как n-компонентный *булев вектор* $\boldsymbol{x}$. Число компонент вектора определяет его длину, или *размерность*. При фиксации значений всех переменных получается *набор значений* переменных $(x_1, x_2, \ldots, x_n)$, задаваемый булевым вектором длины n, состоящим из констант 0 и 1. Очевидно, 2^n – число всех таких векторов. Они образуют *булево пространство*. *Булевой функцией* называется функция $f: \{0, 1\}^n \to \{0, 1\}$. Областью определения булевой функции является булево пространство $M = \{0, 1\}^n$, областью значений – множество $\{0, 1\}$.

Задание булевой функции f на булевом пространстве M делит его на две части: M_f^1 – область, где функция принимает значение 1, и M_f^0 – область, где функция принимает значение 0. Множество M_f^1 называется *характеристическим множеством* функции f.

Универсальным способом задания для любой дискретной функции является табличный способ. Таблица, представляющая функцию и называемая *таблицей истинности*, имеет два столбца. В левом столбце перечислены все наборы значений аргументов, в правом столбце – соответствующие им значения функции. Примером задания булевой функции от трех аргументов является табл. 12.1.

Таблица 12.1

Задание функции $f(x_1, x_2, x_3)$

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Для задания булевой функции можно ограничиться перечислением элементов ее характеристического множества M_f^1. Множество M_f^1 задается булевой матрицей, строки которой представляют элементы этого множества.

Следующая матрица, задающая приведенную выше функцию, является *матричным способом* представления булевой функции:

$$\begin{matrix} x_1 & x_2 & x_3 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Компактность представления характеристического множества M_f^1 можно повысить, используя *троичные векторы*, компоненты которых могут принимать в качестве своих значений кроме символов 0 и 1 также символ «–». В этом случае значения булевой функции будут задаваться не на отдельных элементах, а на интервалах пространства переменных $x_1, x_2, ..., x_n$. Чтобы определить понятие интервала булева пространства, введем отношение ≤ на множестве булевых векторов.

Булевы векторы $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ и $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ находятся в отношении ≤ ($\boldsymbol{a} \le \boldsymbol{b}$) и говорят, что $\boldsymbol{a}$ меньше $\boldsymbol{b}$, если $a_i \le b_i$ для любого $i = 1, 2, \ldots, n$, в противном случае они *несравнимы*. При этом считается, что $0 \le 1$. Тогда *интервалом булева пространства* называется множество векторов, среди которых есть минимальный и максимальный векторы, а также все векторы, меньшие максимального и большие минимального. Интервал представляется троичным вектором, который задает множество всех булевых векторов, получаемых заменой символа «–» на 1 или 0.

Троичная матрица *эквивалентна* булевой матрице, получаемой из нее заменой каждой троичной строки на представляемую ею совокупность булевых строк с последующим устранением повторяющихся строк. Приведенная выше булева матрица оказывается эквивалентной троичной матрице

$$\begin{matrix} x_1 & x_2 & x_3 \end{matrix}$$
$$\begin{bmatrix} - & - & 1 \\ 0 & 1 & - \end{bmatrix},$$

которая представляет ту же булеву функцию $f(x_1, x_2, x_3)$. Такой способ задания булевой функции называют еще *интервальным*. Представление булевой функции троичной матрицей не однозначно, т. е. для одной и той же булевой матрицы существует в общем случае не одна эквивалентная ей троичная матрица.

Две троичные матрицы *эквивалентны*, если они эквивалентны одной и той же булевой матрице, т. е. если они представляют одну и ту же булеву функцию.

Векторное задание булевой функции представляет собой булев вектор, компоненты которого соответствуют наборам значений аргументов. Эти наборы упорядочиваются обычно согласно порядку чисел, двоичные коды которых они представляют. Рассмотренная выше булева функция $f(x_1, x_2, x_3)$ представляется вектором (0 1 1 1 0 1 0 1), показывающим, что функция принимает значение 0 на наборах (0 0 0), (1 0 0), (1 1 0) и значение 1 на наборах (0 0 1), (0 1 0), (0 1 1), (1 0 1), (1 1 1). Заметим, что этот вектор совпадает с правым столбцом табл. 12.1.

Если значения булевой функции определены для всех 2^n наборов значений вектора $\boldsymbol{x}$, она называется *полностью определенной*, в противном случае – *не полностью определенной*, или *частичной*. Задание не полностью определенной булевой функции f разбивает булево пространство на три множества: кроме M_f^1 и M_f^0 в нем присутствует множество M_f^-, где значения функции f не определены. Для задания частичной булевой функции необходимо задать не менее двух множеств. Обычно это M_f^1 и M_f^0.

Далее будет рассмотрен *алгебраический способ* задания булевой функции.

12.2. Элементарные булевы функции и алгебраические формы

Рассматривая векторную форму задания булевой функции, легко определить число булевых функций от n переменных – это число всех 2^n-компонентных булевых векторов, т. е. 2^{2^n}. Однако это число учитывает также функции и от меньшего числа аргументов. Любую функцию от n аргументов можно считать функцией от большего числа аргументов. Для этого вводится понятие *существенной зависимости* и *несущественной зависимости*. Функция $f(x_1, x_2, \ldots, x_n)$ существенно зависит от аргумента x_i, если

$$f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \neq f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n).$$

Переменная x_i в этом случае называется *существенным аргументом*. В противном случае она является *несущественным* или *фиктивным аргументом*.

Элементарными булевыми функциями являются функции от одной и двух переменных. Число функций от одной переменной равно $2^{2^1} = 4$. Эти функции представлены в табл. 12.2. Две из них, f_0 и f_3, являются константами 0 и 1, переменная x для них является несущественным аргументом. Функция f_1 также является тривиальной, любое ее значение совпадает со значением аргумента: $f_1(x) = x$. Нетривиальной функцией является функция $f_2(x) = \bar{x}$, называемая *отрицанием*, или *инверсией*. Ее значение всегда противоположно значению аргумента x. Из табл. 12.2 видно, что $\bar{0} = 1$ и $\bar{1} = 0$.

В табл. 12.3 приведены все булевы функции $f_i(x_1, x_2)$ от двух аргументов. В левом столбце показаны их выражения в терминах нескольких функций, принятых за основные.

Таблица 12.2

Булевы функции от одного аргумента

x	0	1
$f_0 = 0$	0	0
$f_1 = x$	0	1
$f_2 = \bar{x}$	1	0
$f_3 = 1$	1	1

Таблица 12.3

Булевы функции от двух аргументов

x_1	0	0	1	1
x_2	0	1	0	1
$f_0 = 0$ – константа 0	0	0	0	0
$f_1 = x_1 \wedge x_2$ – конъюнкция	0	0	0	1
f_2 – отрицание импликации	0	0	1	0
$f_3 = x_1$	0	0	1	1
f_4 – отрицание обратной импликации	0	1	0	0
$f_5 = x_2$	0	1	0	1
$f_6 = x_1 \oplus x_2$ – сложение по модулю 2	0	1	1	0
$f_7 = x_1 \vee x_2$ - дизъюнкция	0	1	1	1
$f_8 = x_1 \uparrow x_2$ – стрелка Пирса	1	0	0	0
$f_9 = x_1 \sim x_2$ – эквиваленция	1	0	0	1
$f_{10} = \bar{x}_2$	1	0	1	0
$f_{11} = x_2 \rightarrow x_1$ – обратная импликация	1	0	1	1
$f_{12} = \bar{x}_1$	1	1	0	0
$f_{13} = x_1 \rightarrow x_2$ – импликация	1	1	0	1
$f_{14} = x_1 \mid x_2$ – штрих Шеффера	1	1	1	0
$f_{15} = 1$ – константа 1	1	1	1	1

Элементарные функции имеют особое значение в теории булевых функций, поскольку каждая из них может быть выражена одноместной или двухместной алгебраической операцией. Все операции, представленные в табл. 12.3, составляют *алгебру логики*. Любая булева функция от любого числа аргументов может быть представлена формулой алгебры логики. Формулу, содержащую более чем одну операцию, можно рассматривать как *суперпозицию* элементарных функций. Под суперпозицией функций понимается использование одних функций в качестве аргументов других функций. Для булевых функций это является возможным благодаря совпадению области значений функций с областями значений их аргументов. Таким образом, *алгебраическое задание* булевой функции представляет собой формулу, по которой вычисляется значение этой функции. Понятие *формулы* определим индуктивно следующим образом:

1) каждый символ переменной есть формула;

2) если A и B – формулы, то формулами являются $\overline{A}$ и $(A * B)$, где $*$ – любая операция алгебры логики;

3) других формул нет.

Для установления порядка выполнения операций в формулах используются скобки. При отсутствии скобок порядок устанавливается согласно приоритетам операций. Первым приоритетом обладает операция отрицания, затем выполняется $\wedge$. Третьим приоритетом обладают операции $\vee$ и $\oplus$, четвертым приоритетом – операции $\sim$ и $\to$. Для упрощения написания формул иногда символ конъюнкции опускается. Процесс вычисления значений функции $f(x_1, x_2, x_3) = \overline{(x_1 \vee x_2)x_3} \to \bar{x}_1 \vee x_2 x_3$ по ее формуле покажем с помощью табл. 12.4, где последовательность столбцов соответствует порядку выполнения операций.

Таблица 12.4

Вычисление по формуле

x_1	x_2	x_3	$x_1 \vee x_2$	$(x_1 \vee x_2)\, x_3$	$\overline{(x_1 \vee x_2)x_3}$	$\bar{x}_1$	x_2x_3	$\bar{x}_1 \vee x_2x_3$	$f(x_1, x_2, x_3)$
0	0	0	0	0	1	1	0	1	1
0	0	1	0	0	1	1	0	1	1
0	1	0	1	0	1	1	0	1	1
0	1	1	1	1	0	1	1	1	1
1	0	0	1	0	1	0	0	0	0
1	0	1	1	1	0	0	0	0	1
1	1	0	1	0	1	0	0	0	0
1	1	1	1	1	0	0	1	1	1

Две разные формулы могут представлять одну и ту же функцию. Такие формулы называются *равносильными*. Если A и B – равносильные формулы, то $A = B$. В этом случае, если A является частью другой формулы, то вместо нее можно подставить B, в результате чего получится формула, равносильная исходной.

Алгебра, содержащая только три операции $\bar{}$, $\wedge$ и $\vee$, называется *булевой*, так же как и формула, представляющая некоторую композицию этих операций. Легко проверить по табл. 12.3 следующие основные законы булевой алгебры.

Коммутативность:

$x \vee y = y \vee x$; $\qquad x\, y = y\, x$.

Ассоциативность:

$x \vee (y \vee z) = (x \vee y) \vee z$; $\qquad x\,(y\, z) = (x\, y)\, z$.

Дистрибутивность:

$x\,(y \vee z) = x\, y \vee x\, z$; $\qquad x \vee y\, z = (x \vee y)\,(x \vee z)$.

Идемпотентность:

$x \vee x = x$; $\quad x\,x = x$.

Законы де Моргана:

$\overline{x \vee y} = \bar{x}\,\bar{y}$; $\quad \overline{xy} = \bar{x} \vee \bar{y}$.

Законы операций с константами:

$x \vee 0 = x$; $\quad x\,1 = x$;

$x\,0 = 0$; $\quad x \vee 1 = 1$;

$x \vee \bar{x} = 1$; $\quad x\,\bar{x} = 0$.

Закон двойного отрицания:

$\bar{\bar{x}} = x$.

Здесь действует принцип двойственности, т. е. для каждой пары формул, представляющих тот или иной закон, справедливо следующее утверждение: одна из формул получается из другой взаимной заменой всех символов конъюнкции на символы дизъюнкции, всех нулей – на единицы и всех единиц – на нули.

На основании этих формул выводятся следующие соотношения.

Закон поглощения:

$x \vee x\,y = x$; $\quad x\,(x \vee y) = x$.

Действительно,

$x \vee x\,y = x\,1 \vee x\,y = x\,(1 \vee y) = x1 = x$;

$x\,(x \vee y) = x\,x \vee x\,y = x \vee x\,y = x$.

Закон простого склеивания:

$x\,y \vee x\,\bar{y} = x$; $\quad (x \vee y)\,(x \vee \bar{y}) = x$.

Левая формула выводится с помощью закона дистрибутивности конъюнкции относительно дизъюнкции: $x\,y \vee x\,\bar{y} = x(y \vee \bar{y}) = x$. Для вывода правой формулы достаточно раскрыть скобки и применить законы идемпотентности и поглощения.

Закон обобщенного склеивания:

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

Эту формулу можно вывести следующим образом:

$$x\,y \vee \bar{x}\,z \vee y\,z = x\,y \vee \bar{x}\,z \vee y\,z\,(x \vee \bar{x}) = x\,y\,(1 \vee z) \vee \bar{x}\,z\,(1 \vee y) = x\,y \vee \bar{x}\,z,$$

а если подставить $y = 1$, то получим

$$x \vee \bar{x}\,z = x\,1 \vee \bar{x}\,z \vee 1\,z = x \vee z.$$

Все операции алгебры логики можно выразить через булевы операции. Справедливость следующих формул можно доказать простой подстановкой значений из табл. 12.3:

$$x \oplus y = x\,\bar{y} \vee \bar{x}\,y;$$
$$x \sim y = \bar{x}\,\bar{y} \vee x\,y;$$
$$x \to y = \bar{x} \vee y.$$

Пользуясь этими формулами, построим булево выражение, эквивалентное формуле $((x \to y) \vee (x \oplus z))\ \bar{y}$:

$$((x \to y) \vee (x \oplus z))\ \bar{y} = (\ \bar{x} \vee y \vee x\ \bar{z} \vee \bar{x}\,z)\ \bar{y} = (\ \bar{x} \vee y \vee \bar{z})\ \bar{y} = \bar{x}\ \bar{y} \vee \bar{y}\ \bar{z}.$$

12.3. Интерпретации булевой алгебры

Рассматриваемая абстрактная булева алгебра имеет ряд интерпретаций, используемых в различных приложениях.

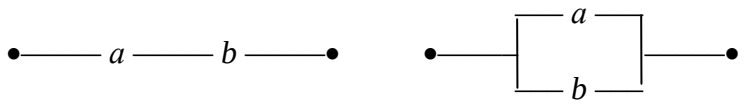
Булева алгебра множеств описана в гл. 1. Здесь значениями переменных служат подмножества универсального множества U. Константам 1 и 0 соответствуют множества U и $\varnothing$. Все соотношения, приведенные в гл. 1, совпадут с основными законами абстрактной булевой алгебры, если операцию дополнения множества заменить на операцию отрицания, а операции $\cap$ и $\cup$ (пересечения и объединения множеств) – соответственно на операции $\wedge$ и $\vee$ (конъюнкции и дизъюнкции).

Интерпретацией абстрактной булевой алгебры является также *алгебра событий*, используемая в теории вероятностей. Алгебру событий составляют семейство подмножеств множества элементарных событий U и определяемые над этими подмножествами операции отрицания ($\neg$), объединения ($\cup$) и пересечения ($\cap$). Любое событие может произойти или не произойти (наступить или не наступить). Отсутствие события A обозначается как $\neg A$. Событие, состоящее в наступлении обоих событий A и B, называется *произведением* событий A и B и обозначается $A \cap B$ или AB. Событие, состоящее в наступлении хотя бы одного из событий A и B, называется *суммой* событий A и B и обозначается $A \cup B$.

Еще одной интерпретацией является *алгебра переключательных схем*. Переменным этой алгебры соответствуют элементы переключательной схемы –

переключатели. Переключательный элемент, состояние которого представляется булевой переменной a, может быть замкнут, тогда через него течет ток и $a = 1$. Если он разомкнут, то тока нет и $a = 0$. По состояниям переключателей в схеме можно определить, проходит ли по данной схеме ток. На рис. 12.1, *а* изображено последовательное соединение двух переключателей a и b. Данная схема будет пропускать ток в том и только в том случае, когда оба переключателя замкнуты, т. е. если $a \wedge b = 1$. На рис. 12.1, *б* изображено параллельное соединение переключателей a и b. Ток будет протекать, если замкнут хотя бы один из переключателей, т. е. если $a \vee b = 1$.

Два или более переключателей можно условно связать таким образом, чтобы они замыкались и размыкались одновременно. Такие переключатели обычно обозначаются одним и тем же символом. Каждому переключателю можно поставить в соответствие другой переключатель так, чтобы когда один из них замкнут, другой был разомкнут. Если один из них обозначить буквой a, то другой примет обозначение $\bar{a}$. В схеме на рис. 12.2 пойдет ток, если и только если $a\,b \vee b\,\bar{c} \vee \bar{a}\,\bar{b} = 1$. Левая часть этого уравнения представляет структуру схемы.



а) *б)*

Рис. 12.1. Примеры соединения переключателей: *а)* последовательное; *б)* параллельное

Другим типом переключательной схемы является схема из электронных логических элементов, где булевы переменные представляются сигналами в виде высокого или низкого потенциала, а элементы (рис. 12.3) реализуют операции отрицания, конъюнкции и дизъюнкции.

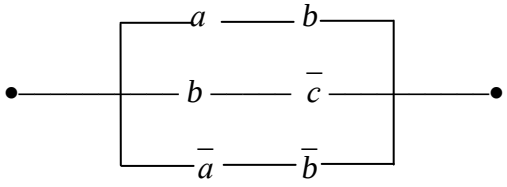


Рис. 12.2. Пример переключательной схемы

Если высокому потенциалу поставить в соответствие 1, а низкому – 0, то на выходе элемента дизъюнкции окажется высокий потенциал тогда, когда хотя бы на одном из входов элемента присутствует высокий потенциал. Низкий потенциал появится на выходе элемента, когда оба его входа будут иметь низкий потенциал.

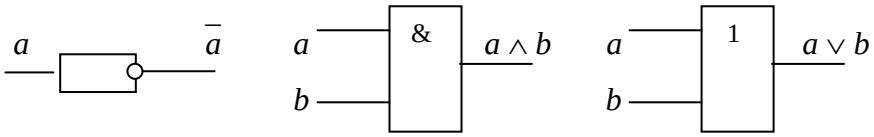


Рис. 12.3. Логические элементы, реализующие операции $\bar{}$, $\wedge$ и $\vee$

На выходе элемента конъюнкции создастся высокий потенциал тогда и только тогда, когда оба его входа будут иметь высокий потенциал.

Выход элемента отрицания примет высокий потенциал в том и только том случае, когда на его входе будет низкий потенциал.

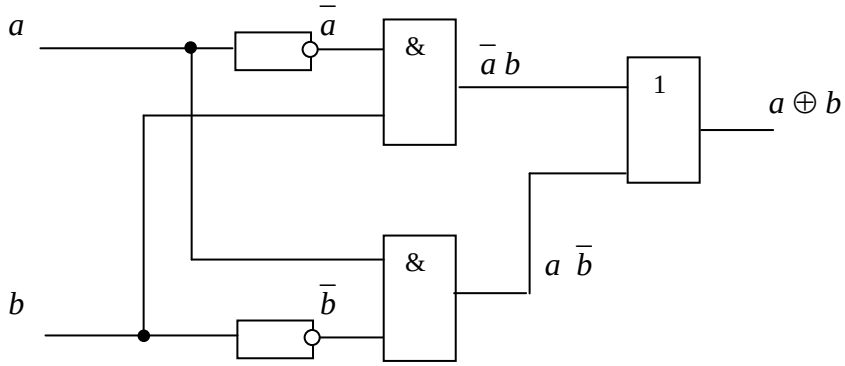


Рис. 12.4. Схема, реализующая операцию «сложение по модулю два»

Отдельные логические элементы можно соединять в схемы, причем структура схемы представляется при этом выражением булевой алгебры. Так, схема, реализующая сложение по модулю два, показана на рис. 12.4. Ее структура представлена правой частью равенства $a \oplus b = a\ \bar{b} \vee \bar{a}\ b$.

В *исчислении высказываний* переменными являются высказывания, принимающие истинные или ложные значения, которые соответствуют константам 1 и 0. Символы операций и их названия в данном случае совпадают не случайно. На основе исчисления высказываний можно выделить *булеву алгебру высказываний*, которая является одной из интерпретаций абстрактной булевой алгебры. Высказывание $\bar{a}$ истинно тогда и только тогда, когда a ложно. Оно читается как «не a» или «не верно, что a». Высказывание $a \wedge b$, читаемое как «a и b», истинно тогда и только тогда, когда истинны оба высказывания a и b. Высказывание $a \vee b$ читается как «a или b». Оно истинно, если хотя бы одно из высказываний a и b истинно, и ложно, если оба высказывания ложны.

Другие операции алгебры логики также могут иметь интерпретации в исчислении высказываний. Союз «или» может быть использован при прочтении высказывания $a \oplus b$. Наряду с «a либо b» его можно читать как «или a, или b». Оно истинно, когда истинно только одно из высказываний a и b, и ложно, когда оба высказывания истинны или оба ложны. Высказывание $a \sim b$

истинно тогда и только тогда, когда значения истинности высказываний a и b совпадают. Это высказывание может быть прочитано следующим образом: «a равносильно b», «a, если и только если b», «a тогда и только тогда, когда b». Импликация $a \to b$ читается как «если a, то b». Это высказывание ложно, когда a истинно, а b ложно. Во всех остальных случаях оно истинно.

12.4. Представление операций над булевыми функциями операциями над их характеристическими множествами

Пусть некоторые булевы функции f_1 и f_2 заданы с помощью характеристических множеств $M^1_{f_1}$ и $M^1_{f_2}$. Чтобы получить характеристическое множество результата некоторой операции над f_1 и f_2, надо выполнить соответствующую теоретико-множественную операцию над $M^1_{f_1}$ и $M^1_{f_2}$, привлекая, если это необходимо, множества $M^0_{f_1}$ и $M^0_{f_2}$, которые являются дополнениями множеств $M^1_{f_1}$ и $M^1_{f_2}$. Таким образом,

если $f = f_1 \wedge f_2$, то $M^1_f = M^1_{f_1} \cap M^1_{f_2}$;

если $f = f_1 \vee f_2$, то $M^1_f = M^1_{f_1} \cup M^1_{f_2}$;

если $f = f_1 \oplus f_2$, то $M^1_f = (M^1_{f_1} \cap M^0_{f_2}) \cup (M^0_{f_1} \cap M^1_{f_2})$;

если $f = f_1 \to f_2$, то $M^1_f = M^0_{f_1} \cup M^1_{f_2}$;

если $f = f_1 \sim f_2$, то $M^1_f = (M^1_{f_1} \cap M^1_{f_2}) \cup (M^0_{f_1} \cap M^0_{f_2})$;

если $f_1 = \overline{f}_2$, то $M^1_{f_1} = M^0_{f_2}$.

Г л а в а 13

Нормальные формы

13.1. Дизъюнктивные нормальные формы

Переменные $x_1, x_2, \ldots, x_n$ и их инверсии назовем *литералами* и введем обозначение a^σ, где $a^\sigma = a$, если $\sigma = 1$, и $a^\sigma = \bar{a}$, если $\sigma = 0$. *Элементарной конъюнкцией* K_i является многоместная конъюнкция попарно различных литералов, т. е. $K_i = x_{i_1}^{\sigma_1} x_{i_2}^{\sigma_2} \ldots x_{i_r}^{\sigma_r}$. К элементарным конъюнкциям относятся также одиночные литералы и константа 1 – конъюнкция, состоящая из пустого множества литералов. Число литералов r элементарной конъюнкции называется ее *рангом*. Элементарная конъюнкция называется *полной* относительно переменных $x_1, x_2, \ldots, x_n$, если она содержит символы всех переменных (со знаком отрицания или без него). Ранг таких конъюнкций равен n.

Дизъюнктивная нормальная форма (ДНФ) – это выражение вида $\bigvee_{i=1}^{m} K_i$, т. е. дизъюнкция элементарных конъюнкций. Примером дизъюнктивной нормальной формы является выражение $x_1 \bar{x}_2 \vee x_2 x_3 x_4 \vee \bar{x}_1 x_3$, где две конъюнкции имеют ранг 2 и одна конъюнкция – ранг 3. Одна элементарная конъюнкция также может считаться ДНФ.

13.2. Дизъюнктивное разложение Шеннона

Т е о р е м а Ш е н н о н а. Любая булева функция $f(x_1, x_2, \ldots, x_n)$ при любом m $(1 \le m \le n)$ может быть представлена в следующем виде:

$$f(x_1, x_2, \ldots, x_n) = \bigvee_{\sigma_1, \sigma_2, \ldots, \sigma_m} x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_m^{\sigma_m} f(\sigma_1, \sigma_2, \ldots, \sigma_m, x_{m+1}, \ldots, x_n), \quad (13.1)$$

где дизъюнкция берется по всевозможным 2^m наборам значений переменных $x_1, x_2, \ldots, x_m$.

Для доказательства теоремы подставим в обе части равенства (13.1) произвольный набор $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ значений всех n переменных. Заметим, что $x^\sigma = 1$ только при $x = \sigma$ и из всех 2^m конъюнкций $x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_m^{\sigma_m}$ правой части формулы (13.1) значение 1 примет единственная конъюнкция, а именно та, для которой $\sigma_1 = \alpha_1$, $\sigma_2 = \alpha_2, \ldots, \sigma_m = \alpha_m$. Остальные конъюнкции будут равны 0. Отсюда получим тождество

$$f(\alpha_1, \alpha_2, \ldots, \alpha_n) = \alpha_1^{\alpha_1} \alpha_2^{\alpha_2} \ldots \alpha_m^{\alpha_m} f(\alpha_1, \alpha_2, \ldots, \alpha_m, \alpha_{m+1}, \ldots, \alpha_n).$$

Представление (13.1) называется *дизъюнктивным разложением функции* $f(x_1, x_2, \ldots, x_n)$ по переменным $x_1, x_2, \ldots, x_m$. Получаемые в результате подстановки констант $\alpha_1, \alpha_2, \ldots, \alpha_m$ вместо переменных $x_1, x_2, \ldots, x_m$ функции

$f(\alpha_1, \alpha_2, \ldots, \alpha_m, x_{m+1}, \ldots, x_n)$, являющиеся коэффициентами разложения, не зависят от переменных $x_1, x_2, \ldots, x_m$.

Из теоремы Шеннона непосредственно вытекают два следующих утверждения, соответствующие двум крайним значениям числа m: $m = 1$ и $m = n$.

Любая булева функция $f(x_1, x_2, \ldots, x_n)$ при любом $i = 1, 2, \ldots, n$ может быть представлена в следующем виде:

$$f(x_1, x_2, \ldots, x_n) = x_i f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \vee \bar{x}_i f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n).$$

Любая булева функция $f(x_1, x_2, \ldots, x_n)$ может быть представлена в следующем виде:

$$f(x_1, x_2, \ldots, x_n) = \bigvee_{\sigma_1, \sigma_2, \ldots, \sigma_n} x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_n^{\sigma_n} f(\sigma_1, \sigma_2, \ldots, \sigma_n). \quad (13.2)$$

Последнее выражение является представлением булевой функции $f(x_1, x_2, \ldots, x_n)$ в *совершенной дизъюнктивной нормальной форме* (СДНФ). Здесь $f(\sigma_1, \sigma_2, \ldots, \sigma_n)$ является значением функции на наборе значений аргументов $(\sigma_1, \sigma_2, \ldots, \sigma_n)$, т. е. константой 0 или 1.

Легко построить СДНФ, представляющую произвольную булеву функцию, заданную в табличной форме. Для этого достаточно выделить наборы $(\sigma_1, \sigma_2, \ldots, \sigma_n)$, на которых функция принимает значение 1, и для каждого из них ввести в СДНФ полную элементарную конъюнкцию, где любая переменная x_i присутствует с отрицанием, если $\sigma_i = 0$, и без отрицания, если $\sigma_i = 1$.

Очевидно, для любой булевой функции $f(x_1, x_2, \ldots, x_n)$, кроме константы 0, существует единственная СДНФ (с точностью до порядка литералов и конъюнкций). Поэтому данная форма представления булевой функции является *канонической*. Например, СДНФ для функции от трех аргументов, заданной табл. 13.1, имеет следующий вид:

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

Таблица 13.1

Задание функции $f(x, y, z)$

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

Константа 1 представляется в виде СДНФ, которая содержит все различные полные элементарные конъюнкции, которые называют *конституентами единицы* (в литературе используется также термин *минтерм*). Конституента единицы принимает значение 1 на единственном наборе значений переменных.

13.3. Конъюнктивные нормальные формы

Элементарной дизъюнкцией D_i является многоместная конъюнкция попарно различных литералов, т. е. $D_i = x_{i_1}^{\sigma_1} \vee x_{i_2}^{\sigma_2} \vee ... \vee x_{i_r}^{\sigma_r}$. К элементарным дизъюнкциям относятся также одиночные литералы и константа 0 – дизъюнкция, состоящая из пустого множества литералов. Число литералов r элементарной дизъюнкции называется ее *рангом*. Элементарная дизъюнкция называется *полной* относительно переменных $x_1, x_2, \ldots, x_n$, если она содержит символы всех переменных (со знаком отрицания или без него). Ранг таких дизъюнкций равен n.

Конъюнктивная нормальная форма (КНФ) – это выражение вида $\bigwedge_{i=1}^{m} D_i$, т. е. конъюнкция элементарных дизъюнкций. Примером конъюнктивной нормальной формы является выражение $(x_2 \vee \bar{x}_3 \vee x_4)(x_1 \vee \bar{x}_2)$. Одна элементарная дизъюнкция также может считаться КНФ.

Согласно принципу двойственности выражение (13.1) можно преобразовать в следующее выражение, которое также справедливо:

$$f(x_1, x_2, \ldots, x_n) = \bigwedge_{\sigma_1, \sigma_2, \ldots, \sigma_m} (x_1^{\bar{\sigma}_1} \vee x_2^{\bar{\sigma}_2} \vee ... \vee x_m^{\bar{\sigma}_m} \vee f(\sigma_1, \sigma_2, \ldots, \sigma_m, x_{m+1}, \ldots, x_n)).$$

Эта формула называется *конъюнктивным разложением функции* $f(x_1, x_2, \ldots, x_n)$ по переменным $x_1, x_2, \ldots, x_m$. Справедливость ее может быть доказана так же, как справедливость формулы (13.1). Так же крайними случаями конъюнктивного разложения являются разложение по одной переменной и по всем переменным. Последнее называется *совершенной конъюнктивной нормальной формой* (СКНФ) и имеет вид

$$f(x_1, x_2, \ldots, x_n) = \bigwedge_{\sigma_1, \sigma_2, \ldots, \sigma_n} (x_1^{\bar{\sigma}_1} \vee x_2^{\bar{\sigma}_2} \vee ... \vee x_n^{\bar{\sigma}_n} \vee f(\sigma_1, \sigma_2, \ldots, \sigma_n)).$$

СКНФ, представляющую произвольную булеву функцию, так же, как ее СДНФ, легко построить по табличному заданию этой функции. Согласно формуле достаточно выделить наборы $(\sigma_1, \sigma_2, \ldots, \sigma_n)$, на которых функция принимает значение 0 (если $f(\sigma_1, \sigma_2, \ldots, \sigma_n) = 1$, то весь сомножитель $(x_1^{\bar{\sigma}_1} \vee x_2^{\bar{\sigma}_2} \vee ... \vee x_n^{\bar{\sigma}_n} \vee 1)$ обращается в 1), и для каждого из них ввести в СДНФ полную элементарную дизъюнкцию, где любая переменная x_i присутствует с отрицанием, если $\sigma_i = 1$, и без отрицания, если $\sigma_i = 0$.

Очевидно, для любой булевой функции $f(x_1, x_2, \ldots, x_n)$, кроме константы 1, существует единственная СКНФ (с точностью до порядка литералов и дизъюнкций). Так же, как СДНФ, эта форма представления булевой функции является *канонической*. СКНФ для функции, которую задает табл. 13.1, имеет следующий вид:

$$(x_1 \vee x_2 \vee x_3)(x_1 \vee x_2 \vee \bar{x}_3)(x_1 \vee \bar{x}_2 \vee \bar{x}_3)(\bar{x}_1 \vee \bar{x}_2 \vee x_3)(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3).$$

Константа 0 представляется в виде СКНФ, которая содержит все различные полные элементарные дизъюнкции, которые называют *конституентами нуля* (в литературе используется также термин *макстерм*). Конституента нуля принимает значение 0 на единственном наборе значений переменных.

Г л а в а 14

Графическое представление булева пространства и булевых функций

14.1. Булев гиперкуб

Булево пространство M можно представить в виде графа, вершины которого соответствуют элементам пространства, а ребра представляют *отношение соседства* между элементами пространства. Два вектора являются соседними, если они отличаются друг от друга значением только одной компоненты. Например, векторы (1 0 0 1) и (1 1 0 1), значения одноименных компонент которых, кроме одной второй компоненты, совпадают, являются соседними. Данный граф, представляющий n-мерное булево пространство, имеет 2^n вершин и $n2^{n-1}$ ребер. Он называется *полным булевым графом*, или *n-мерным гиперкубом*. Рассмотрим построение такого гиперкуба для различных значений размерности пространства.

Одномерный гиперкуб состоит из двух вершин, связанных ребром. Одной из этих вершин приписывается константа 0, другой – константа 1, которые являются кодами данных вершин. Чтобы получить двумерный гиперкуб, надо продублировать одномерный гиперкуб и каждую вершину исходного гиперкуба соединить ребром с ее дублем. Коды вершин построенного двумерного гиперкуба получаются добавлением нулей справа к кодам вершин исходного гиперкуба и единиц – к кодам дублей вершин. Аналогично получаются трехмерный гиперкуб, четырехмерный гиперкуб и т. д. Последовательность гиперкубов от одномерного до четырехмерного представлена на рис. 14.1.

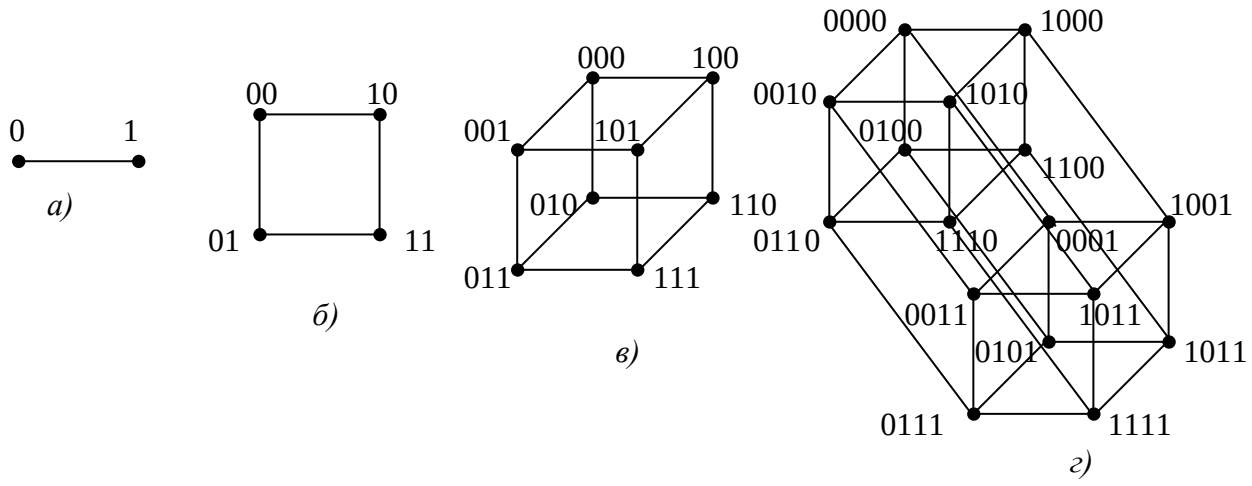


Рис.14.1. Графическое представление булева пространства: *а)* одномерное; *б)* двумерное; *в)* трехмерное; *г)* четырехмерное

Сформулируем общее правило увеличения размерности гиперкуба: для перехода от m-мерного гиперкуба к $(m + 1)$-мерному надо исходный m-мерный

гиперкуб продублировать и каждую вершину исходного гиперкуба соединить ребром с ее дублем. В полученном гиперкубе к кодам вершин исходного m-мерного гиперкуба добавляются справа нули, а к кодам их дублей – единицы.

В гиперкубе выделяются гиперграни, которые являются порожденными подграфами, представляющими собой гиперкубы меньшей размерности, чем рассматриваемый гиперкуб. Это может быть отдельное ребро, двумерная грань, трехмерный куб и т. п. Подграф, представляющий гиперграни, порождается множеством вершин, составляющих интервал булева пространства.

14.2. Представление булевых функций на гиперкубе

Любой интервал булева пространства является характеристическим множеством функции, выражаемой в алгебраической форме одной элементарной конъюнкцией. Например, конъюнкции $x_1 \bar{x}_3 x_4$ соответствует интервал четырехмерного пространства, представляемый троичным вектором (1 – 0 1). Интервалу приписывается ранг той конъюнкции, которую он представляет.

На гиперкубе булева функция $f(x_1, x_2, \ldots, x_n)$ задается выделением вершин, представляющих элементы ее характеристического множества M_f^1. Например, задание функции $f(x_1, x_2, \ldots, x_n) = \bar{x}_1 \bar{x}_2 x_3 \vee \bar{x}_1 x_2 \bar{x}_3 \vee \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 x_3$ может быть показано светлыми кружками, как на рис. 14.2.

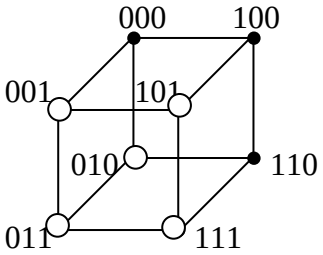


Рис.14.2. Трехмерный гиперкуб с заданной на нем булевой функцией

В изображенном гиперкубе легко заметить две гиперграни, составляющие множество M_f^1. Они представляют интервалы, задаваемые троичными векторами (– – 1) и (0 1 –), которые являются характеристическими множествами элементарных конъюнкций x_3 и $\bar{x}_1 x_2$ соответственно. Поэтому рассматриваемую функцию можно задать как $f(x_1, x_2, x_3) = x_3 \vee \bar{x}_1 x_2$. Выполнение простого склеивания над исходной формулой дает тот же результат. Таким образом, графическое представление булевой функции дает возможность непосредственно получить ее задание в виде компактной формулы.

Справедливость формул, выведенных в гл. 12, наглядно демонстрируется на гиперкубе. На рис. 14.3*а* видно, что ребра, представляемые векторами (0 1 –) и (1 1 –), образуют гиперграни более высокой размерности, представляемую вектором (– 1 –). Это соответствует простому склеиванию: $\bar{x}_1 x_2 \vee x_1 x = x_2$.

На рис. 14.3*б* видно, что интервал, соответствующий конъюнкции $x_1 x_2$, поглощает элемент булева пространства, соответствующий конъюнкции $x_1 x_2 \bar{x}_3$. Тем самым демонстрируется простое поглощение, выражаемое формулой $x_1 x_2 \vee x_1 x_2 \bar{x}_3 = x_1 x_2$.

Рис. 14.3*в* демонстрирует формулу обобщенного склеивания: $\bar{x}_1 \bar{x}_2 \vee x_1 \bar{x}_3 = \bar{x}_1 \bar{x}_2 \vee x_1 \bar{x}_3 \vee \bar{x}_2 \bar{x}_3$. Продукт обобщенного склеивания (0 – 0) поглощается совокупностью интервалов (0 0 –) и (– 1 0).

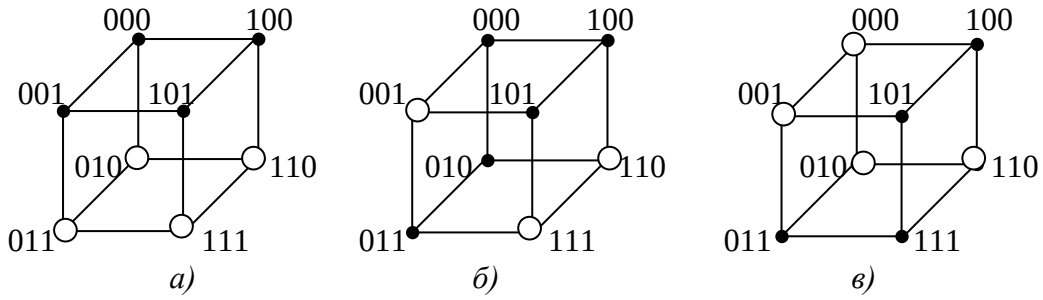


Рис.14.3. Графическое представление некоторых формул булевой алгебры: *а)* простое склеивание; *б)* простое поглощение; *в)* обобщенное склеивание

14.3. Развертка гиперкуба на плоскости. Карта Карно

Как видно из рис. 14.1, с увеличением размерности булева пространства его графическое представление в виде гиперкуба быстро становится трудным для восприятия. Более удобным является развертка гиперкуба на плоскости. Пример такой развертки трехмерного гиперкуба, которая получена удалением двух ребер и расположением верхних и нижних ребер в две параллельных линии, показан на рис. 14.4. На этом же рисунке представлена также функция $f(x_1, x_2, x_3) = x_1 x_2 \vee \bar{x}_1 \bar{x}_2 x_3$.

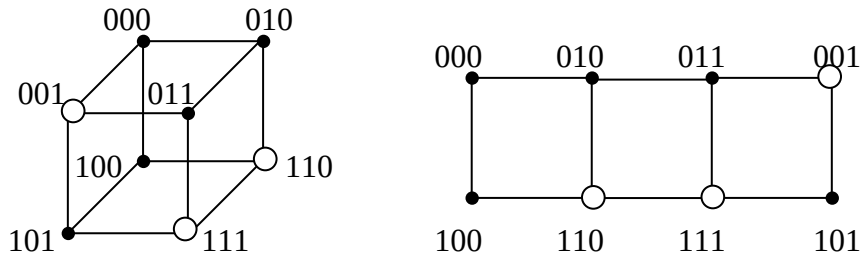


Рис.14.4. Трехмерный гиперкуб и его развертка на плоскости

Еще более удобной формой представления булева пространства является двумерная таблица, которую принято называть *картой Карно*. Карта Карно имеет ту же структуру, что и развертка гиперкуба на плоскости. Каждая ее клетка соответствует элементу булева пространства. Булеву функцию можно

задать расположением нулей и единиц в клетках в соответствии с теми значениями, которые принимает функция на соответствующих элементах булева пространства. Другим способом задания булевой функции на карте Карно является разметка клеток, соответствующих элементам множества M_f^1. При этом клетки, соответствующие элементам множества M_f^0, остаются пустыми. Оба способа представлены на рис. 14.5, где показан пример задания функции $f(x_1, x_2, x_3) = x_1 x_2 \vee \bar{x}_1 \bar{x}_2 x_3$. Коды строк и столбцов, из которых составляются коды клеток, представлены отрезками прямых. Отрезок вертикальной прямой возле нижней строки показывает, что переменная x_1 в коде этой строки имеет значение 1, а его отсутствие у верхней строки говорит, что в ее коде $x_1 = 0$. Аналогично горизонтальные отрезки показывают значения переменных x_2 и x_3 в кодах столбцов.

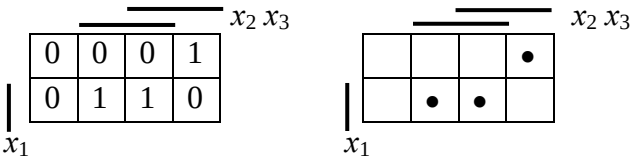


Рис. 14.5. Задание булевой функции с помощью карты Карно

Удобство работы с картами Карно обеспечивается применением *кода Грея* для кодирования ее строк и столбцов.

Пусть надо закодировать в коде Грея последовательность некоторых объектов, число которых N. Коды этих объектов, так же как и коды в виде двоичных чисел, являются булевыми векторами. Длина кода n должна быть такой, чтобы выполнялось $N \le 2^n$, или $n = \lceil \log_2 N \rceil$, где $\lceil a \rceil$ – ближайшее к a сверху целое число. Первому объекту присваивается код, состоящий только из нулей – 0 0 … 0. Далее коды определяются по следующему правилу.

Для получения следующего кода берется последний код и в нем меняется значение той самой правой компоненты, изменение значения которой приводит к новому коду.

Коды соседних в последовательности объектов оказываются, таким образом, отличающимися только значением одной компоненты.

Другой способ построения кода Грея заключается в следующем. Сначала берется последовательность из двух однокомпонентных кодов: (0), (1). Приписав к этой последовательности те же коды в обратном порядке, получим (0), (1), (1), (0). Добавляем слева 0 к элементам исходной последовательности и 1 – к приписанной. Получим (0 0), (0 1), (1 1), (1 0). К этой последовательности приписываем (1 0), (1 1), (0 1), (0 0) и снова добавляем к исходной части слева 0, к приписанной – 1. Получаем (0 0 0), (0 0 1), (0 1 1), (0 1 0), (1 1 0), (1 1 1), (1 0 1), (1 0 0). Эти действия повторяем нужное число раз в зависимости от количества кодируемых объектов.

Благодаря коду Грея, два соседних элемента булева пространства или два соседних интервала расположены на карте Карно симметрично некоторой оси, т. е. отношение соседства элементов булева пространства представляется

отношением симметрии в карте Карно. На рис. 14.6 показана шестимерная карта Карно с осями симметрии. Оси симметрии проходят в местах изменения значений переменных в кодах строк и столбцов. Каждая ось имеет свою *зону симметрии*, ширина которой определяется *рангом* оси.

Оси, связанной с переменной, наиболее часто меняющей свое значение в последовательности кодов строк (или столбцов), придается ранг 1. Если ось связана с переменной, которая меняет свое значение в два раза меньше, ей приписывается ранг 2, если в четыре раза меньше, – ранг 3 и т. д. Ширина оси симметрии ранга k равна 2^k (рис. 14.6).

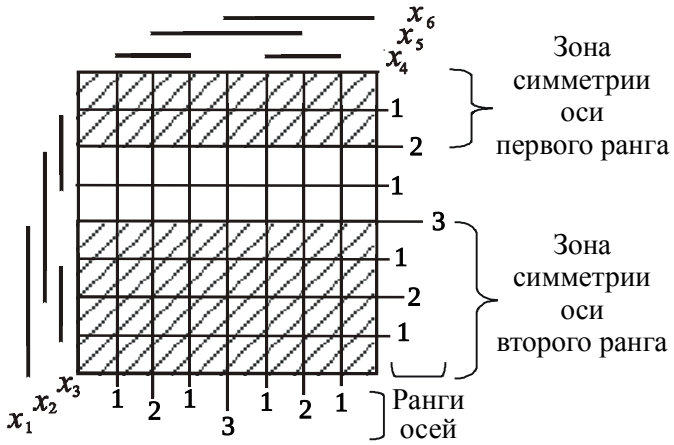


Рис. 14.6. Зоны симметрии карты Карно

По карте Карно легко построить упрощенную ДНФ функции, которая задана с помощью этой карты. Для этого надо выделить интервалы, на которых функция принимает значение 1. На карте Карно они представлены единичными областями, симметричными относительно некоторых осей. Каждый интервал должен быть *максимальным*, т. е. не быть собственным подмножеством другого интервала. Элементарная конъюнкция, соответствующая такому интервалу, не содержит переменных, с которыми связаны данные оси. Если на всем интервале некоторая переменная x имеет значение 0, то она берется с отрицанием, если 1, то без отрицания.

Рекомендуется в первую очередь выделять те максимальные интервалы, где имеется элемент, для которого данный максимальный интервал является единственным, его содержащим. Такие интервалы называются *обязательными*, а соответствующие элементы – *определяющими*.

Если пользоваться «жадным» способом, т. е. стараться покрыть одним интервалом как можно большее число элементов, то в полученной ДНФ может оказаться избыточная элементарная конъюнкция, как, например, для функции $f(x_1\, x_2\, x_3\, x_4) = x_1\, \bar{x}_2\, x_3 \vee x_1\, x_2\, x_4 \vee \bar{x}_1\, x_2\, x_3 \vee \bar{x}_1\, \bar{x}_2\, x_4$, представленной картой Карно на рис. 14.7. Самый большой интервал, представленный конъюнкцией $x_3 x_4$, покрывается остальными интервалами, поэтому является избыточным.

Примером получения упрощенной ДНФ является получение ее для функции, представленной картой Карно на рис. 14.8, где определяющие элементы отмечены кружками. ДНФ этой функции имеет вид

$$x_2\ \bar{x}_3\, x_4\ \bar{x}_6 \vee\ \bar{x}_1\, x_3\ \bar{x}_4\, x_5 \vee x_2\, x_3\ \bar{x}_4\, x_5 \vee x_2\, x_3\, x_5\, x_6 \vee\ \bar{x}_1\, x_3\, x_4\, x_6 \vee\ \bar{x}_1\, x_2\ \bar{x}_3\, x_5\ \bar{x}_6$$

Эта ДНФ является минимальной, поскольку из шести интервалов, покрывающих множество M_f^1, пять являются обязательными.

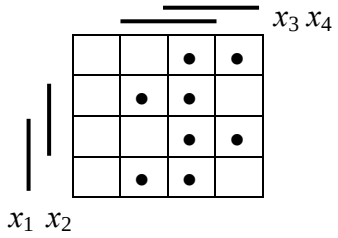


Рис. 14.7. Область M_f^1с избыточным максимальным интервалом

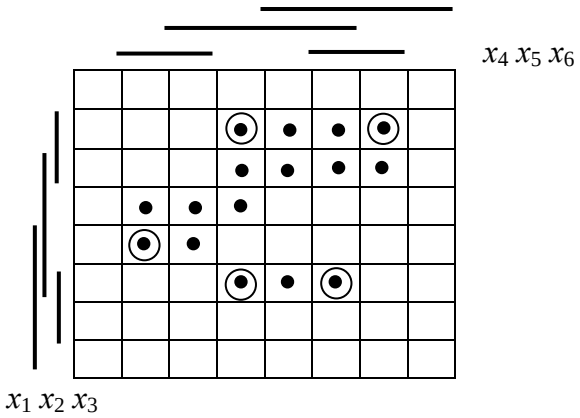


Рис. 14.8. Задание булевой функции с помощью карты Карно.
Кружками отмечены определяющие элементы

Г л а в а 15

Полные системы булевых функций. Реализация функций комбинационными схемами

15.1. Функциональная полнота

Система булевых функций $\{f_1, f_2, \dots, f_m\}$ называется *функционально полной*, или просто *полной*, если любая булева функция может быть представлена в виде суперпозиции этих функций. Полную систему булевых функций называют еще *базисом*.

Минимальным базисом называется такой базис $\{f_1, f_2, \dots, f_m\}$, для которого удаление хотя бы одной из функций $f_1, f_2, \dots, f_m$ превращает его в неполную систему.

Функции от двух переменных, представляемые булевыми операциями $\bar{}$ (отрицание), $\wedge$ (конъюнкция) и $\vee$ (дизъюнкция), образуют полную систему. Действительно, из теоремы Шеннона следует, что любую булеву функцию можно представить в виде совершенной ДНФ, которая представляет собой суперпозицию отрицания, конъюнкции и дизъюнкции.

Базис $\{\bar{}, \wedge, \vee\}$ не является минимальным. Одну из операций, $\wedge$ или $\vee$, из него можно удалить. Пользуясь правилами де Моргана и законом двойного отрицания, можно дизъюнкцию выразить через отрицание и конъюнкцию, а конъюнкцию – через отрицание и дизъюнкцию:

$$a \vee b = \overline{\overline{a \vee b}} = \overline{\bar{a} \wedge \bar{b}};$$

$$a \wedge b = \overline{\overline{a \wedge b}} = \overline{\bar{a} \vee \bar{b}}.$$

Система $\{\wedge, \vee\}$ не является полной, т. к. операцию отрицания нельзя выразить через операции $\wedge$ и $\vee$.

Чтобы убедиться в полноте некоторой системы функций, достаточно через эти функции выразить любую функцию из некоторой известной полной системы. Покажем, что каждая из операций $|$ (штрих Шеффера) и $\uparrow$ (стрелка Пирса) составляет полную систему, использовав для этого базис $\{\bar{}, \wedge\}$:

$$\bar{a} = a \mid a; \qquad a \wedge b = \overline{\overline{a \wedge b}} = \overline{a \mid b} = (a \mid b) \mid (a \mid b);$$

$$\bar{a} = a \uparrow a; \qquad a \wedge b = \overline{\overline{a \wedge b}} = \overline{\bar{a} \vee \bar{b}} = \bar{a} \uparrow \bar{b} = (a \uparrow a) \uparrow (b \uparrow b);$$

Примером полной системы булевых функций является система, содержащая константу 1, а также функции, выражаемые операцией $\wedge$ и операцией $\oplus$ (сложение по модулю два). Действительно,

$$\bar{a} = a \oplus 1;$$
$$a \vee b = \overline{\overline{a \vee b}} = \overline{\bar{a} \wedge \bar{b}} = (a \oplus 1)(b \oplus 1) \oplus 1.$$

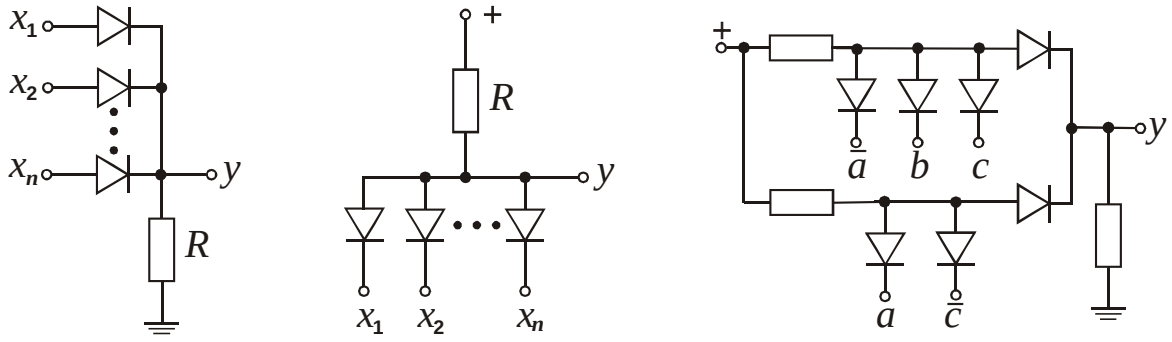
Последнее выражение упрощается с учетом коммутативности и ассоциативности операции $\oplus$ и дистрибутивности операции $\wedge$ относительно $\oplus$, в чем можно убедиться, обратившись к табл. 8.3: $a \vee b = a \oplus b \oplus ab$.

Вопрос о функциональной полноте системы булевых функций имеет практический смысл: набор логических элементов, из которых строятся разнообразные схемы, должен содержать элементы, реализующие все функции из заданного базиса.

15.2. Реализация булевых функций комбинационными схемами

Задачей логического проектирования является построение схемы из логических элементов, реализующей заданное поведение дискретного устройства. Логические элементы, реализующие простейшие булевы функции, строятся на основе полупроводниковой технологии. Для этого используются диоды и транзисторы.

Диодные схемы показаны на рис. 15.1. Константы 0 и 1 представляются соответственно низким и высоким уровнем потенциала. В схеме, реализующей дизъюнкцию (рис. 15.1*а*), высокий потенциал на выходе будет тогда и только тогда, когда высокий потенциал появится хотя бы на одном из входов и потечет ток через сопротивление *R*.



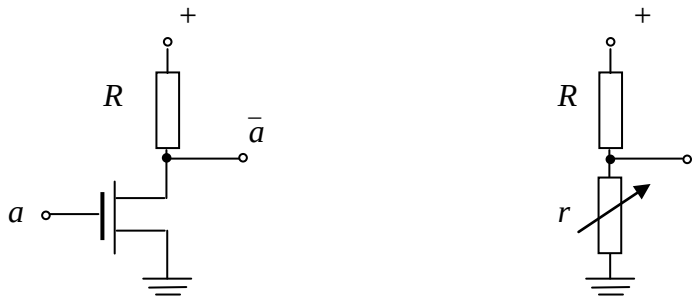
а) *б)* *в)*

Рис. 15.1. Диодные схемы, реализующие: *а)* дизъюнкцию; *б)* конъюнкцию; *в)* дизъюнктивную нормальную форму $\bar{a}\, b\, c \vee a\, \bar{c}$

В схеме, реализующей конъюнкцию (рис. 15.1*б*), потенциал на выходе будет близким к потенциалу источника питания, если на все входы будет подан высокий потенциал. В противном случае через сопротивление *R* пойдет большой ток, создавая падение напряжения, приводящее к низкому потенциалу на выходе.

С помощью одних только диодов и сопротивлений нельзя создать схему, реализующую операцию отрицания. Для реализации произвольной ДНФ достаточно подавать на схему вместе с входными сигналами их инверсии, как показано на рис. 15.1*в*.

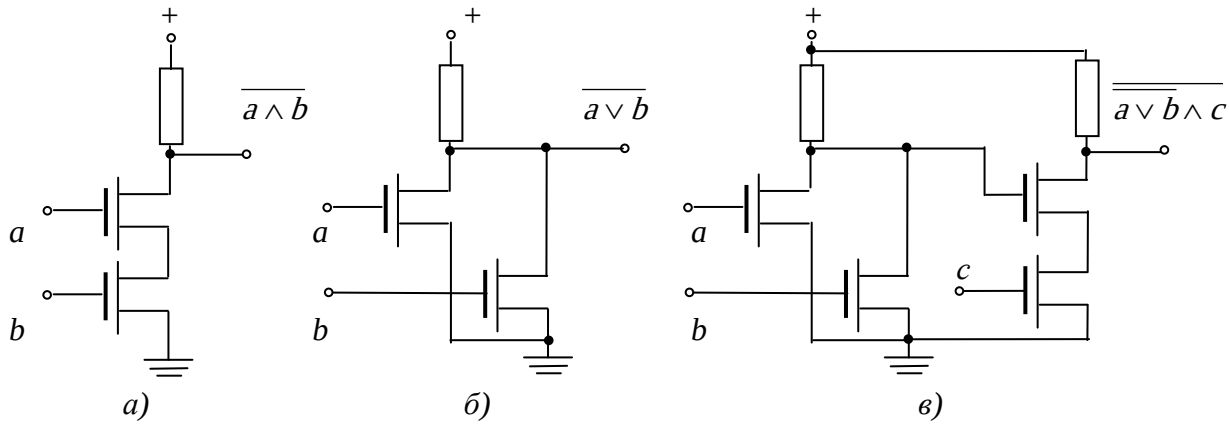
Операцию отрицания можно реализовать с помощью транзистора. На рис. 15.2*а* изображена простейшая схема усилителя-инвертора на транзисторе. В рассматриваемой модели эту схему можно рассматривать как делитель напряжения (рис. 15.2*б*), одно из сопротивлений которого r меняет свое значение под действием управляющего сигнала так, что $R >> r$ при высоком потенциале на входе a и $R << r$ при низком. Таким образом, при высоком потенциале на входе потенциал на выходе близок к потенциалу земли, а при низком – к потенциалу источника питания.



а) *б)*

Рис. 15.2. Реализация операции отрицания: *а)* схема инвертора; *б)* эквивалентная схема

На рис. 15.3 представлены транзисторные схемы, реализующие некоторые булевы функции. В схеме на рис. 15.3*а* низкий потенциал на выходе будет тогда и только тогда, когда на обоих входах будет высокий потенциал, а в схеме на рис. 15.3*б* высокий потенциал на выходе будет тогда и только тогда, когда обоих входах будет низкий потенциал. Эти схемы реализуют соответственно отрицание конъюнкции (штрих Шеффера) и отрицание дизъюнкции (стрелку Пирса). Более сложные функции можно реализовать, соединяя выходы одних схем с входами других, как показано на рис. 15.3*в*.



а) *б)* *в)*

Рис. 15.3. Реализация функций: *а)* $f = \overline{a \wedge b}$; *б)* $f = \overline{a \vee b}$; *в)* $f = \overline{\overline{a \vee b} \wedge c}$.

Г л а в а 16

Троичные векторы и матрицы

16.1. Отношения на множестве троичных векторов. Операции над троичными векторами. Эквивалентность матриц

Ранее было определено понятие троичного вектора. Напомним, что его компоненты принимают значения из множества $\{0, 1, -\}$. Троичный вектор можно рассматривать как множество булевых векторов, получаемых из него подстановкой нулей и единиц вместо знаков «–». Так вектор (0 – 1 0 – 1) задает множество {(0 0 1 0 0 1), (0 0 1 0 1 1), (0 1 1 0 0 1), (0 1 1 0 1 1)}, представляющее интервал булева пространства.

Напомним также, что троичный вектор можно интерпретировать как характеристическое множество элементарной конъюнкции. Например, вектор (0 – 1 0 – 1) представляет конъюнкцию $\bar{x}_1 x_3 \bar{x}_4 x_6$. Тогда всякую троичную матрицу (строками которой являются троичные векторы) можно считать представлением ДНФ некоторой булевой функции.

Определим следующие бинарные отношения на множестве троичных векторов одинаковой размерности.

Ортогональность. Троичные векторы ***u*** и ***v*** ортогональны по *i*-й компоненте, если и только если *i*-я компонента имеет значение 0 в одном из этих векторов и 1 – в другом. Троичные векторы ортогональны, если они ортогональны хотя бы по одной компоненте. Например, векторы (0 – 1 0 – 1) и (0 1 0 – 1 0) ортогональны по третьей и шестой компонентам.

Пересечение. Если векторы ***u*** и ***v*** неортогональны, то они находятся в отношении пересечения. Это понятие согласуется с понятием пересечения множеств: пересекающиеся троичные векторы представляют пересекающиеся интервалы. Примером пересекающихся векторов являются векторы (0 – 1 0 – 1) и (0 0 1 – 1–).

Смежность. Векторы ***u*** и ***v***, ортогональные только по одной компоненте, являются смежными. Соответствующие элементарные конъюнкции тоже смежны. Над ними можно выполнять операцию обобщенного склеивания. Векторы (0 – 1 0 – 1) и (0 1 0 – 1 –) являются смежными, так как они ортогональны только по третьей компоненте.

Соседство. Векторы ***u*** и ***v*** являются соседними, если по некоторой *i*-й компоненте они ортогональны, а значения остальных одноименных компонент совпадают. Такими, например, являются векторы (0 – 1 0 – 1) и (0 – 1 0 – 0).

Поглощение. Вектор ***u*** поглощает вектор ***v***, если и только если все компоненты вектора ***u***, значения которых отличны от «–», совпадают с одноименными компонентами вектора ***v***. Интервал, представляемый вектором ***v***, является подмножеством интервала, представляемого вектором ***u***. Например, вектор (0 – 1 0 – –) поглощает вектор (0 – 1 0 – 0).

Нетрудно видеть, что отношения ортогональности, пересечения, смежности и соседства обладают свойством симметричности, а отношение поглощения транзитивно. Кроме того, отношения пересечения и поглощения рефлексивны.

16.2. Эквивалентность матриц

Троичная матрица $\boldsymbol{U}$ *эквивалентна* булевой матрице $\boldsymbol{W}$, если каждая из строк матрицы $\boldsymbol{W}$ поглощается хотя бы одной строкой матрицы $\boldsymbol{U}$, а любой вектор, не совпадающий ни с одной из строк матрицы $\boldsymbol{W}$, не поглощается ни одной строкой матрицы $\boldsymbol{U}$.

Троичные матрицы $\boldsymbol{U}$ и $\boldsymbol{V}$ *эквивалентны*, если существует булева матрица, эквивалентная обеим матрицам $\boldsymbol{U}$ и $\boldsymbol{V}$. Бинарное отношение эквивалентности матриц рефлексивно, симметрично и транзитивно.

Ясно, что эквивалентные матрицы представляют одну и ту же область булева пространства. Булеву матрицу можно интерпретировать как представление совершенной ДНФ. Две эквивалентные троичные матрицы представляют ДНФ одной и той же булевой функции. Всякая замена троичной матрицы $\boldsymbol{U}$ эквивалентной ей матрицей $\boldsymbol{V}$ называется равносильным преобразованием матрицы $\boldsymbol{U}$. Рассмотрим некоторые простейшие равносильные преобразования троичной матрицы.

Склеивание соседних строк. Две соседние строки можно заменить одной строкой, где значения компонент определяются следующим образом. Компонента, по которой исходные строки ортогональны, приобретает значение «–». Значения остальных компонент совпадают со значениями соответствующих компонент исходных строк. Например,

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

Поглощение. Строка, поглощаемая другой строкой той же матрицы, может быть удалена. Например,

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

Обобщенное склеивание смежных строк. Если в матрице присутствуют две смежных строки, то в эту матрицу можно добавить строку, значения компонент которой определяются следующим образом. Компонента, по которой исходные строки ортогональны, приобретает значение «–». Если хотя бы одна из одноименных компонент смежных строк имеет значение 0 или 1, то

соответствующая компонента новой строки приобретает это же значение. В противном случае она получает значение «–». Например,

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

Разложение строки по i-й компоненте. Строку $\boldsymbol{u}$, имеющую значение «–» в i-й компоненте, можно заменить парой строк, одна из которых получается из $\boldsymbol{u}$ присвоением i-й компоненте значения 0, другая – значения 1. Например,

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

Эта операция является обратной по отношению к операции простого склеивания. Ее последовательное применение по всем компонентам троичного вектора, имеющим значение «–», приводит к множеству булевых векторов, образующих интервал, представляемый данным вектором.

16.3. Анализ троичной матрицы на вырожденность

Троичная матрица $\boldsymbol{U}$ является *вырожденной*, если не существует троичного вектора, ортогонального каждой строке матрицы $\boldsymbol{U}$. Такая матрица представляет совокупность интервалов, покрывающую все булево пространство, и если ее интерпретировать как ДНФ некоторой булевой функции, то эта функция является константой 1.

Поставим задачу следующим образом. Для заданной троичной матрицы $\boldsymbol{U}$ требуется найти троичный вектор $\boldsymbol{v}$, ортогональный каждой ее строке, или убедиться в том, что такого вектора не существует. Вектор $\boldsymbol{v}$ в этом случае представляет набор значений аргументов, обращающий в нуль функцию, задаваемую матрицей $\boldsymbol{U}$.

Троичную матрицу можно рассматривать как сжатую форму булевой матрицы, если считать, что всякий троичный вектор представляет множество булевых векторов, получаемых заменой значений «–» на всевозможные комбинации нулей и единиц. Троичный вектор, имеющий k компонент со значением «–», представляет множество 2^k булевых векторов. Будем говорить, что любой из этих булевых векторов *покрывается* данным троичным вектором. Например, матрица

$$\begin{bmatrix} 1 & - & - \\ - & 1 & 1 \end{bmatrix}$$

является сжатой формой следующей булевой матрицы (заметим, что, если специально не оговорено, рассматриваются матрицы, не имеющие одинаковых строк):

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

Троичный вектор (0, 0, –) ортогонален обеим строкам приведенной троичной матрицы. Он представляет множество из двух булевых векторов, (0, 0, 0) и (0, 0, 1), ни один из которых не является строкой соответствующей булевой матрицы. Очевидно, что если некоторая троичная матрица с n столбцами является вырожденной, то для любого n-компонентного булева вектора в данной матрице имеется покрывающая его строка. Если же существует булев вектор, не покрываемый ни одной строкой троичной матрицы, число столбцов которой равно размерности данного вектора, то данная матрица не вырождена. Следовательно, решить задачу о вырожденности троичной матрицы можно простым перебором всех 2^n различных булевых векторов, сопровождаемым поиском для каждого вектора покрывающей его строки. Однако более эффективным является рассматриваемый ниже *редукционный метод*.

Данный метод опирается на комбинаторный поиск. Текущая ситуация характеризуется двумя переменными величинами: троичным вектором $\boldsymbol{w}$, число компонент которого фиксировано и равно числу столбцов в заданной матрице $\boldsymbol{U}$, и троичной матрицей $\boldsymbol{T}$, значениями которой будут служить некоторые *миноры* матрицы $\boldsymbol{U}$. Под минором матрицы понимается ее часть, образованная заданным подмножеством строк и заданным подмножеством столбцов. Перебор значений вектора $\boldsymbol{w}$ должен привести к искомому вектору $\boldsymbol{v}$, если он существует.

Положим, что в текущей ситуации уже определены значения некоторых компонент вектора $\boldsymbol{w}$, т. е. им приписаны значения 0 или 1, и отыскиваются значения остальных компонент, такие, чтобы вектор $\boldsymbol{w}$ стал ортогональным каждой из строк матрицы $\boldsymbol{T}$ (ее текущего значения), столбцы которой ставятся в соответствие этим компонентам. В начальной ситуации матрица $\boldsymbol{T}$ совпадает с матрицей $\boldsymbol{U}$, а вектор $\boldsymbol{w}$ полностью неопределен, т. е. все его компоненты имеют значение «–».

Очередной шаг заключается в приписывании значения 0 или 1 некоторой компоненте вектора $\boldsymbol{w}$ или в упрощении матрицы $\boldsymbol{T}$ путем удаления некоторых строк и столбцов с сохранением обозначений остающихся. Каждый раз в матрице $\boldsymbol{T}$ остаются только те строки, которые еще не являются

ортогональными вектору $\boldsymbol{w}$, и столбцы, которые соответствуют некоторым компонентам вектора $\boldsymbol{w}$. Это те компоненты, которые можно использовать для обеспечения ортогональности вектора $\boldsymbol{w}$ данным строкам. Перед выполнением очередного шага, если позволяют условия, текущая ситуация упрощается по следующим *правилам редукции*.

Правило 1. Из матрицы $\boldsymbol{T}$ удаляются столбцы, не содержащие ни значений 0, ни значений 1. (Какое бы значение ни приписывалось компонентам вектора $\boldsymbol{w}$, соответствующим таким столбцам, ни одна из строк матрицы $\boldsymbol{T}$ не будет ортогональной вектору $\boldsymbol{w}$ по этим компонентам.)

Правило 2. Из матрицы $\boldsymbol{T}$ удаляются строки, ортогональные вектору $\boldsymbol{w}$, а затем столбцы, которым соответствуют компоненты вектора $\boldsymbol{w}$ со значением 0 или 1.

Правило 3. Если в матрице $\boldsymbol{T}$ имеется строка, где лишь одна компонента обладает значением, отличным от «–», то соответствующей компоненте вектора $\boldsymbol{w}$ приписывается инверсное значение. (Только таким образом можно обеспечить в текущей ситуации ортогональность данной строки вектору $\boldsymbol{w}$.)

Правило 4. Если в матрице $\boldsymbol{T}$ существует столбец, не содержащий значения 0 (или 1), то это значение приписывается соответствующей компоненте вектора $\boldsymbol{w}$. (Если в столбце присутствуют как нули, так и единицы, то, приписывая соответствующей компоненте какое-то из этих значений, мы делаем одни строки ортогональными вектору $\boldsymbol{w}$ и теряем возможность использовать данную компоненту для обеспечения ортогональности других строк. Такой потери не происходит, когда выполняется данное правило при указанном условии. Текущая ситуация при этом упрощается.)

Когда редуцирование становится невозможным, производится расщепление текущей ситуации.

Правило расщепления предписывает перебор значений 0 и 1 некоторой компоненты вектора $\boldsymbol{w}$. При этом рекомендуется выбирать такую компоненту, которая соответствует максимально определенному столбцу матрицы $\boldsymbol{T}$, т. е. столбцу, имеющему минимальное число значений «–».

Правило нахождения решения. Если непосредственно после удаления некоторой строки из матрицы $\boldsymbol{T}$ по правилу 2 матрица становится пустой, текущее значение вектора $\boldsymbol{w}$ представляет искомое решение $\boldsymbol{v}$.

Правило возврата. Если матрица $\boldsymbol{T}$ становится пустой непосредственно после удаления некоторого столбца или если она содержит строку без значений 0 и 1, то на данной ветви дерева поиска вектор $\boldsymbol{v}$ найти невозможно и следует продолжить обход дерева поиска, возвратившись к последней из точек ветвления с незавершенным перебором.

Правило прекращения поиска. Если при полном обходе дерева поиска вектор $\boldsymbol{v}$ найти не удалось, то это свидетельствует о вырожденности матрицы $\boldsymbol{U}$.

Рассмотрим для примера следующую троичную матрицу, столбцы которой для удобства обозначим теми же буквами *a, b, c, d, e, f*, что и соответствующие им компоненты вектора $\boldsymbol{w} = (a, b, c, d, e, f)$:

$$\begin{array}{c} a\ \ b\ \ c\ \ d\ \ e\ \ f \\ \left[\begin{array}{cccccc} 1 & – & – & – & – & 0 \\ 1 & – & 1 & – & 1 & – \\ 1 & 0 & – & – & 1 & 1 \\ 1 & – & – & – & 0 & 1 \\ 1 & 1 & 0 & – & – & 1 \\ 0 & 1 & – & – & – & 1 \\ 0 & – & – & – & 0 & 1 \\ 0 & 0 & – & – & 1 & – \\ 0 & – & – & 0 & 1 & 0 \\ 0 & 1 & – & 1 & 1 & 0 \\ 0 & 1 & – & – & 0 & 0 \\ 0 & – & 1 & – & 0 & 0 \\ 0 & 0 & 0 & – & 0 & 0 \end{array}\right] \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \end{array}$$

Начальная ситуация характеризуется матрицей ***T***, совпадающей с исходной матрицей и вектором ***w*** = (– – – – – –). Непосредственное сокращение матрицы ***T*** невозможно, поскольку не выполняются условия применения правил редукции. Поэтому воспользуемся правилом расщепления и образуем точку ветвления процесса поиска вектора ***v***, соответствующую выбору значения компоненты *a* вектора ***w***. Положим *a* = 1. Тогда, согласно правилам 2 и 1, определение остальных значений компонент вектора ***w*** сведется к поиску вектора, ортогонального матрице

$$\boldsymbol{T} = \begin{array}{c} b\ \ c\ \ e\ \ f \\ \left[\begin{array}{cccc} – & – & – & 0 \\ – & 1 & 1 & – \\ 0 & – & 1 & 1 \\ – & – & 0 & 1 \\ 1 & 0 & – & 1 \end{array}\right] \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}.$$

Обратив внимание на строку 1 и применяя правило 3, припишем компоненте *f* вектора ***w*** значение 1, после чего матрица сокращается по правилу 2 до следующего вида:

$$\boldsymbol{T} = \begin{array}{c} \begin{matrix} b & c & e \end{matrix} \\ \begin{bmatrix} - & 1 & 1 \\ 0 & - & 1 \\ - & - & 0 \\ 1 & 0 & - \end{bmatrix} \end{array} \begin{matrix} \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}.$$

Далее опять применяется правило 3, компонента e получает значение 1, т. е. теперь $\boldsymbol{w} = (1 - - - 1\ 1)$, и матрица $\boldsymbol{T}$ сокращается до

$$\boldsymbol{T} = \begin{array}{c} \begin{matrix} b & c \end{matrix} \\ \begin{bmatrix} - & 1 \\ 0 & - \\ 1 & 0 \end{bmatrix} \end{array} \begin{matrix} \\ 2 \\ 3 \\ 5 \end{matrix}.$$

Согласно правилу 3 полагаем $c = 0$ и $b = 1$, т. е. $\boldsymbol{w} = (1\ 1\ 0 - 1\ 1)$. Далее срабатывает правило возврата, поскольку матрица $\boldsymbol{T}$ становится пустой после удаления столбцов. Это означает, что, направляясь в дереве поиска по ветви, соответствующей $a = 1$, не получаем искомого вектора $\boldsymbol{v}$. При присвоении всех возможных значений оставшимся компонентам строка 5 остается не ортогональной вектору $\boldsymbol{w}$.

Возвратившись к точке ветвления, полагаем теперь $a = 0$. Последующее редуцирование приводит к следующему значению переменной матрицы $\boldsymbol{T}$:

$$\boldsymbol{T} = \begin{array}{c} \begin{matrix} b & c & d & e & f \end{matrix} \\ \begin{bmatrix} 1 & - & - & - & 1 \\ - & - & - & 0 & 1 \\ 0 & - & - & 1 & - \\ - & - & 0 & 1 & 0 \\ 1 & - & 1 & 1 & 0 \\ 1 & - & - & 0 & 0 \\ - & 1 & - & 0 & 0 \\ 0 & 0 & - & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \end{matrix}.$$

Поскольку дальнейшее редуцирование невозможно, применяем правило расщепления. Выберем компоненту e и положим $e = 1$. Матрица $\boldsymbol{T}$ сокращается до

$$\boldsymbol{T} = \begin{matrix} b & d & f & \\ \begin{bmatrix} 1 & - & 1 \\ 0 & - & - \\ - & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} & & & \begin{matrix} 6 \\ 8 \\ 9 \\ 10 \end{matrix} \end{matrix}.$$

Далее по правилу 3 компонента b получает значение 1 и матрица $\boldsymbol{T}$ сокращается до

$$\boldsymbol{T} = \begin{matrix} d & f & \\ \begin{bmatrix} - & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} & & \begin{matrix} 6 \\ 9 \\ 10 \end{matrix} \end{matrix}.$$

Затем следует выбор значения 0 для компоненты f и получение остатка

$$\boldsymbol{T} = \begin{matrix} d & \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \begin{matrix} 9 \\ 10 \end{matrix} \end{matrix},$$

который оказывается вырожденным: компонента d должна получить одновременно значения 0 и 1, что невозможно. Опять срабатывает правило возврата.

Рассмотрим теперь оставшийся вариант, положив $e = 0$. Здесь необходимо найти вектор, ортогональный матрице

$$\boldsymbol{T} = \begin{matrix} b & c & f & \\ \begin{bmatrix} 1 & - & 1 \\ - & - & 1 \\ 1 & - & 0 \\ - & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} & & & \begin{matrix} 6 \\ 7 \\ 11 \\ 12 \\ 13 \end{matrix} \end{matrix}.$$

В соответствии с правилом 3 последовательно выбираются значения $f = 0$, $b = 0$, $c = 0$, после чего становится очевидным, что нельзя сделать вектор $\boldsymbol{w}$ ортогональным строке 13.

Перебор значений вектора $\boldsymbol{w}$ завершен и установлено, что вектора $\boldsymbol{v}$, ортогонального всем строкам матрицы $\boldsymbol{U}$, не существует, т. е. матрица $\boldsymbol{U}$ оказывается вырожденной.

Дерево поиска, соответствующее описанному процессу, изображено на рис. 16.1, где вершины обозначены символами компонент вектора $\boldsymbol{w}$, а дуги – их значениями.

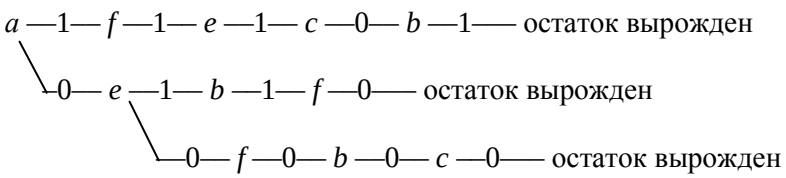


Рис. 16.1. Дерево поиска ортогонального вектора

Г л а в а 17

Локальные упрощения ДНФ

Дизъюнктивная нормальная форма *безызбыточна*, если из нее нельзя удалить ни одной элементарной конъюнкции и ни одного литерала из какой-либо конъюнкции. Это равносильно тому, что из представляемой данную ДНФ троичной матрицы нельзя удалить ни одну из строк и ни одно из значений 0 или 1 нельзя заменить на «–». Локальные упрощения ДНФ сводятся к поиску и последовательному удалению таких элементарных конъюнкций и литералов до тех пор, пока данная ДНФ не станет безызбыточной. Простейшие случаи подобного сокращения определяются, например, формулами, полученными в гл. 8:

$$A\,x \vee A = A; \quad A\,\bar{x} \vee x = A \vee x; \quad A\,x \vee B\,\bar{x} \vee AB = A\,x \vee B\,\bar{x}.$$

Более сложный случай представляет ДНФ $\bar{x}_1\,\bar{x}_2\,x_3 \vee x_1\,\bar{x}_2\,x_4 \vee x_1\,x_2\,x_3 \vee \bar{x}_1\,x_2\,x_4 \vee x_3\,x_4$, где конъюнкция $x_3\,x_4$ является избыточной. Действительно, если ее заменить на $x_3\,x_4\,1 = x_3\,x_4\,(x_1\,x_2 \vee x_1\,\bar{x}_2 \vee \bar{x}_1\,x_2 \vee \bar{x}_1\,\bar{x}_2)$, а затем раскрыть скобки, то каждая из конъюнкций ранга 4 окажется поглощаемой некоторой из конъюнкций ранга 3, присутствующей в исходной ДНФ.

Исходя из выше сказанного, отметим два вида избыточности:

$$D = k \vee D' = D' \quad \text{и} \quad D = xk \vee D' = k \vee D',$$

где k – элементарная конъюнкция, x – литерал, входящий в элементарную конъюнкцию xk, D – некоторая ДНФ, D' – ДНФ, получаемая из D удалением конъюнкции k.

17.1. Удаление избыточных элементарных конъюнкций

В первом случае элементарная конъюнкция k избыточна, если $k \vee D' = D'$. Это значит, что k и D' находятся в отношении *формальной импликации*, т. е. $k \Rightarrow D'$. Функция g *имплицирует* функцию f, если f имеет значение 1 везде, где имеет значение 1 функция g. В рассматриваемом случае ДНФ D' обращается в единицу при любом наборе значений переменных, обращающем конъюнкцию k в единицу, независимо от того, какие значения принимают переменные, не входящие в k.

Пусть троичная матрица $\boldsymbol{V}$ представляет ДНФ D', а троичный вектор $\boldsymbol{v}$ – элементарную конъюнкцию k. Тогда результатом подстановки в D' значений переменных, обращающих конъюнкцию k в единицу, является минор матрицы $\boldsymbol{V}$, образованный строками, не ортогональными вектору $\boldsymbol{v}$ и столбцами,

соответствующими компонентам вектора $\boldsymbol{v}$, имеющими значение «–». Если этот минор является вырожденной матрицей, т. е. D' тождественно равна единице, то конъюнкция k избыточна. В противном случае вектор, ортогональный всем строкам полученного минора, представляет набор значений переменных, обращающий D' в нуль.

Рассмотрим следующую троичную матрицу и проверим на избыточность ее первую строку:

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & \\ \left[\begin{matrix} 0 & 1 & - & 0 & 1 & - \\ 0 & 1 & 1 & 0 & - & 0 \\ 0 & - & 0 & 0 & 1 & - \\ - & 1 & 1 & - & 1 & 1 \\ - & 1 & 0 & - & - & 1 \\ 1 & - & 1 & - & 0 & - \end{matrix}\right. & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ \left.\begin{matrix} 6 \end{matrix}\right. \end{matrix}.$$

Минор, образованный столбцами x_3 и x_6, где элементы первой строки имеют значение «–», и строками 2, 3, 4 и 5, не ортогональными первой строке, имеет вид

$$\begin{matrix} x_3 & x_6 & \\ \left[\begin{matrix} 1 & 0 \\ 0 & - \\ 1 & 1 \\ 0 & 1 \end{matrix}\right. & \begin{matrix} 2 \\ 3 \\ 4 \\ \left.\begin{matrix} 5 \end{matrix}\right. \end{matrix}.$$

Эта матрица является вырожденной, следовательно, первая строка избыточна. Любой входящий в нее булев вектор принадлежит некоторому интервалу, представляемому какой-либо из строк данной матрицы.

Удалив строку 1, получим матрицу, в которой строка 2 ортогональна всем остальным ее строкам. Это значит, что никакой булев вектор, принадлежащий интервалу, представляемому данной строкой, не принадлежит никакому из других интервалов, представляемых остальными строками. Соответствующий минор является пустой матрицей (с пустым множеством строк). Такая матрица представляет константу 0. Таким образом, строка 2 не является избыточной.

Что касается строки 3, то соответствующий минор является однострочной невырожденной матрицей:

$$\begin{matrix} x_2 & x_6 & \\ \left[\begin{matrix} 1 & 1 \end{matrix}\right. & \left.\begin{matrix} 5 \end{matrix}\right. \end{matrix}.$$

Ортогональным вектором для данной строки является (0 –). Подставив 0 во вторую компоненту строки 3, получим вектор, ортогональный всем строкам матрицы. Строка 3 также является неизбыточной для заданной матрицы.

Выполняя подобные построения над остальными строками, убедимся, что они также не являются избыточными.

17.2. Удаление избыточных литералов

Рассмотрим второй вид избыточности в ДНФ, когда $D = xk \vee D' = k \vee D'$. Здесь избыточным является литерал x. Правую часть этого равенства можно представить следующим образом:

$$k \vee D' = k(x \vee \bar{x}) \vee D' = xk \vee D' \vee \bar{x}k = D \vee \bar{x}k.$$

Отсюда видно, что литерал x в выражении $xk \vee D'$ является избыточным, если конъюнкция $\bar{x}k$ является избыточной в выражении $D \vee \bar{x}k$. Следовательно, задача определения избыточности литерала в ДНФ сводится к предыдущей задаче – задаче определения избыточности элементарной конъюнкции.

Удаление литерала из ДНФ в матричном представлении выражается в замене нуля или единицы в троичной матрице на значение «–». На основании предыдущих рассуждений это можно сделать, если вектор, полученный из строки, содержащей данный нуль или единицу, заменой этого значения на противоположное ему значение (т. е. 0 на 1 или 1 на 0), является избыточным для рассматриваемой матрицы.

Таким образом, для того, чтобы решить вопрос о том, можно ли заменить 0 (или 1) в i-й строке и j-м столбце на значение «–», надо построить минор, образованный столбцами, где i-я строка имеет значения «–», и строками, не ортогональными вектору, полученному из i-й строки заменой нуля (или единицы) в j-м столбце на противоположное значение. Если полученный минор оказался вырожденной матрицей, то такая замена возможна.

Рассмотрим матрицу

$$\begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ \left[\begin{array}{cccccc} 0 & 1 & 1 & 0 & - & 0 \\ 0 & - & 0 & 0 & 1 & - \\ - & 1 & 1 & - & 1 & 1 \\ - & 1 & 0 & - & - & 1 \\ 1 & - & 1 & - & 0 & - \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \end{array}.$$

Чтобы узнать, является ли 0 в строке 1 и столбце x_6 избыточным, построим минор, образованный единственным столбцом x_5, где строка 1 имеет значение «–», и единственной строкой 3, не ортогональной вектору (0 1 1 0 – 1).

Единственный элемент в этом миноре имеет значение 1. Он является невырожденной матрицей. Следовательно, нуль в строке 1 и столбце x_6 нельзя заменить на «–».

Рассмотрим теперь единицу в строке 3 и столбце x_3. Минор, образованный столбцами x_1 и x_4 и строками 2 и 4, не ортогональными вектору (– 1 0 – 1 1), имеет вид

$$\begin{matrix} x_1 & x_4 & \\ \begin{bmatrix} 0 & 0 \\ – & – \end{bmatrix} & & \begin{matrix} 2 \\ 4 \end{matrix} \end{matrix}.$$

Вырожденность этого минора говорит о том, данную единицу можно заменить значением «–». Выполнив такую замену, получим матрицу, эквивалентную исходной матрице:

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & \\ \begin{bmatrix} 0 & 1 & 1 & 0 & – & 0 \\ 0 & – & 0 & 0 & 1 & – \\ – & 1 & – & – & 1 & 1 \\ – & 1 & 0 & – & – & 1 \\ 1 & – & 1 & – & 0 & – \end{bmatrix} & & & & & & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \end{matrix}.$$

Г л а в а 18

Минимизация ДНФ

Задача минимизации ДНФ заключается в нахождении такой ДНФ для заданной булевой функции, которая содержала бы минимальное число элементарных конъюнкций или литералов. В первом случае результат решения называется *кратчайшей* ДНФ, во втором – *минимальной*.

18.1. Метод Квайна-МакКласки

Для описания метода введем некоторые понятия. В предыдущей главе было введено понятие формальной импликации: функция g имплицирует функцию f, т. е. $g \Rightarrow f$, если f имеет значение 1 везде, где это значение имеет g. В этом случае функция g называется *импликантой* функции f. Очевидно, что всякая элементарная конъюнкция, входящая в ДНФ некоторой функции, является импликантой этой функции. Дизъюнкция любого множества импликант является также импликантой.

Простая импликанта – это импликанта в виде элементарной конъюнкции, которая перестает быть импликантой при удалении любого литерала. Заметим, что удаление литералов до пустого множества приводит к конъюнкции, представляющей константу 1. Характеристическим множеством простой импликанты является *максимальный интервал*, т. е. интервал, целиком содержащийся в единичной области M^1 функции f и не являющийся подмножеством другого интервала из M^1.

Дизъюнкция всех простых импликант некоторой булевой функции называется *сокращенной* ДНФ этой функции.

Метод Квайна-МакКласки требует представление заданной булевой функции в виде совершенной ДНФ, т. е. такой ДНФ, каждая конъюнкция которой имеет ранг, равный числу аргументов функции. Процесс минимизации состоит из двух этапов: 1) нахождение множества всех простых импликант заданной функции; 2) выделение из этого множества минимального подмножества, составляющего ДНФ данной функции.

Первый этап выполняется путем применения операции простого склеивания над конъюнкциями. Пусть n – число аргументов заданной функции f. Рассмотрим последовательность множеств $C_0, C_1, \ldots, C_k$, где C_0 – множество конъюнкций ранга n, составляющих совершенную ДНФ функции f, C_i – множество конъюнкций ранга $n - i$, полученных путем склеивания конъюнкций из множества C_{i-1}, и C_k – множество конъюнкций ранга $n - k$, где нет ни одной пары соседних конъюнкций и дальнейшее склеивание невозможно. Если склеиваемым конъюнкциям приписывать некоторую метку, то неотмеченные конъюнкции составят множество всех простых импликант.

Выразим этот процесс через операции простого склеивания над булевыми и троичными векторами. Булева функция при этом задана своим характеристическим множеством M^1 – множеством элементов булева пространства, на которых она имеет значение 1. Удобно при этом сгруппировать исходные булевы векторы в подмножества, состоящие из векторов, имеющих одинаковое число единиц, и упорядочить эти подмножества по возрастанию (или убыванию) числа единиц в векторе. Тогда для каждого вектора соседние с ним векторы будут находиться только в соседних подмножествах в полученной последовательности.

В качестве примера рассмотрим булеву функцию, заданную следующей матрицей:

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \\ \left[\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{matrix}\right. \end{matrix}.$$

Используя для множеств векторов те же обозначения, что использовались для соответствующих конъюнкций, получим следующую последовательность матриц, где склеиваемые строки отмечены знаком «*»:

$$C_0 = \begin{matrix} x_1 & x_2 & x_3 & x_4 \\ \left[\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{matrix}\right. \end{matrix} \begin{matrix} \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{matrix}, \quad C_1 = \begin{matrix} x_1 & x_2 & x_3 & x_4 \\ \left[\begin{matrix} 0 & 0 & - & 0 \\ - & 0 & 0 & 0 \\ 0 & 0 & 1 & - \\ 1 & 0 & 0 & - \\ 0 & - & 1 & 1 \\ - & 0 & 1 & 1 \\ 1 & 0 & - & 1 \\ - & 1 & 1 & 1 \\ 1 & - & 1 & 1 \end{matrix}\right. \end{matrix} \begin{matrix} \\ * \\ \\ \\ * \\ * \\ \\ * \\ * \end{matrix}, \quad C_2 = \begin{matrix} x_1 & x_2 & x_3 & x_4 \\ \left[\begin{matrix} - & - & 1 & 1 \end{matrix}\right. \end{matrix}.$$

Сокращенная ДНФ, которая является результатом выполнения первого этапа минимизации, представится следующей матрицей:

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & - & 0 \\ - & 0 & 0 & 0 \\ 0 & 0 & 1 & - \\ 1 & 0 & 0 & - \\ 1 & 0 & - & 1 \\ - & - & 1 & 1 \end{bmatrix}.$$

Второй этап сводится к задаче кратчайшего покрытия, описанной в гл. 7. В данном случае множество M^1 надо покрыть минимальным числом интервалов, представленных строками троичной матрицы, задающей сокращенную ДНФ.

Продолжая решать наш пример и пользуясь обозначениями из гл. 7, поставим задачу следующим образом. Заданы множество $A = M^1$ и совокупность подмножеств $B_1, B_2, \ldots, B_m$ множества A в виде матриц

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{matrix}$$

и

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & - & 0 \\ - & 0 & 0 & 0 \\ 0 & 0 & 1 & - \\ 1 & 0 & 0 & - \\ 1 & 0 & - & 1 \\ - & - & 1 & 1 \end{bmatrix} \begin{matrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \end{matrix}.$$

Требуется выделить минимум подмножеств B_i, покрывающих все множество A. Перейдя к матричной форме этой задачи, получим следующую матрицу:

$$\begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \end{matrix}.$$

Здесь надо выбрать минимальное количество строк так, чтобы каждый столбец имел единицу хотя бы в одной из них.

Строка B_6 является единственной строкой, которая покрывает столбцы a_6 и a_7. Поэтому ее включаем в решение. Удалив эту строку и покрываемые ею столбцы, получим матрицу

$$\begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_5 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \end{matrix}.$$

Строки B_3 и B_5 удаляются по правилу редукции, как покрываемые строками B_1 и B_4 соответственно. Из оставшихся строк строки B_1 и B_4 покрывают оставшиеся столбцы. Таким образом, искомое кратчайшее покрытие составляют строки B_1, B_4 и B_6, а кратчайшая ДНФ, которая является также минимальной для заданной булевой функции, представляется матрицей

$$\begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix} \\ \begin{bmatrix} 0 & 0 & - & 0 \\ 1 & 0 & 0 & - \\ - & - & 1 & 1 \end{bmatrix} \end{array}.$$

В алгебраической форме это решение имеет вид $\bar{x}_1\,\bar{x}_2\,\bar{x}_4 \vee x_1\,\bar{x}_2\,\bar{x}_3 \vee x_3\,x_4$.

Иногда бывает возможность уменьшить размерность матрицы покрытия, выделив интервалы, входящие в любую безызбыточную ДНФ. Если некоторый элемент булева пространства $m_i \in M^1$ принадлежит лишь одному из максимальных интервалов $U \subseteq M^1$, то очевидно, что любое кратчайшее покрытие множества M^1 максимальными интервалами содержит этот интервал.

Элемент m_i в этом случае назовем *определяющим элементом*, а интервал U – *обязательным интервалом*.

Чтобы определить, является ли некоторый элемент m_i определяющим, достаточно найти в M^1 все соседние с ним элементы, а затем построить содержащий их *минимальный поглощающий интервал*.

Минимальный поглощающий интервал U для элементов $m_1, m_2, \ldots, m_k$ булева пространства M представляется вектором $\boldsymbol{u}$, который получается следующим образом: если i-я компонента во всех векторах $m_1, m_2, \ldots, m_k$ имеет значение 0, то вектор $\boldsymbol{u}$ имеет в этой компоненте 0, если 1, то 1. Если i-я компонента имеет различные значения в этих векторах, то i-я компонента вектора $\boldsymbol{u}$ имеет значение «–».

Например, для элементов булева пространства (0 0 1 0 1 0), (0 0 1 0 1 1) и (0 0 0 0 1 1) минимальный поглощающий интервал представляется вектором (0 0 – 0 1 –).

Если все элементы полученного таким образом интервала U принадлежат M^1, то он является максимальным в M^1 и притом обязательным, а m_i является определяющим элементом. В противном случае U не содержится целиком в M^1, а m_i не является определяющим ни для какого интервала.

Чтобы определить, содержится ли интервал U во множестве M^1, достаточно для матрицы, представляющей множество M^1, построить минор, определяемый столбцами, где вектор $\boldsymbol{u}$ имеет значение «–», и строками, не ортогональными вектору $\boldsymbol{u}$. Число строк в этом миноре не превышает 2^p, где p – число компонент вектора $\boldsymbol{u}$, имеющих значение «–». Очевидно, интервал U целиком содержится в M^1 тогда и только тогда, когда число строк в этом миноре равно 2^p.

Обратимся к рассмотренному выше примеру. Элемент (0 0 0 0) не является определяющим. Действительно, минимальный поглощающий интервал для него и соседних с ним элементов (0 0 1 0) и (1 0 0 0) задается троичным вектором (– 0 – 0). Соответствующий минор матрицы, представляющей M^1, имеет вид

$$\begin{matrix} x_1 & x_3 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix},$$

где число строк меньше чем $4 = 2^2$.

Аналогичным образом устанавливается, что элементы (0 0 1 0), (1 0 0 0), (0 0 1 1) и (1 0 0 1) также не являются определяющими. Что касается элемента (0 1 1 1), то для него и соседних с ним элементов (0 0 1 1) и (1 1 1 1) соответствующим троичным вектором является (– – 1 1) и соответствующим минором –

$$\begin{matrix} x_1 & x_2 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix},$$

число строк которого равно 4. Следовательно, элемент (0 1 1 1) является определяющим, а интервал, представляемый вектором (– – 1 1), – обязательным.

Этот интервал включается в решение, и все покрываемые им элементы исключаются из рассмотрения. Очевидно, что если среди них имеется какой-либо определяющий элемент, то он определяет тот же самый интервал. Затем отыскиваются интервалы, содержащие непокрытые элементы. Для этих элементов и интервалов решается задача покрытия, представляемая матрицей

$$\begin{matrix} a_1 & a_2 & a_3 & a_5 & \\ \end{matrix}$$
$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \end{matrix}.$$

Дальнейший ход решения не отличается от предыдущего, и в итоге получается тот же результат: $\bar{x}_1\ \bar{x}_2\ \bar{x}_4 \vee x_1\ \bar{x}_2\ \bar{x}_3 \vee x_3\, x_4$.

18.2. Метод Блейка-Порецкого

Метод Квайна-МакКласки, требующий представление исходной булевой функции в совершенной ДНФ, может оказаться не очень удобным, если булева функция задана в произвольной ДНФ, а ее совершенная ДНФ является довольно громоздкой. Например, совершенная ДНФ функции, заданной произвольной ДНФ $x_1\, x_2\, x_3\, x_5 \vee x_2\, x_3\, x_4\, x_5 \vee \bar{x}_1$, имеет 18 элементарных конъюнкций. В то же время минимальную ДНФ легко получить, применив операции обобщенного склеивания и поглощения:

$$x_1\, x_2\, x_3\, x_5 \vee x_2\, x_3\, x_4\, x_5 \vee \bar{x}_1 = x_1\, x_2\, x_3\, x_5 \vee x_2\, x_3\, x_4\, x_5 \vee \bar{x}_1 \vee x_2\, x_3\, x_5 = \bar{x}_1 \vee x_2\, x_3\, x_5.$$

Метод Блейка-Порецкого основан на применении операций обобщенного склеивания и простого поглощения, определяемых формулами

$$A\,x \vee B\,\bar{x} = A\,x \vee B\,\bar{x} \vee A\,B \quad \text{и} \quad A \vee A\,B = A.$$

Чтобы получить сокращенную ДНФ, надо для всех пар смежных элементарных конъюнкций получить продукты обобщенного склеивания и проверить, не поглощаются ли они другими конъюнкциями, входящими в ДНФ, и не поглощают ли они сами некоторые конъюнкции в ДНФ. Поглощаемые конъюнкции удаляются. Вновь получаемые конъюнкции также участвуют в операциях обобщенного склеивания. Процесс заканчивается, когда не удается ввести в ДНФ новую конъюнкцию.

У т в е р ж д е н и е 18.1. В результате описанного процесса получаются все простые импликанты, т. е. получается сокращенная ДНФ.

Покажем это следующим образом. Пусть L – множество конъюнкций заданной ДНФ, L_1 – результат преобразования L путем всевозможных обобщенных склеиваний и поглощений конъюнкций из L, L_2 – результат такого же преобразования множества L_1 и т. д. Получаем последовательность $L_0, L_1, \ldots, L_m$, где $L_0 = L$, а L_m таково, что $L_{m+1} = L_m$.

Пусть $C_0, C_1, \ldots, C_k$ – последовательность множеств конъюнкций рангов $n, n-1, \ldots, n-k$, полученная в результате выполнения первого этапа метода Квайна-МакКласки. Очевидно, что для каждой конъюнкции $k_0 \in C_0$ найдется такая конъюнкция $l_0 \in L_0$, что $k_0 \Rightarrow l_0$. Теперь покажем, что если для каждой $k_i \in C_i$ найдется $l_j \in L_j$ такая, что $k_i \Rightarrow l_j$, то для каждой $k_{i+1} \in C_{i+1}$ найдется $l_{j+1} \in L_{j+1}$ такая, что $k_{i+1} \Rightarrow l_{j+1}$. (Такой метод доказательства называется методом полной математической индукции). Если мы это докажем, то докажем, что для всякой конъюнкции из любого C_i найдется имплицируемая ею конъюнкция в L_m. Отметим, что если $k \Rightarrow l$, то k содержит все литералы, которые имеются в l. Например, $x_2\,\bar{x}_4\,x_5\,x_6 \Rightarrow \bar{x}_4\,x_6$.

Может оказаться, что k_{i+1} является импликантой для некоторой $l_j \in L_j$. Тогда очевидно, что и в L_{j+1} будет конъюнкция, для которой k_{i+1} является импликантой. Если k_{i+1} не является импликантой ни для какой конъюнкции из L_j, то рассмотрим соседние конъюнкции $k_i' = x\,k_{i+1}$ и $k_i'' = \bar{x}\,k_{i+1}$. По условию k_i' и k_i'' таковы, что для них есть конъюнкции в L_j, для которых k_i' и k_i'' являются импликантами, причем это разные конъюнкции, иначе имел бы место случай $k_{i+1} \Rightarrow l_j$. Покажем, что они находятся в отношении смежности.

Пусть $k_i' \Rightarrow l_j'$ и $k_i'' \Rightarrow l_j''$. По предположению k_i' не является импликантой для l_j'', а k_i'' – импликантой для l_j'. Следовательно, l_j' состоит из каких-то литералов, принадлежащих k_{i+1}, и литерала x, а l_j'' – из каких-то литералов, также принадлежащих k_{i+1} (возможно, других), и литерала $\bar{x}$. Следовательно, l_j' и l_j'' смежны, а их продукт обобщенного склеивания имплицируется конъюнкцией k_{i+1}.

Таким образом, для любой конъюнкции из любого множества C_i, среди которых находятся все простые импликанты, имеется имплицируемая ею конъюнкция в множестве L_m. Следовательно все простые импликанты

находятся в L_m. Непростых импликант там нет, так как они удалены в результате выполнения простого поглощения.

Если исходная ДНФ задана в виде троичной матрицы, то рекомендуется организовать процесс получения сокращенной ДНФ (троичной матрицы, где каждая строка представляет простую импликанту) следующим образом. Строки исходной матрицы просматриваются сверху вниз, и для очередной строки отыскиваются смежные с ней строки, расположенные сверху от нее. Получаемые продукты обобщенного склеивания добавляются в матрицу снизу. Поглощаемые строки удаляются. Процесс заканчивается на самой нижней строке, когда результаты обобщенного склеивания со смежными с ней строками не дают новых строк для матрицы.

Пусть булева функция задана следующей троичной матрицей:

$$\begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 & \\ \left[\begin{array}{ccccc} 1 & 1 & - & 0 & 0 \\ 1 & 0 & - & 1 & 1 \\ 0 & 0 & 0 & - & 1 \\ 0 & - & 1 & 0 & - \\ - & 1 & 1 & 1 & 1 \\ 1 & - & 1 & 0 & 0 \\ 1 & 1 & - & 0 & 1 \end{array}\right. & & & & & \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \end{array}$$

В этой матрице находим следующие пары смежных строк: (2, 3), (1, 4), (3, 4), (2, 5), (4, 5). Выполняя над ними операцию обобщенного склеивания, получим строки

$$\begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 & \\ \left[\begin{array}{ccccc} - & 0 & 0 & 1 & 1 \\ - & 1 & 1 & 0 & 0 \\ 0 & 0 & - & 0 & 1 \\ 1 & - & 1 & 1 & 1 \\ 0 & 1 & 1 & - & 1 \end{array}\right. & & & & & \begin{array}{l} 8(2,3) \\ 9(1,4) \\ 10(3,4)' \\ 11(2,5) \\ 12(4,5) \end{array} \end{array}$$

которые не поглощаются заданными строками и сами не поглощают другие строки. Справа приведена нумерация строк, которая продолжает исходную нумерацию, а в скобках приведены номера склеиваемых строк.

Дальнейший поиск дает пары смежных строк (4, 6), (1, 7), (4, 7), (5, 7), (6, 7), для которых получим строки

$$\begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \begin{bmatrix} - & - & 1 & 0 & 0 \\ 1 & 1 & - & 0 & - \\ - & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & - & 1 \\ 1 & 1 & - & 0 & - \end{bmatrix} \end{array} \begin{matrix} 13(4,6) \\ 14(1,7) \\ 15(4,7) \\ 16(5,7) \\ 17(6,7) \end{matrix}.$$

Строка 13 поглощает строки 6 и 9, а строка 17 – строки 1 и 7. Строка 17 совпадает со строкой 14. Поэтому строки 1, 6, 7, 9 и 17 удаляются. Далее получаются следующие непоглощаемые строки:

$$\begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \begin{bmatrix} - & 1 & 1 & 0 & - \\ - & 1 & 1 & - & 1 \end{bmatrix} \end{array} \begin{matrix} 18(4,14) \\ 19(5,15) \end{matrix}.$$

Последняя строка поглощает строки 5, 12, 15 и 16. Окончательно получаем следующую матрицу (с естественной нумерацией строк), представляющую сокращенную ДНФ:

$$\begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & - & 1 & 1 \\ 0 & 0 & 0 & - & 1 \\ 0 & - & 1 & 0 & - \\ - & 0 & 0 & 1 & 1 \\ 0 & 0 & - & 0 & 1 \\ 1 & - & 1 & 1 & 1 \\ - & - & 1 & 0 & 0 \\ 1 & 1 & - & 0 & - \\ - & 1 & 1 & 0 & - \\ - & 1 & 1 & - & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{matrix}.$$

Второй этап процесса минимизации можно выполнить так же, как это делается в методе Квайна-МакКласки, т. е. свести его к решению задачи покрытия, разбив полученные интервалы на булевы векторы – элементы множества M^1. Комбинаторный поиск при решении задачи покрытия будет значительно сокращен, если удастся выделить *ядро* – множество обязательных интервалов. При существовании ядра иногда удается выделить *антиядро* – множество интервалов, покрываемое ядром. Тогда элементы ядра включаются в

решение, элементы антиядра исключаются из рассмотрения и оставшимися интервалами покрываются элементы множества M^1, не покрытые ядром.

Если максимальный интервал не принадлежит ядру, то после его удаления из всей совокупности максимальных интервалов оставшиеся интервалы будут покрывать множество M^1. Таким образом, задача поиска ядра сводится к нахождению избыточных конъюнкций в ДНФ.

Можно предложить другой способ выполнения второго этапа, который не требует разбиения интервалов на отдельные элементы множества M^1.

Пусть $\boldsymbol{U}$ – троичная матрица, представляющая сокращенную ДНФ некоторой булевой функции. Рассмотрим совокупность $P(\boldsymbol{U})$ непустых подмножеств множества номеров строк матрицы $\boldsymbol{U}$ такую, что для каждого элемента множества M^1 имеется подмножество в совокупности $P(\boldsymbol{U})$, состоящее из номеров всех строк матрицы $\boldsymbol{U}$, представляющих интервалы, содержащие данный элемент, и других подмножеств в $P(\boldsymbol{U})$ нет. Например, если матрицей $\boldsymbol{U}$ является приведенная выше матрица, то для вектора (1 0 0 1 1) таким подмножеством является {1, 4}.

Введем обозначение: $t(\boldsymbol{u}, \boldsymbol{U})$ – множество номеров тех строк матрицы $\boldsymbol{U}$, которые представляют интервалы, содержащие булев вектор $\boldsymbol{u}$. Например, для приведенной выше матрицы и $\boldsymbol{u}$ = (1 0 0 1 1) имеем $t(\boldsymbol{u}, \boldsymbol{U}) = \{1, 4\}$. Ясно, что $t(\boldsymbol{u}, \boldsymbol{U})$ является элементом множества $P(\boldsymbol{U})$.

Теперь задачу кратчайшего покрытия множества M^1 максимальными интервалами можно сформулировать следующим образом: для заданной матрицы $\boldsymbol{U}$ построить минимальную совокупность ее строк такую, что любой элемент множества $P(\boldsymbol{U})$ содержит номер хотя бы одной строки из данной совокупности.

Остановимся на способе получения множества $P(\boldsymbol{U})$. Пусть для двух троичных матриц $\boldsymbol{U}_1$ и $\boldsymbol{U}_2$ с одинаковым числом строк получены множества $P(\boldsymbol{U}_1)$ и $P(\boldsymbol{U}_2)$.

У т в е р ж д е н и е 18.2. Если из матриц $\boldsymbol{U}_1$ и $\boldsymbol{U}_2$ построить матрицу $\boldsymbol{U}$, приписав столбцы матрицы $\boldsymbol{U}_2$ к столбцам $\boldsymbol{U}_1$, то множество $P(\boldsymbol{U})$ можно получить, взяв за его элементы всевозможные непустые пересечения элементов $P(\boldsymbol{U}_1)$ с элементами $P(\boldsymbol{U}_2)$.

Для того, чтобы это показать, преобразуем матрицы $\boldsymbol{U}_1$ и $\boldsymbol{U}_2$ следующим образом. К матрице $\boldsymbol{U}_1$ припишем справа столько столбцов, сколько их имеет матрица $\boldsymbol{U}_2$. Всем элементам этих столбцов придадим значение «–». Такие же столбцы, число которых равно числу столбцов исходной матрицы $\boldsymbol{U}_1$, припишем слева к матрице $\boldsymbol{U}_2$. Очевидно, $P(\boldsymbol{U}_1)$ и $P(\boldsymbol{U}_2)$ при этом не изменятся. Произвольно выберем вектор $\boldsymbol{u}$, принадлежащий какому-нибудь интервалу, представляемому строкой матрицы $\boldsymbol{U}$. По построению матрицы $\boldsymbol{U}$ имеем $t(\boldsymbol{u}, \boldsymbol{U}) = t(\boldsymbol{u}, \boldsymbol{U}_1) \cap t(\boldsymbol{u}, \boldsymbol{U}_2)$, т. е. каждый элемент множества $P(\boldsymbol{U})$ является пересечением двух множеств, одно из которых является элементом множества $P(\boldsymbol{U}_1)$, а другое – элементом множества $P(\boldsymbol{U}_2)$. С другой стороны, для каждой пары пересекающихся элементов, один из которых взят из $P(\boldsymbol{U}_1)$, а другой – из

$P(\boldsymbol{U}_2)$, найдется вектор $\boldsymbol{u}$ такой, что $t(\boldsymbol{u}, \boldsymbol{U}_1)$ и $t(\boldsymbol{u}, \boldsymbol{U}_2)$ являются теми самыми элементами. Множество $t(\boldsymbol{u}, \boldsymbol{U}_1) \cap t(\boldsymbol{u}, \boldsymbol{U}_2)$ состоит из номеров всех строк матриц $\boldsymbol{U}_1$ и $\boldsymbol{U}_2$, каждая из которых представляет интервал, содержащий $\boldsymbol{u}$. Пересечение интервалов, представляемых строками матриц $\boldsymbol{U}_1$ и $\boldsymbol{U}_2$ с общим номером i, равно интервалу, представляемому i-й строкой матрицы $\boldsymbol{U}$. Следовательно, для любой пары $t(\boldsymbol{u}, \boldsymbol{U}_1)$ и $t(\boldsymbol{u}, \boldsymbol{U}_2)$ найдется множество $t(\boldsymbol{u}, \boldsymbol{U})$ такое, что $t(\boldsymbol{u}, \boldsymbol{U}) = t(\boldsymbol{u}, \boldsymbol{U}_1) \cap t(\boldsymbol{u}, \boldsymbol{U}_2)$.

Если матрица $\boldsymbol{U}_1$ представляет собой один столбец и произвольное число одноэлементных строк, то $P(\boldsymbol{U}_1)$ состоит не более, чем из двух элементов. Один из них представляет множество, состоящее из номеров всех строк, имеющих нули и «–», второй – множество, состоящее из номеров всех строк, имеющих единицы и «–». Если все элементы одностолбцовой матрицы $\boldsymbol{U}_1$ имеют одно и то же значение, то $P(\boldsymbol{U}_1)$ состоит из одного элемента.

Пусть матрица $\boldsymbol{U}$ имеет m столбцов. Для матрицы $\boldsymbol{U}_1$, состоящей из единственного первого столбцы получим $P(\boldsymbol{U}_1)$, как указано выше. Множество $P(\boldsymbol{U}_2)$ для матрицы, состоящей из двух первых столбцов матрицы $\boldsymbol{U}$, получим путем всевозможных пересечений элементов множества $P(\boldsymbol{U}_1)$ с элементами подобного множества для второго столбца. Для матрицы $\boldsymbol{U}_3$, состоящей из трех первых столбцов матрицы $\boldsymbol{U}$, получим $P(\boldsymbol{U}_3)$ аналогично, путем всевозможных пересечений элементов множества $P(\boldsymbol{U}_2)$ с элементами подобного множества для третьего столбца. Продолжая таким образом выполнять операцию пересечения множеств, получим, наконец, $P(\boldsymbol{U}_m) = P(\boldsymbol{U})$. Если в результате вычисления некоторого $P(\boldsymbol{U}_i)$ $(1 \le i \le m)$ появилось одноэлементное множество, то строка матрицы $\boldsymbol{U}$ с соответствующим номером включается в решение как принадлежащая ядру.

Продемонстрируем процесс получения $P(\boldsymbol{U})$ на примере матрицы

$$\boldsymbol{U} = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & - & 1 & 1 \\ 0 & 0 & 0 & - & 1 \\ 0 & - & 1 & 0 & - \\ - & 0 & 0 & 1 & 1 \\ 0 & 0 & - & 0 & 1 \\ 1 & - & 1 & 1 & 1 \\ - & - & 1 & 0 & 0 \\ 1 & 1 & - & 0 & - \\ - & 1 & 1 & 0 & - \\ - & 1 & 1 & - & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{matrix},$$

для которой получаем следующее:

$P(\boldsymbol{U}_1)$ = {(1,4,6,7,8,9,10), (2,3,4,5,7,9,10)};
$P(\boldsymbol{U}_2)$ = {(1,4,6,7), (6,7,8,9,10), (2,3,4,5,7), (3,7,9,10)};
$P(\boldsymbol{U}_3)$ = {(1,4), (1,6,7), (8), (6,7,8,9,10), (2,4,5), (3,5,7), (3,7,9,10)};
$P(\boldsymbol{U}_4)$ = {(1,4), (1,6), (7), (8), (6,10), (7,8,9,10), (2,4), (2,5), (3,5,7), (10), (3,7,9,10)};
$P(\boldsymbol{U}_5)$ = {(1,4), (1,6), (7), (8), (6,10), (8,9,10), (7,8,9), (2,4), (2,5), (3,5), (3,7), (10), (3,9,10), (3,7,9)}.

Сформулируем правила редукции, которые можно применять при решении задачи покрытия и которые согласуются с правилами, сформулированными в гл. 7.

1. Если $A \in P(\boldsymbol{U})$, $B \in P(\boldsymbol{U})$ и $A \subseteq B$, то B удаляется из $P(\boldsymbol{U})$.

2. Если номер i присутствует только в тех элементах множества $P(\boldsymbol{U})$, где присутствует номер k, то i удаляется отовсюду.

Применив правило 1, получим $P(\boldsymbol{U})$ = {(1,4), (1,6), (7), (8), (2,4), (2,5), (3,5), (10)}. Строки 7, 8 и 10 составляют ядро. Антиядро состоит из одной строки 9.

По правилу 2 удаляются номера 3 и 6: $P(\boldsymbol{U})$ = {(1,4), (1), (7), (8), (2,4), (2,5), (5), (10)}.

Применив снова правило 1, получим $P(\boldsymbol{U})$ = {(1), (7), (8), (2,4), (5), (10)}.

Окончательно имеем два решения рассматриваемого примера:

$$\begin{array}{c} x_1\ x_2\ x_3\ x_4\ x_5 \\ \left[\begin{array}{ccccc} 1 & 0 & - & 1 & 1 \\ 0 & 0 & 0 & - & 1 \\ 0 & 0 & - & 0 & 1 \\ - & - & 1 & 0 & 0 \\ 1 & 1 & - & 0 & - \\ - & 1 & 1 & - & 1 \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 5 \\ 7 \\ 8 \\ 10 \end{array} \quad \text{и} \quad \begin{array}{c} x_1\ x_2\ x_3\ x_4\ x_5 \\ \left[\begin{array}{ccccc} 1 & 0 & - & 1 & 1 \\ - & 0 & 0 & 1 & 1 \\ 0 & 0 & - & 0 & 1 \\ - & - & 1 & 0 & 0 \\ 1 & 1 & - & 0 & - \\ - & 1 & 1 & - & 1 \end{array}\right] \end{array} \begin{array}{c} 1 \\ 4 \\ 5 \\ 7 \\ 8 \\ 10 \end{array}.$$

Г л а в а 19

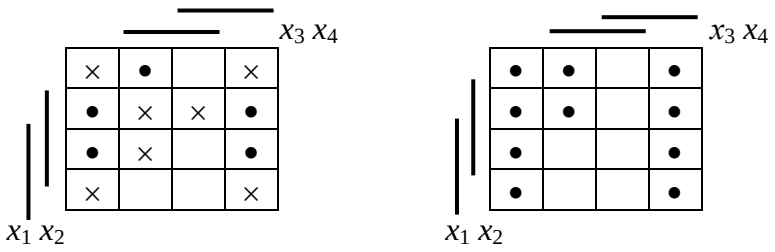
Минимизация не полностью определенных булевых функций

19.1. Постановка задачи

Описание поведения проектируемого устройства может допускать такие ситуации, когда реакция устройства на некоторые комбинации входных сигналов не определена. В частности, может оказаться, что некоторые входные сигналы не являются независимыми. Например, некоторые сигналы таковы, что соответствующие им переменные x_i и x_j не могут принимать одновременно значение 1. Значения функций, соответствующих выходным сигналам, на всех наборах, где $x_i = x_j = 1$ считаются безразличными. На таких наборах значений входных переменных функции можно доопределять как угодно, и желательно давать им при этом такие значения, которые приводят к упрощению реализующих схем.

Не полностью определенная функция разбивает булево пространство M на три подмножества: M^1, M^0 и M^- – области, где соответственно функция имеет значения 1, 0 и не определена. Для задания функции достаточно задать два подмножества, например, M^1 и M^0.

Не полностью определенную булеву функцию можно задавать картой Карно, помещая знак × в клетки, где функция не определена. Пусть, например, функция задана картой Карно на рис. 19.1*а*. Оптимальный вариант ее доопределения представлен на рис. 19.1*б*. В этом случае область M^1 покрывается двумя интервалами из объединения $M^0 \cup M^-$ и на этих интервалах функции придаем значение 1. Соответствующая ДНФ имеет вид $\bar{x}_1\,\bar{x}_4 \vee \bar{x}_3$.



а) *б)*

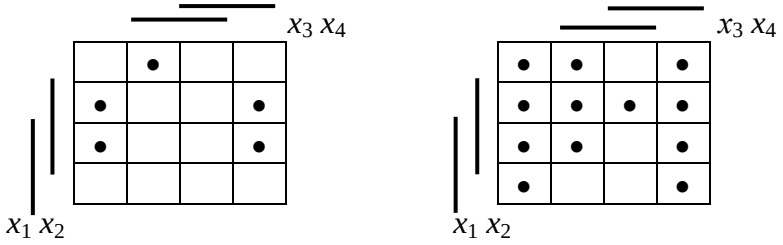
Рис. 19.1. Доопределение булевой функции: *а)* задание функции; *б)* ее доопределение

На множестве полностью и не полностью определенных функций введем отношение *реализации*, которое обозначим символом $\langle$ или $\rangle$ и положим, что $f \langle g$ (функция f реализуется функцией g) или $g \rangle f$ (функция g реализует функцию f), если $M^1{}_f \subseteq M^1{}_g$ и $M^0{}_f \subseteq M^0{}_g$.

Задача минимизации не полностью определенной функции f ставится следующим образом: для функции f найти минимальную (или кратчайшую) ДНФ среди всех ДНФ всех функций g, удовлетворяющих условию $f \langle g$.

19.2. Применение метода Квайна-МакКласки

Любую не полностью определенную булеву функцию f можно рассматривать как множество полностью определенных функций, получаемых различными вариантами доопределения. Число таких вариантов $2^{|M_f^-|}$. Однако нет необходимости перебирать их все. Среди вариантов доопределения функции f выделим функции $f_{\min}$ и $f_{\max}$, которые получаются соответственно приданием значения 0 и значения 1, где f не определена. То есть $M^1_{f_{\min}} = M^1_f$, $M^0_{f_{\min}} = M^0_f \cup M^-_f$ и $M^1_{f_{\max}} = M^1_f \cup M^-_f$, $M^0_{f_{\max}} = M^0_f$. Для функции f, заданной картой Карно на рис. 19.1*а*, варианты доопределения $f_{\min}$ и $f_{\max}$ показаны на рис. 19.2.



а) *б)*

Рис. 19.2. Варианты доопределения функции, заданной на рис. 19.1*а*: *а)* $f_{\min}$; *б)* $f_{\max}$

Легко показать, что искомая минимальная ДНФ состоит из простых импликант функции $f_{\max}$. Функция g, ДНФ которой является решением поставленной задачи, обладает свойством $f_{\min} \Rightarrow g \Rightarrow f_{\max}$. Следовательно, g имеет значение 1 везде, где это значение имеет $f_{\min}$, и имеет значение 0 везде, где это значение имеет $f_{\max}$.

Процесс минимизации не полностью определенной булевой функции f методом Квайна-МакКласки состоит из двух этапов: 1) нахождение множества всех простых импликант функции $f_{\max}$; 2) выделение из этого множества минимального подмножества, покрывающего совершенную ДНФ функции $f_{\min}$.

Рассмотрим матрицы для функции f, заданной картой Карно на рис. 19.1*а*:

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix},$$

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix}$$
$$\begin{bmatrix} 0 & - & - & 0 \\ - & 1 & - & 0 \\ - & - & 0 & - \\ 0 & 1 & - & - \end{bmatrix}.$$

Левая матрица представляет множество $M^1_{f_{\min}}$, правая – совокупность всех максимальных интервалов, на которых функция $f_{\max}$ имеет значение 1. Теперь надо этими интервалами покрыть множество $M^1_{f_{\min}}$. Строим матрицу покрытия, где $a_1, a_2, a_3\, a_4, a_5$ – элементы множества $M^1_{f_{\min}}$, а $B_1, B_2, B_3\, B_4$ – максимальные интервалы области $M^1_{f_{\max}}$:

$$\begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & \\ \left[\begin{matrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{matrix}\right. & & & & & \begin{matrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{matrix} \end{matrix}.$$

Кратчайшее покрытие этой матрицы составляют строки B_1 и B_3. Окончательное решение имеет вид $f = \bar{x}_1\, \bar{x}_4 \vee \bar{x}_3$, что совпадает с решением, полученным визуально по карте Карно.

19.3. Минимизация слабо определенной функции

Часто на практике встречаются булевы функции, область определения которых занимает небольшую часть булева пространства ее аргументов, т. е. $|M^1| + |M^0| << |M^-|$. Если применять описанный метод для минимизации такой функции f, то многие из простых импликант функции $f_{\max}$ будут соответствовать интервалам, которые целиком содержатся в множестве M^- и бесполезны для выполнения второго этапа минимизации. Поэтому сто́ит развивать методы, учитывающие данную специфику.

Для описания одного из таких методов, выполнение которого исключает рассмотрение элементов множества M^-, введем некоторые понятия.

Интервально поглощаемым множеством называется множество элементов из M^1, для которых существует интервал, содержащий все эти элементы и не пересекающийся с множеством M^0. Интервально поглощаемое множество является *максимальным*, если оно не содержится в качестве собственного подмножества в другом интервально поглощаемом множестве.

Рассматриваемый метод заключается в получении всех максимальных интервально поглощаемых множеств и в последующем получении кратчайшего покрытия ими элементов множества M^1. Интервалы, соответствующие элементам полученного покрытия определят кратчайшую ДНФ для заданной функции. Ранги конъюнкций, составляющих полученную ДНФ следует уменьшить, максимально расширив соответствующие интервалы так, чтобы они не пересекались с множеством M^0.

Для получения всех максимальных интервально поглощаемых множеств можно использовать лексико-графический перебор, подобный перебору, описанному в гл. 4 при изложении метода поиска максимальных независимых множеств в графе. При этом принимается во внимание тот факт, что если некоторое множество не является интервально поглощаемым, то не является таковым и любое множество, содержащее его в качестве подмножества.

Проверка, является ли некоторое подмножество M^1_i множества M^1 интервально поглощаемым множеством, довольно проста. Надо построить *минимальный покрывающий интервал* для M^1_i, т. е. наименьший по мощности интервал, содержащий все элементы множества M^1_i, и затем проверить, не пересекается ли он с множеством M^0.

Компоненты вектора, представляющего минимальный покрывающий интервал для M^1_i, находятся следующим образом: если значения одноименных компонент булевых векторов, принадлежащих M^1_i, совпадают, то это значение присваивается соответствующей компоненте получаемого троичного вектора, а если нет, то данная компонента принимает значение «–».

Например, для трех векторов (1 0 1 1 1 0 0), (0 0 1 0 1 1 0) и (1 0 1 1 0 1 0) получим троичный вектор (– 0 1 – – – 0), который представляет минимальный покрывающий интервал для данных булевых векторов.

Пусть булева функция задана следующими матрицами:

$$M^1 = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix}, \quad M^0 = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array}.$$

Данная функция является слабо определенной, так как из всех 64-х элементов булева пространства только 12 составляют область определения функции.

Множество $\{1, 2\}$ является интервально поглощаемым, так как минимальный покрывающий интервал (– 0 1 – – 0) не пересекается с множеством M^0. Добавление к нему элемента 3 превращает его в множество, которое не является интервально поглощаемым, так как минимальный покрывающий интервал для него (– – – – – 0) содержит векторы (0 0 0 1 0 0) и (0 1 0 0 1 0), принадлежащие множеству M^0. Нельзя также добавлять к множеству $\{1, 2\}$ элементов 4 и 5. Максимальным интервально поглощаемым множеством является множество $\{1, 2, 6, 7\}$, так как минимальный покрывающий интервал для него (– – 1 – – 0) не пересекается с множеством M^0

и дальнейшее расширение этого множества без потери свойства интервальной поглощаемости невозможно.

Продолжая данный перебор, получаем следующие максимальные интервально поглощаемые множества и соответствующие им минимальные покрывающие интервалы:

$$\begin{array}{ll} \{1, 2, 6, 7\}, & (-\,-\,1\,-\,-\,0); \\ \{1, 3\}, & (-\,-\,-\,1\;1\;0); \\ \{1, 5, 7\}, & (-\,0\,-\,-\,1\,-); \\ \{2, 4, 5, 7\}, & (-\,0\,-\,0\,-\,-); \\ \{2, 4, 6\}, & (-\,-\,1\;0\;0\,-). \end{array}$$

Множества $\{1, 2, 6, 7\}$, $\{1, 3\}$ и $\{2, 4, 5, 7\}$ составляют кратчайшее покрытие множества M^1. После расширения второго интервала получаем троичную матрицу

$$\begin{array}{c} x_1\;\;x_2\;\;x_3\;\;x_4\;\;x_5\;\;x_6 \\ \begin{bmatrix} - & - & 1 & - & - & 0 \\ - & - & - & 1 & 1 & - \\ - & 0 & - & 0 & - & - \end{bmatrix} \end{array}$$

и соответствующую ей ДНФ $x_3\,\bar{x}_6 \vee x_4\,x_5 \vee \bar{x}_2\,\bar{x}_4$.

19.4. Расширение интервалов

В данном примере минимальные покрывающие интервалы оказались довольно широкими, и расширить удалось только один из них и только в одном направлении (заменив 0 на «–» во втором интервале). Расширять интервал можно, последовательно заменяя единицы и нули на символ «–» в соответствующем троичном векторе (двигаясь слева направо или справа налево), пока очередное расширение не захватит элемент из множества M^0. Но в этом случае нет гарантии получения минимума ранга интервала. Например, надо расширить интервал, представляемый вектором (0 1 – 1 0), но так, чтобы он не пересекался с множеством булевых векторов, заданном матрицей

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

Если по порядку расположения компонент пытаться заменять единицы и нули на «–», то получим не самый широкий интервал (– 1 – 1 0), в то время как можно получить лучше: (0 1 – – –).

Поставим следующую задачу. В троичном векторе $\boldsymbol{u}$, ортогональном всем строкам троичной матрицы $\boldsymbol{U}$, заменить максимальное число значений 0 и 1 на «–» так, чтобы он оставался ортогональным всем строкам матрицы $\boldsymbol{U}$.

Можно решать эту задачу следующим образом. Строим *матрицу различий* вектора $\boldsymbol{u}$ по отношению к строкам матрицы $\boldsymbol{U}$. Строки матрицы различий соответствуют строкам матрицы $\boldsymbol{U}$, и каждая из них показывает, по каким компонентам вектор $\boldsymbol{u}$ отличается от соответствующей строки матрицы $\boldsymbol{U}$. Для этого строка матрицы различий получается путем покомпонентного сложения по модулю два соответствующей строки матрицы $\boldsymbol{U}$ с сектором $\boldsymbol{u}$. При этом считается, что $0 \oplus - = 1 \oplus - = 0$. Для матрицы различий надо найти минимальную совокупность столбцов такую, чтобы каждая строка матрицы различий имела в данной совокупности хотя бы одну единицу. То есть надо покрыть строки минимальным числом столбцов. В векторе $\boldsymbol{u}$ оставляются прежние значения только для тех компонент, которые соответствуют столбцам из полученной совокупности. Остальные компоненты принимают значение «–».

Для приведенной выше матрицы и вектора $\boldsymbol{u}$ = (0 1 – 1 0) матрица различий имеет вид

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

Кратчайшее *столбцовое* покрытие этой матрицы составляют ее первые два столбца. Следовательно, решением является вектор (0 1 – – –).

Г л а в а 20

Минимизация системы булевых функций

20.1. Минимизация системы ДНФ

Задача минимизации системы ДНФ формулируется следующим образом: для заданной системы булевых функций получить такую систему ДНФ, в которой общее число различных элементарных конъюнкций было бы минимальным.

Если минимизировать каждую функцию по отдельности, то число различных элементарных конъюнкций во всех ДНФ может оказаться больше, чем при совместной минимизации функций. Рассмотрим систему из двух функций, представленную картами Карно на рис. 20.1.

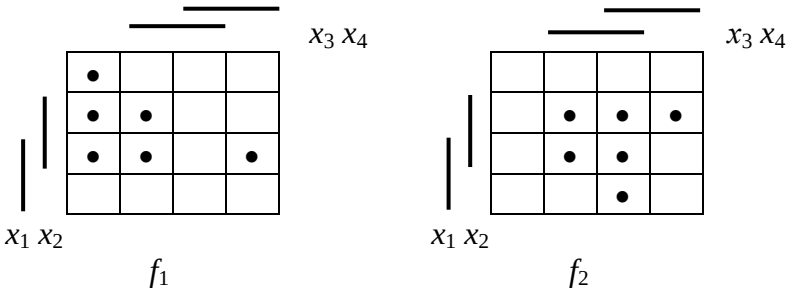


Рис. 20.1. Система булевых функций

Если минимизировать каждую функцию по отдельности, то получим следующие ДНФ:

$$f_1(x_1, x_2, x_3, x_4) = \bar{x}_1\ \bar{x}_3\ \bar{x}_4 \vee x_1\ x_2\ \bar{x}_3 \vee x_2\ \bar{x}_4;$$
$$f_2(x_1, x_2, x_3, x_4) = \bar{x}_1\ x_2\ x_4 \vee x_1\ x_3\ x_4 \vee x_2\ x_3.$$

Общее число различных элементарных конъюнкций в этой системе равно шести. Совместная минимизация данных функций дает следующий результат:

$$f_1(x_1, x_2, x_3, x_4) = \bar{x}_1\ \bar{x}_3\ \bar{x}_4 \vee x_1\ x_2\ \bar{x}_3 \vee x_2\ x_3\ \bar{x}_4;$$
$$f_2(x_1, x_2, x_3, x_4) = \bar{x}_1\ x_2\ x_4 \vee x_1\ x_3\ x_4 \vee x_2\ x_3\ \bar{x}_4.$$

В полученных ДНФ присутствует одна общая элементарная конъюнкция $x_2\ x_3\ \bar{x}_4$, и за счет этого их число снизилось на единицу.

Посмотрим, как метод Квайна-МакКласки можно распространить на случай системы булевых функций.

Пусть F – некоторая система булевых функций. Элементарная конъюнкция является *обобщенной простой импликантой* для множества булевых функций $F' \subseteq F$, если она является импликантой для любой функции из F' и перестает быть импликантой при удалении из нее любого литерала и не является импликантой ни для какой функции, не принадлежащей F'.

Процесс минимизации системы ДНФ по данному методу состоит из двух этапов: 1) нахождение множества всех обобщенных простых импликант для заданной системы; 2) выделение из этого множества минимального подмножества, удовлетворяющего условию, что для всякой функции, принадлежащей заданной системе, можно из этого подмножества получить задающую ее ДНФ.

Исходная система булевых функций должна быть представлена в виде системы совершенных ДНФ, которую будем задавать парой булевых матриц – матрицей аргументов и матрицей функций.

Первый этап выполняется с помощью простого склеивания. При этом учитываются *характеристики* импликант. Под характеристикой понимается множество функций, для которого данная конъюнкция является импликантой. Начальные значения характеристик задаются строками матрицы функций. Результату склеивания приписывается в качестве характеристики пересечение характеристик склеиваемых конъюнкций, которое получается поразрядной конъюнкцией соответствующих векторов. Естественно, что если результатом поразрядной конъюнкции является вектор, все компоненты которого нули, то результат склеивания не рассматривается.

Так же, как при минимизации одной функции, склеиваемым конъюнкциям приписывается знак «*», но приписывается он только тем конъюнкциям, характеристика которых совпадает с характеристикой результата склеивания.

Рассмотрим процесс получения обобщенных простых импликант на примере системы функций, заданных картами Карно на рис. 20.1. Получаем следующую последовательность пар матриц:

$$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \quad \begin{array}{cc} f_1 & f_2 \end{array}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{array}{c} 1* \\ 2* \\ 3* \\ 4* \\ 5* \\ 6* \\ 7* \\ 8* \\ 9* \\ 10* \end{array} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix};$$

$$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \quad \begin{array}{cc} f_1 & f_2 \end{array}$$

$$\begin{bmatrix} 0 & - & 0 & 0 \\ - & 1 & 0 & 0 \\ 0 & 1 & - & 0 \\ 1 & 1 & - & 0 \\ 1 & 1 & 0 & - \\ - & 1 & 1 & 0 \\ 0 & 1 & 1 & - \\ 0 & 1 & - & 1 \\ 1 & 1 & 1 & - \\ - & 1 & 1 & 1 \\ 1 & - & 1 & 1 \end{bmatrix} \begin{array}{c} \\ * \\ * \\ * \\ \\ \\ * \\ \\ * \\ * \\ \end{array} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix};$$

$$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \quad \begin{array}{cc} f_1 & f_2 \end{array}$$

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

Полученную *сокращенную систему ДНФ*, которая содержит все обобщенные простые импликанты представим следующей парой матриц, где собраны строки, не отмеченные знаком «*»:

$$\boldsymbol{X} = \begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix} \begin{bmatrix} 0 & - & 0 & 0 \\ 1 & 1 & 0 & - \\ - & 1 & 1 & 0 \\ 0 & 1 & - & 1 \\ 1 & - & 1 & 1 \\ - & 1 & - & 0 \\ - & 1 & 1 & - \end{bmatrix}, \qquad \boldsymbol{Y} = \begin{matrix} f_1 & f_2 \end{matrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Строки матрицы $\boldsymbol{X}$ представляют интервалы булева пространства аргументов. В матрице $\boldsymbol{Y}$ единица в столбце f_i и j-й строке показывает, что на всем j-м интервале функция f_i имеет значение 1 (j-я конъюнкция входит в ДНФ функции f_i). Такая форма названа ранее матричным представлением системы ДНФ.

Рассмотрим теперь выполнение второго этапа минимизации, который сводится к задаче покрытия. Пара одноименных строк матриц $\boldsymbol{X}$ и $\boldsymbol{Y}$ представляет множество пар вида (m_i, f_j), где m_i – элемент булева пространства аргументов, а f_j – функция, принимающая значение 1 на этом элементе. Интервал и его характеристика являются сомножителями декартова произведения, результат которого представляет собой множество с элементами вида (m_i, f_j). Очевидно, всего пар вида (m_i, f_j) в задании системы булевых функций столько, сколько единиц в исходной матрице функций. Необходимо из полученной совокупности интервалов выбрать минимум так, чтобы выбранные интервалы со своими характеристиками покрыли все множество пар (m_i, f_j) в задании исходной системы.

Для решения рассматриваемого примера построим следующую матрицу, строкам которой соответствуют пары строк матриц $\boldsymbol{X}$ и $\boldsymbol{Y}$, а столбцам – пары вида (m_i, f_j):

$(1, f_1)$	$(2, f_1)$	$(3, f_1)$	$(4, f_1)$	$(6, f_1)$	$(7, f_1)$	$(4, f_2)$	$(5, f_2)$	$(6, f_2)$	$(8, f_2)$	$(9, f_2)$	$(10, f_2)$
1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	1	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	1	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	1
0	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	1	1	0	1

Кратчайшее строчное покрытие приведенной матрицы соответствует кратчайшей системе ДНФ, представляемой следующими матрицами:

$$\boldsymbol{X} = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix} \\ \begin{bmatrix} 0 & - & 0 & 0 \\ 1 & 1 & 0 & - \\ - & 1 & 1 & 0 \\ 0 & 1 & - & 1 \\ 1 & - & 1 & 1 \end{bmatrix} \end{array}, \qquad \boldsymbol{Y} = \begin{array}{c} \begin{matrix} f_1 & f_2 \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array}.$$

Этот результат совпадает с тем, что был получен ранее для данной системы булевых функций.

Решение задачи минимизации системы ДНФ ведет к получению оптимальной структуры программируемой логической матрицы.

20.2. Минимизация системы слабо определенных булевых функций

Булева функция является слабо определенной, если $|M^1| + |M^0| << |M^-|$. Пусть $F = \{f_1, f_2, \ldots, f_m\}$ – система слабо определенных булевых функций, где любая f_i задана с помощью множеств $M^1{}_i$ – область единичных значений и $M^0{}_i$ – область нулевых значений функции f_i. Очевидно, кратчайшая система ДНФ для системы F сводится к нахождению такого минимального множества интервалов булева пространства M, чтобы каждое из множеств $M^1{}_i$ покрывалось теми из них, которые не пересекаются с множеством $M^0{}_i$.

Рассматриваемый метод минимизации привлекает понятие *интервально поглощаемого множества*. В данном случае это множество элементов вида (m_j, f_k), где m_j – элемент булева пространства M, а f_k – функция, которая имеет значение 1 на этом интервале, т. е. $m_j \in M^1{}_k$. Множество элементов такого вида является интервально поглощаемым, если существует такой интервал пространства M, который для каждой пары (m_j, f_k) из этого множества содержит m_j и не пересекается с множеством $M^0{}_k$. Интервально поглощаемое множество

называется максимальным, если оно не является подмножеством никакого другого интервально поглощаемого множества.

Задача состоит в том, чтобы для заданной системы булевых функций F найти минимальную совокупность интервально поглощаемых множеств, покрывающую все пары вида (m_j, f_k). При этом достаточно рассматривать только *максимальные* интервально поглощаемые множества. Для каждого множества из полученного покрытия надо взять соответствующий ему интервал и максимально расширить его так, чтобы он не пересекался ни с одним из множеств $M^0{}_k$ тех функций f_k, которые принадлежат парам (m_j, f_k), входящим в данное множество.

В качестве примера рассмотрим систему слабо определенных булевых функций, заданную следующей парой матриц:

$$\boldsymbol{X} = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3, \\ 4 \\ 5 \\ 6 \end{matrix} \quad \boldsymbol{Y} = \begin{array}{c} \begin{matrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array}.$$

Эти матрицы можно интерпретировать как задание поведения некоторого устройства. На каждый набор двоичных входных сигналов, представленный матрицей $\boldsymbol{X}$, устройство реагирует набором выходных сигналов, представленный соответствующей строкой матрицы $\boldsymbol{Y}$. Поведение устройства не определено на всех наборах входных сигналов, которые не представлены матрицей $\boldsymbol{X}$. Очевидно, столбец f_i матрицы $\boldsymbol{Y}$ представляет множества $M^1{}_i$ и $M^0{}_i$ функции f_i.

Всякое интервально поглощаемое множество является декартовым произведением $M_p \times F_p$, где M_p – множество некоторых элементов булева пространства, а F_p – множество функций, принимающих значение 1 на этих элементах. Поэтому для компактности интервально поглощаемые множества будем представлять парами сомножителей. Используя лексико-графический перебор, получим следующие максимальные интервально поглощаемые множества и соответствующие им интервалы:

$\{1\}$,	$\{f_1, f_2, f_4, f_5\}$;	(0 1 0 1 0);
$\{1, 2, 4\}$,	$\{f_4\}$;	(– – – 1 –);
$\{1, 3, 5\}$,	$\{f_2\}$;	(0 1 – – –);
$\{1, 4\}$,	$\{f_1, f_4, f_5\}$;	(0 – – 1 0);
$\{1, 4, 6\}$,	$\{f_4\}$;	(– – – – 0);
$\{1, 5\}$,	$\{f_2, f_5\}$;	(0 1 0 – –);

{2},	{f_2, f_3, f_4, f_6};	(1 0 1 1 1);
{2, 3},	{f_2, f_3};	(– – 1 – 1);
{2, 3, 5},	{f_2};	(– – – – 1);
{2, 6},	{f_3, f_4, f_6};	(1 – 1 – –);
{3, 6},	{f_3};	(– 1 1 0 –).

Построим матрицу покрытия, строкам которой соответствуют полученные максимальные интервально поглощаемые множества, а столбцам пары (1, f_1), (4, f_1), (1, f_2), (2, f_2), (3, f_2), (5, f_2), (2, f_3), (3, f_3), (6, f_3), (1, f_4), (2, f_4), (4, f_4), (6, f_4), (1, f_5), (4, f_5), (5, f_5), (2, f_6), (6, f_6), определяемые по единицам матрицы $\boldsymbol{Y}$:

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

Строчное покрытие этой матрицы составляют четвертая, шестая, восьмая и десятая строки. После расширения соответствующих интервалов получим следующую систему ДНФ:

$$\boldsymbol{X} = \begin{array}{c}
\begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\
\begin{bmatrix}
0 & – & – & 1 & – \\
0 & 1 & – & – & – \\
– & – & 1 & – & 1 \\
1 & – & 1 & – & –
\end{bmatrix}
\end{array}, \quad
\boldsymbol{Y} = \begin{array}{c}
\begin{matrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \end{matrix} \\
\begin{bmatrix}
1 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1
\end{bmatrix}
\end{array}.$$

Г л а в а 21

Декомпозиция булевых функций

Задача декомпозиции булевой функции состоит в том, чтобы представить заданную функцию в виде суперпозиции более простых функций. Фактически реализация булевой функции схемой из логических элементов, или синтез комбинационной схемы, сводится к задаче декомпозиции, когда получаемая суперпозиция должна содержать функции, реализуемые отдельными логическими элементами.

Примером суперпозиции вида $f(\boldsymbol{x}) = \varphi(g_1, g_2, \ldots, g_m)$, где $g_i = g_i(\boldsymbol{z}_i)$, а $\boldsymbol{z}_i$ – булев вектор, составленный из компонент вектора $\boldsymbol{x}$ ($i = 1, 2, \ldots, m$), является дизъюнктивная нормальная форма, где в качестве функций g_i выступают элементарные конъюнкции, а в качестве φ – дизъюнкция.

Задачи синтеза комбинационных схем разнообразны, что обусловлено применением различных критериев оптимизации и учетом различных технологических ограничений. Поэтому задача декомпозиции булевых функций может иметь много разных постановок. Рассмотрим наиболее простые задачи декомпозиции, связанные с минимизацией числа аргументов получаемых функций. Технически это выражается в минимизации числа внешних полюсов блоков схемы, реализующих данные функции.

21.1. Двухблочная разделительная декомпозиция

Наиболее простая декомпозиция получает представление некоторой функции $f(\boldsymbol{x})$, где $\boldsymbol{x}$ – вектор, компонентами которого являются переменные из множества X, в виде

$$f(\boldsymbol{x}) = \varphi(g(\boldsymbol{z}_1), \boldsymbol{z}_2).$$

Здесь $\boldsymbol{z}_1$ и $\boldsymbol{z}_2$ – векторы, компоненты которых составляют непустые множества Z_1 и Z_2 соответственно, причем

$$Z_1 \cup Z_2 = X, \quad Z_1 \cap Z_2 = \varnothing.$$

Такой тип декомпозиции порождает схему соединения двух блоков, изображенную на рис. 21.1. Он называется *двухблочной разделительной декомпозицией* ввиду того, что исходная функция $f(\boldsymbol{x})$ разлагается на функции φ и g, реализуемые двумя блоками, а множество аргументов X разделено на непересекающиеся подмножества.

Естественно рассматривать только нетривиальные случаи, когда

$$1 < |Z_1| < n = |X|,$$

поскольку при $|Z_1| = n$ или $|Z_1| = 1$ одна из функций φ или g имеет аргументов не меньше, чем исходная функция f.

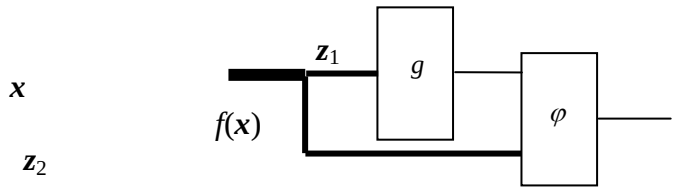


Рис. 21.1. Схема, реализующая суперпозицию функций $f(\boldsymbol{x}) = \varphi(g(\boldsymbol{z}_1), \boldsymbol{z}_2)$

Чтобы определить свойство булевых функций, допускающих нетривиальную двухблочную разделительную декомпозицию, рассмотрим задание функции двумерной таблицей, строки которой кодируются значениями переменных из множества Z_1 (значениями векторной переменной $\boldsymbol{z}_1$), а столбцы – значениями переменных из множества Z_2 (значениями векторной переменной $\boldsymbol{z}_2$). На пересечении строки и столбца помещается значение функции, которое она принимает на соответствующих наборах значений переменных. Такая таблица называется *картой декомпозиции* для множеств переменных (Z_1, Z_2). В данном случае карта декомпозиции имеет $2^{|Z_1|}$ строк и $2^{|Z_2|}$ столбцов. Порядок расположения строк и столбцов здесь не имеет особого значения, но если для их кодирования использовать код Грея, то карта декомпозиции принимает вид карты Карно.

У т в е р ж д е н и е 21.1. Полностью определенная булева функция $f(\boldsymbol{x})$ допускает двухблочную разделительную декомпозицию в форме $f(\boldsymbol{x}) = \varphi(g(\boldsymbol{z}_1), \boldsymbol{z}_2)$ тогда и только тогда, когда в карте декомпозиции для (Z_1, Z_2) имеется не более двух различных значений строк.

Действительно, допустим, что существует указанное представление некоторой заданной функции $f(\boldsymbol{x})$, а в ее карте декомпозиции для (Z_1, Z_2) имеется три различных значения строк. Пусть эти строки соответствуют наборам значений $\boldsymbol{\sigma}_u$, $\boldsymbol{\sigma}_v$ и $\boldsymbol{\sigma}_w$ переменных из множества Z_1. Эти строки представляют соответственно булевы функции

$$\varphi(g(\boldsymbol{\sigma}_u), \boldsymbol{z}_2),\ \varphi(g(\boldsymbol{\sigma}_v), \boldsymbol{z}_2),\ \varphi(g(\boldsymbol{\sigma}_w), \boldsymbol{z}_2),$$

получаемые в результате подстановки констант $g(\boldsymbol{\sigma}_u)$, $g(\boldsymbol{\sigma}_v)$, $g(\boldsymbol{\sigma}_w)$ и зависящие только от переменных множества Z_2. Эти функции попарно различны, поскольку они представляются различными строками карты декомпозиции. Отсюда должно следовать, что константы $g(\boldsymbol{\sigma}_u)$, $g(\boldsymbol{\sigma}_v)$ и $g(\boldsymbol{\sigma}_w)$ тоже попарно различны, а это невозможно, так как имеется всего две константы: 0 и 1.

Пусть теперь в карте декомпозиции для некоторой функции $f(\boldsymbol{x})$ и множеств переменных Z_1 и Z_2 имеется только два различных значения строк. Пусть строки с одним значением составляют множество C_1, а строки с другим значением – множество C_2. Определим функцию $g(\boldsymbol{z}_1)$ так, что $g(\boldsymbol{z}_1) = 1$ на всех значениях векторной переменной $\boldsymbol{z}_1$, соответствующих строкам из множества C_1, и $g(\boldsymbol{z}_1) = 0$ – на остальных ее значениях. Можно построить используемое табличное представление с двумя строками для функции от $|Z_2| + 1$ переменных, где одна строка совпадает со всеми строками из множества C_1, а вторая строка совпадает со всеми строками из множества C_2. Эта функция является искомой функцией φ. Легко убедиться, что для любых наборов значений переменных из множества X значения функций f и φ совпадают.

Пусть, например, для функции, представленной в табл. 21.1, задано разбиение (Z_1, Z_2) на множестве ее аргументов, где $Z_1 = \{x_1, x_2, x_3\}$ и $Z_2 = \{x_4, x_5\}$. Требуется найти разложение функции $f(\boldsymbol{x})$ в виде

$$f(x_1, x_2, x_3, x_4, x_5) = \varphi(g(x_1, x_2, x_3), x_4, x_5)\,.$$

Табл. 21.1 является картой декомпозиции этой функции для $Z_1 = \{x_1, x_2, x_3\}$ и $Z_2 = \{x_4, x_5\}$.

Таблица 21.1

Задание функции $f(x_1, x_2, x_3, x_4, x_5)$

	x_4, x_5			
x_1, x_2, x_3	00	01	10	11
0 0 0	1	1	0	0
0 0 1	0	1	1	1
0 1 0	0	1	1	1
0 1 1	0	1	1	1
1 0 0	0	1	1	1
1 0 1	0	1	1	1
1 1 0	1	1	0	0
1 1 1	1	1	0	0

Нетрудно заметить, что функция $f(X)$ допускает заданную декомпозицию, так как имеется всего два вида строк. На наборах значений переменных x_1, x_2, x_3, соответствующих строкам вида

1	1	0	0

,

зададим функцию $g(x_1, x_2, x_3) = 1$, на остальных наборах – $g(x_1, x_2, x_3) = 0$. Тогда

$$g(x_1, x_2, x_3) = \bar{x}_1\ \bar{x}_2\ \bar{x}_3 \vee x_1\, x_2.$$

Из табл. 21.2 получаем представление функции φ в виде следующей ДНФ:

$$\varphi(g, x_4, x_5) = g\ \bar{x}_4 \vee \bar{g}\, x_4 \vee \bar{x}_4 x_5.$$

Таблица 21.2

Задание функции $\varphi(g, x_4, x_5)$

g	x_4, x_5 00	01	10	11
0	0	1	1	1
1	1	1	0	0

21.2. Двухблочная разделительная декомпозиция не полностью определенных булевых функций

Эта задача состоит в том, чтобы для заданной не полностью определенной функции $f(\boldsymbol{x})$ и разбиения множества аргументов (Z_1, Z_2) найти суперпозицию $\varphi(g(\mathbf{z}_1), \mathbf{z}_2)$, реализующую $f(\boldsymbol{x})$.

Очевидно, всякая не полностью определенная булева функция допускает двухблочную разделительную декомпозицию, если ее можно доопределить до функции, для которой существует искомая суперпозиция. Пусть, например, табл. 21.3 представляет заданную не полностью определенную булеву функцию $f(x_1, x_2, x_3, x_4)$. Эта же таблица является картой декомпозиции для $Z_1 = \{x_1, x_2\}$, $Z_2 = \{x_3, x_4\}$. Табл. 21.4 является картой декомпозиции результата доопределения заданной функции, допускающего двухблочную разделительную декомпозицию.

Таблица 21.3

Карта декомпозиции частичной функции

x_1, x_2	x_3, x_4 00	01	10	11
0 0	0	0	1	–
0 1	1	0	–	1
1 0	1	0	0	1
1 1	–	0	–	0

Таблица 21.4

Карта декомпозиции доопределения функции

x_1, x_2	x_3, x_4 00	01	10	11
0 0	0	0	1	0
0 1	1	0	0	1
1 0	1	0	0	1
1 1	0	0	1	0

Так же, как в предыдущем примере, определим функцию g следующим образом:

$$g(x_1, x_2) = x_1 x_2 \vee \bar{x}_1\ \bar{x}_2\,.$$

Для функции φ получим табл. 21.5, по которой найдем ДНФ данной функции:

$$\varphi(g, x_3, x_4) = g\, x_3\, \bar{x}_4 \vee \bar{g}\, x_3\, x_4 \vee \bar{g}\, \bar{x}_3\, \bar{x}_4 .$$

Строки карты декомпозиции можно рассматривать как троичные векторы. Например, карту декомпозиции не полностью определенной функции, показанную на табл. 21.3, можно представить в виде матрицы

$$\begin{bmatrix} 0 & 0 & 1 & - \\ 1 & 0 & - & 1 \\ 1 & 0 & 0 & 1 \\ - & 0 & - & 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} .$$

Таблица 21.5

Задание функции $\varphi(g, x_3, x_4)$

	x_3, x_4			
g	00	01	10	11
0	1	0	0	1
1	0	0	1	0

Построим граф G отношения ортогональности на множестве строк карты декомпозиции. Для нашего примера – это граф, изображенный на рис. 21.2. Его вершины соответствуют строкам, и две вершины связаны ребром, если и только если соответствующие строки ортогональны.

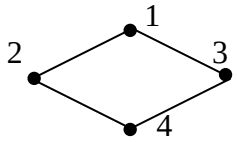


Рис. 21.2. Граф ортогональности строк карты декомпозиции

У т в е р ж д е н и е 21.2. Не полностью определенная булева функция допускает двухблочную разделительную декомпозицию, если и только если граф ортогональности строк ее карты декомпозиции является бихроматическим.

Если построенный для заданной функции граф G является бихроматическим, раскрасим его вершины в два цвета. Одному цвету припишем константу 0, другому – константу 1, определив таким образом функцию g на значениях векторной переменной $\boldsymbol{z}_1$, кодирующих соответствующие строки карты декомпозиции.

Для строк, соответствующих вершинам одного цвета, построим вектор, каждую компоненту которого определим следующим образом: если i-я компонента хотя бы одной из этих строк имеет определенное значение (0 либо

1), то это же значение припишем i-й компоненте получаемого вектора. В противном случае данной компоненте припишем значение «–». Полученные векторы представляют строки таблицы, задающей функцию φ.

В рассматриваемом примере одноцветным вершинам 1 и 4 соответствует вектор 0 0 1 0, а другим одноцветным вершинам, 2 и 3, – вектор 1 0 0 1. Приведенное выше решение получим, если искомой функции g на наборах значений переменных x_1 и x_2, соответствующих строкам 1 и 4, припишем значение 1, а на остальных наборах – значение 0.

21.3. Многоблочные разделительные декомпозиции

Результатом многоблочной декомпозиции является множество функций, суперпозиция которых представляет некоторую многоблочную структуру. При многоблочной разделительной декомпозиции задается разбиение множества X аргументов исходной функции $f(\boldsymbol{x})$ на более чем два подмножества: $X = Z_1 \cup Z_2 \cup \ldots \cup Z_m$, $Z_i \cap Z_j = \varnothing$, $i \neq j$. Переменные из множеств $Z_1, Z_2, \ldots, Z_m$ являются компонентами соответствующих булевых векторов $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m$. Рассмотрим два вида многоблочной разделительной декомпозиции: последовательную и параллельную.

Последовательная разделительная декомпозиция представляет собой многократную двухблочную декомпозицию, выполняемую следующим образом. Сначала получается суперпозиция

$$f(\boldsymbol{x}) = \varphi(g(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{m-1}), \boldsymbol{z}_m),$$

затем разлагается функция g:

$$g(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{m-1}) = \varphi'(g'(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{m-2}), \boldsymbol{z}_{m-1})$$

и т. д. В результате получается суперпозиция следующего вида (если, конечно, соответствующие функции допускают двухблочную разделительную декомпозицию):

$$f(\boldsymbol{x}) = \varphi(g_{m-1}(g_{m-2}(\ldots(g_1(\boldsymbol{z}_1), \boldsymbol{z}_2)\ldots), \boldsymbol{z}_{m-1}), \boldsymbol{z}_m),$$

что соответствует структуре, изображенной на рис. 2.3.

Пусть, например, область единичных значений полностью определенной булевой функции $f(x_1, x_2, x_3, x_4)$ составляют следующие наборы значений переменных:

x_1	x_2	x_3	x_4
0	0	0	1
0	0	1	1

0 1 0 0
0 1 1 1
1 0 0 0
1 0 1 1
1 1 0 1
1 1 1 1.

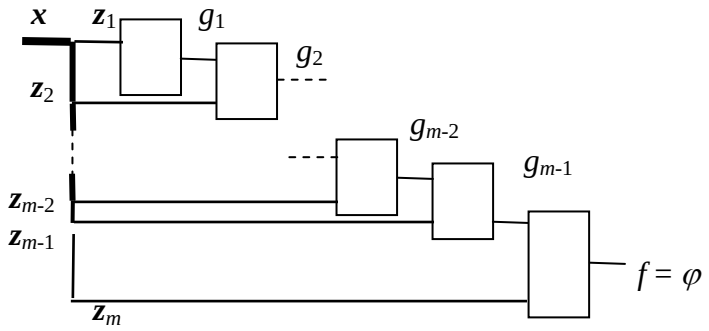


Рис. 21.3. Структура при последовательной разделительной декомпозиции

Требуется получить последовательную разделительную декомпозицию по разбиению

$$Z_1 = \{x_1, x_2\}, Z_2 = \{x_3\}, Z_3 = \{x_4\}.$$

Приведенная в табл. 21.6 карта декомпозиции для $\{x_1, x_2, x_3\}$, $\{x_4\}$ говорит о существовании суперпозиции

$$f(x_1, x_2, x_3, x_4) = \varphi'(g(x_1, x_2, x_3), x_4).$$

Табл. 21.7 задает функцию φ'. Таким образом, имеем

Таблица 21.6

Карта декомпозиции для функции $f(x_1, x_2, x_3, x_4)$

x_1, x_2, x_3	x_4 = 0	x_4 = 1
0 0 0	0	1
0 0 1	0	1
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1
1 1 1	0	1

Таблица 21.7

Задание функции φ'

g	x_4 = 0	x_4 = 1
0	0	1
1	1	0

$$g(x_1, x_2, x_3) = x_1\ \bar{x}_2\ \bar{x}_3 \vee \bar{x}_1 x_2\ \bar{x}_3,$$
$$\varphi'(g, x_4) = g\ \bar{x}_4 \vee \bar{g}\, x_4.$$

Для функции g и разбиения (Z_1, Z_2) карта декомпозиции представлена в виде табл. 21.8.

Таблица 21.8

Карта декомпозиции для функции g

	x_3	
$x_1\ x_2$	0	1
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	0

Окончательно получаем

$$f(x_1, x_2, x_3, x_4) = \varphi(g_2(g_1(x_1, x_2), x_3), x_4) = g_2\ \bar{x}_4 \vee \bar{g}_2 x_4;$$
$$g_2(g_1, x_3) = g_1\ \bar{x}_3;$$
$$g_1(x_1, x_2) = x_1\ \bar{x}_2 \vee \bar{x}_1 x_2\,.$$

Параллельная разделительная декомпозиция по разбиению $(Z_1, Z_2, \dots, Z_m)$ приводит функцию $f(\boldsymbol{x})$ к виду

$$f(\boldsymbol{x}) = \varphi(g_1(\mathbf{z}_1), g_2(\mathbf{z}_2), \dots, g_m(\mathbf{z}_m)). \qquad (21.1)$$

Эта форма соответствует схеме, приведенной на рис. 21.4.

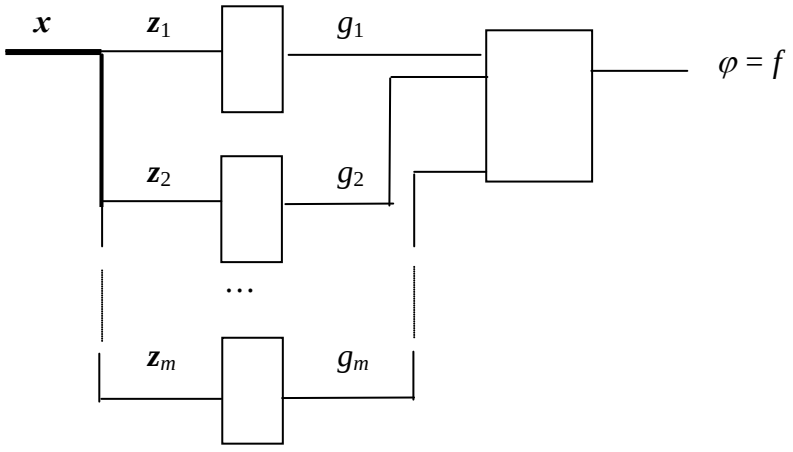


Рис. 21.4. Структура, соответствующая суперпозиции $f(\boldsymbol{x}) = \varphi(g_1, g_2, \dots, g_m)$

У т в е р ж д е н и е 21.3. Булева функция $f(\boldsymbol{x})$ допускает параллельную разделительную декомпозицию вида (21.1) тогда и только тогда, когда она допускает двухблочные разделительные декомпозиции вида

$$f(\boldsymbol{x}) = \varphi_1(g_1(\boldsymbol{z}_1), \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m);$$
$$f(\boldsymbol{x}) = \varphi_2(\boldsymbol{z}_1, g_2(\boldsymbol{z}_2), \ldots, \boldsymbol{z}_m);$$
$$\ldots$$
$$f(\boldsymbol{x}) = \varphi_m(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, g_m(\boldsymbol{z}_m)).$$

Пусть функция $f(\boldsymbol{x})$ задана с помощью табл. 21.9. Эта таблица является картой декомпозиции для разбиения $(\{x_1, x_2, x_3\}, \{x_4, x_5, x_6\})$. Среди ее строк имеется только два различных значения, следовательно, заданная функция допускает декомпозицию вида

$$f(\boldsymbol{x}) = \varphi_1(g_1(x_1, x_2, x_3), x_4, x_5, x_6),$$

а если строки заменить одноименными столбцами, то убедимся, что существует суперпозиция

$$f(\boldsymbol{x}) = \varphi_2(x_1, x_2, x_3, g_2(x_4, x_5, x_6)).$$

Таблица 21.9

Задание функции $f(\boldsymbol{x})$

$x_1x_2x_3$ \ $x_4x_5x_6$	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0 0 0	1	0	1	1	1	0	1	0
0 0 1	1	0	1	1	1	0	1	0
0 1 0	0	1	0	0	0	1	0	1
0 1 1	0	1	0	0	0	1	0	1
1 0 0	0	1	0	0	0	1	0	1
1 0 1	0	1	0	0	0	1	0	1
1 1 0	0	1	0	0	0	1	0	1
1 1 1	1	0	1	1	1	0	1	0

Таким образом, заданная функция $f(\boldsymbol{x})$ допускает параллельную разделительную декомпозицию вида $f(x) = \varphi(g_1, g_2)$.

Определим функцию g_1 как

$$g_1(x_1, x_2, x_3) = \bar{x}_1\ \bar{x}_2 \vee x_1 x_2 x_3.$$

Тогда табл. 21.10 будет задавать функцию φ_1. По этой таблице получим

$$g_2(x_4, x_5, x_6) = \bar{x}_6 \vee \bar{x}_4 x_5.$$

После соответствующего кодирования столбцов значениями переменной g_2 будем иметь

$$\varphi(g_1, g_2) = \bar{g}_1 g_2 \vee g_1 \bar{g}_2.$$

Таблица 21.10

Задание функции $\varphi_1(g_1, x_4, x_5, x_6)$

	$x_4x_5x_6$							
g_1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0	0	1	0	0	0	1	0	1
1	1	0	1	1	1	0	1	0

21.4. Неразделительная декомпозиция

При неразделительной декомпозиции множества аргументов $Z_1, Z_2, \ldots, Z_m$ могут пересекаться. Рассмотрим двухблочную неразделительную декомпозицию

$$f(\boldsymbol{x}) = \varphi(g(\mathbf{z}_1), \mathbf{z}_2),$$

где компонентами векторов $\boldsymbol{z}_1$ и $\boldsymbol{z}_2$ являются переменные из множеств Z_1 и Z_2 соответственно, причем

$$Z_1 \cup Z_2 = X, \quad Z_1 \cap Z_2 \neq \varnothing.$$

Естественно, имеет смысл решать задачу такой декомпозиции только при $|Z_1| < |X|$ и $|Z_2| + 1 < |X|$.

Особенностью карты декомпозиции для такой задачи является то, что некоторые ее элементы находятся на пересечении строки и столбца, в кодах которых одна и та же переменная (принадлежащая как Z_1, так и Z_2) имеет противоположные значения. Значение функции для такого элемента не определено. Это можно представить как внесение новых аргументов, число которых равно $|Z_1 \cap Z_2|$. При этом каждый из них связан с каким-либо из заданных аргументов так, что он не может принимать неодинаковое с ним значение. Таким образом, данную задачу можно решить как задачу разделительной декомпозиции не полностью определенной булевой функции.

У т в е р ж д е н и е 21.4. Булева функция $f(\boldsymbol{x})$ допускает двухблочную неразделительную декомпозицию вида $f(\boldsymbol{x}) = \varphi(g(\mathbf{z}_1), \mathbf{z}_2)$ тогда и только тогда,

когда граф ортогональности строк карты декомпозиции для (Z_1, Z_2) является бихроматическим.

Это справедливо как для полностью, так и для не полностью определенных функций.

Пусть функция $f(x_1, x_2, x_3, x_4, x_5)$ имеет значение 1 на следующих наборах значений переменных:

$x_1\ x_2\ x_3\ x_4\ x_5$

0 0 0 0 0
0 0 0 1 0
0 0 1 1 0
0 0 1 1 1
0 1 0 0 0
0 1 0 0 1
0 1 1 0 0
0 1 1 0 1
1 0 0 0 0
1 0 0 1 0
1 0 1 1 0
1 0 1 1 1
1 1 0 0 0
1 1 0 0 1
1 1 1 0 0.

Положим, что для нее требуется получить суперпозицию

$$f(x_1, x_2, x_3, x_4, x_5) = \varphi(g(x_1, x_2, x_3), x_2, x_4, x_5).$$

Карта декомпозиции для такого случая представлена в табл. 2.11.

Таблица 21.11

Карта декомпозиции для $f(x_1, x_2, x_3, x_4, x_5) = \varphi(g(x_1, x_2, x_3), x_2, x_4, x_5)$

$x_1x_2x_3$ \ $x_2x_4x_5$	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0 0 0	1	0	1	0	–	–	–	–
0 0 1	0	0	1	1	–	–	–	–
0 1 0	–	–	–	–	1	1	0	0
0 1 1	–	–	–	–	1	1	0	0
1 0 0	1	0	1	0	–	–	–	–
1 0 1	0	0	1	1	–	–	–	–
1 1 0	–	–	–	–	1	1	0	0
1 1 1	–	–	–	–	1	0	0	0

Граф ортогональности строк данной карты декомпозиции изображен на рис. 21.5, где вершины обозначены кодами соответствующих строк карты декомпозиции, а в скобках показана раскраска вершин графа – цвет «0» и цвет «1». Считая данные коды наборами значений аргументов x_1, x_2, x_3 функции g, а величины в скобках – ее соответствующими значениями, получим

$$g(x_1, x_2, x_3) = \bar{x}_2 x_3 \vee \bar{x}_1 x_2 \vee x_2 \bar{x}_3.$$

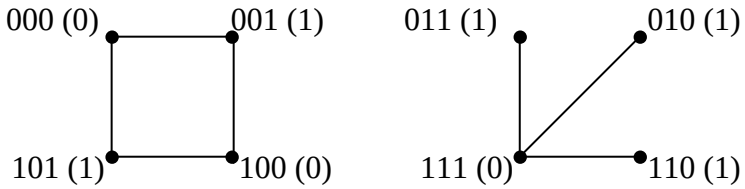


Рис. 21.5. Граф ортогональности строк карты декомпозиции

По табл. 21.12, задающей функцию φ, получим ДНФ этой функции:

$$\varphi(g, x_2, x_4, x_5) = g x_2 \bar{x}_4 \vee g \bar{x}_2 x_4 \vee \bar{g} \bar{x}_4 \bar{x}_5 \vee \bar{g} \bar{x}_2 \bar{x}_5.$$

Многоблочные неразделительные декомпозиции получаются аналогично двухблочным неразделительным декомпозициям.

Таблица 21.12

Задание функции $\varphi(g, x_2, x_4, x_5)$

g_1	$x_2x_4x_5$ 0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0	1	0	1	0	1	0	0	0
1	0	0	1	1	1	1	0	0

21.5. Декомпозиция систем булевых функций

Рассмотрим следующую задачу. Пусть задана система не полностью определенных булевых функций $f_1, f_2, \ldots, f_m$ от общего множества аргументов $X = \{x_1, x_2, \ldots, x_n\}$. Для заданной пары подмножеств (Z_1, Z_2) множества X, такой, что $X = Z_1 \cup Z_2$, требуется найти суперпозиции

$$f_i(\mathbf{x}) = \varphi_i(g_1(\mathbf{z}_1), g_2(\mathbf{z}_1), \ldots, g_k(\mathbf{z}_1), \mathbf{z}_2), \quad i = 1, 2, \ldots, m, \qquad (21.2)$$

где $\mathbf{z}_1$ и $\mathbf{z}_1$ – векторы, компонентами которых служат переменные из множеств Z_1 и Z_2, причем

$$k < |Z_1|, \quad k + |Z_2| < n$$

и не важно, пересекаются или нет подмножества Z_1 и Z_2.

В векторной форме данную задачу сформулируем следующим образом: для системы функций $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ требуется найти суперпозицию

$$\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2), \boldsymbol{w} = \boldsymbol{g}(\boldsymbol{z}_1),$$

такую, чтобы число компонент вектора $\boldsymbol{w}$ было меньше числа компонент вектора $\boldsymbol{z}_1$, а сумма чисел компонент векторов $\boldsymbol{w}$ и $\boldsymbol{z}_2$ была меньше длины вектора $\boldsymbol{x}$. Дополнительно можно еще поставить задачу минимизации числа компонент вектора $\boldsymbol{w}$.

Эта задача имеет приложение в синтезе логических схем из блоков, реализующих достаточно сложные системы булевых функций, с ограниченным числом внешних полюсов.

Для решения рассматриваемой задачи используется карта декомпозиции. Так же, как и для одной функции, строки ее соответствуют значениям векторной переменной $\boldsymbol{z}_1$, столбцы – значениям векторной переменной $\boldsymbol{z}_2$. На пересечении строки и столбца помещается значение вектора $\boldsymbol{y}$ (компоненты которого задают значения, которые принимают функции системы на соответствующих наборах значений аргументов). Ввиду неполной определенности функций здесь не обязательно должно быть $2^{|Z_1|}$ строки и $2^{|Z_2|}$ столбца.

Каждую строку карты декомпозиции рассматриваем как троичный вектор, составленный из векторов значений функций.

У т в е р ж д е н и е 21.5. Система не полностью определенных булевых функций $f_1, f_2, \ldots, f_m$ допускает декомпозицию в форме (21.2) тогда и только тогда, когда хроматическое число графа ортогональности строк ее карты декомпозиции для (Z_1, Z_2) не превышает 2^k.

Если удастся раскрасить упомянутый граф в 2^k цвета, то закодируем строки карты декомпозиции значениями векторной переменной

$$\boldsymbol{w} = \boldsymbol{g}(\boldsymbol{z}_1) = (g_1(\boldsymbol{z}_1), g_2(\boldsymbol{z}_1), \ldots, g_k(\boldsymbol{z}_1))$$

так, чтобы строки, соответствующие одноцветным вершинам, были закодированы одним кодом, а строки, соответствующие вершинам различных цветов, – различными кодами. Таким образом, значениям векторной переменной $\boldsymbol{z}_1$ ставятся в соответствие значения векторной переменной $\boldsymbol{w}$, т. е. построена система функций $g_1(\boldsymbol{z}_1), g_2(\boldsymbol{z}_1), \ldots, g_k(\boldsymbol{z}_1)$.

Карту декомпозиции, строкам которой приписаны значения переменной $\boldsymbol{w}$, а столбцам – значения векторной переменной $\boldsymbol{z}_2$, можно рассматривать как представление системы функций $\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$.

Пусть, например, задана система слабо определенных булевых функций

$$y_1 = f_1(\boldsymbol{x}), y_2 = f_2(\boldsymbol{x}), y_3 = f_3(\boldsymbol{x}),$$

где $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5, x_6)$, и заданы векторные переменные

$$\boldsymbol{z}_1 = (x_1, x_2, x_3, x_4), \quad \boldsymbol{z}_2 = \{x_5, x_6\}.$$

Заданная система представлена в виде пары булевых матриц, одна из которых содержит наборы значений аргументов (значения вектора $\boldsymbol{x}$), другая представляет значения функций на этих наборах (значения вектора $\boldsymbol{y}$). Считается, что на всех наборах значений аргументов, которые не присутствуют в соответствующей матрице, все функции заданной системы не определены. Матрицы имеют следующий вид:

$$\begin{array}{c} x_1\ x_2\ x_3\ x_4\ x_5\ x_6 \\ \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right] \end{array}, \quad \begin{array}{c} y_1\ y_2\ y_3 \\ \left[\begin{array}{ccc} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{array}\right] \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array}.$$

Карта декомпозиции системы булевых функций приведена в табл. 21.13. Граф ортогональности строк матрицы изображен на рис. 2.7. У вершин графа показаны наборы значений переменных x_1, x_2, x_3, x_4, которые приписаны соответствующим строкам карты декомпозиции.

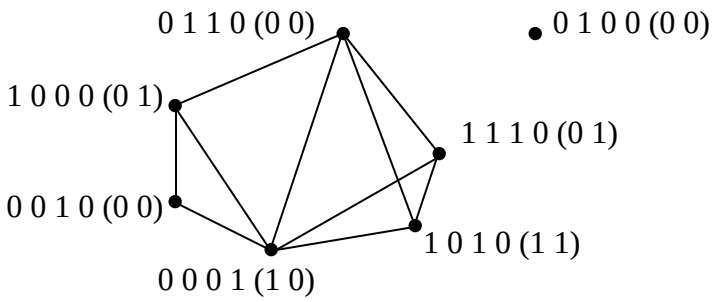


Рис. 21.6. Граф ортогональности строк карты декомпозиции

Таблица 21.13

Карта декомпозиции системы слабо определенных булевых функций
$y_1 = f_1(\boldsymbol{x}),\ y_2 = f_2(\boldsymbol{x}),\ y_3 = f_3(\boldsymbol{x})$

$x_1\,x_2\,x_3\,x_4$ \ $x_5\,x_6$	0 0	0 1	1 1	1 0
1 0 0 0	0 1 1	1 0 0	– – –	– – –
0 1 0 0	– – –	– – –	0 1 0	– – –
0 1 1 0	0 0 1	– – –	– – –	1 0 1
0 0 0 1	– – –	1 0 1	– – –	1 1 0
1 1 1 0	– – –	– – –	– – –	0 1 1
0 0 1 0	– – –	0 0 1	– – –	– – –
1 0 1 0	– – –	– – –	– – –	1 0 0

Граф раскрашивается не менее чем в четыре цвета, поскольку в нем легко заметен полный четырехвершинный подграф, все вершины которого должны быть раскрашены в разные цвета. В скобках приведены коды цветов в виде двухкомпонентных булевых векторов $\boldsymbol{w} = (w_1, w_2)$. В то же время четыре цвета оказываются достаточными – в результате находится схема, показанная на рис. 21.7.

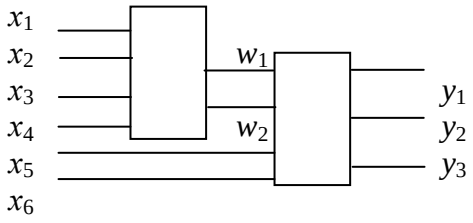


Рис. 21.7. Схема, реализующая систему из трех функций от шести переменных

Систему функций $w_1 = g_1(x_1, x_2, x_3, x_4)$, $w_2 = g_2(x_1, x_2, x_3, x_4)$ представим следующими матрицами:

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix} \qquad \begin{matrix} w_1 & w_2 \end{matrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix}.$$

Табл. 21.14, полученную из табл. 21.13 совмещением неортогональных одноцветных строк, можно считать заданием системы функций $\boldsymbol{y = \varphi(w, z_2)}$.

Таким образом, заданная система не полностью определенных булевых функций $\boldsymbol{y = f(x)}$ разложена на две системы:

$$\boldsymbol{w = g(z_1)} \text{ и } \boldsymbol{y = \varphi(w, z_2)}.$$

Последнюю представим парой матриц:

$$\begin{matrix} w_1 & w_2 & x_5 & x_6 \end{matrix} \qquad \begin{matrix} y_1 & y_2 & y_3 & \end{matrix}$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{matrix}.$$

Таблица 21.14

Задание системы булевых функций $\boldsymbol{y = \varphi(w, z_2)}$

$w_1\ w_2$ \ $x_5\ x_6$	0 0	0 1	1 1	1 0
0 1	0 1 1	1 0 0	– – –	0 1 1
0 0	0 0 1	0 0 1	0 1 0	1 0 1
1 0	– – –	1 0 1	– – –	1 1 0
1 1	– – –	– – –	– – –	1 0 0

После минимизации получим следующие ДНФ:

$w_1 = x_1\ \bar{x}_2\ x_3 \vee x_4$;

$w_2 = x_1$;

$y_1 = \bar{w}_2\ x_5\ \bar{x}_6 \vee w_1 \vee w_2\ x_6$;

$y_2 = x_5\ x_6 \vee\ \bar{w}_1\ w_2\ \bar{x}_6 \vee w_1\ \bar{w}_2\ x_5$;

$y_3 = \bar{w}_1\ \bar{x}_6 \vee\ \bar{w}_2\ \bar{x}_5$.

Г л а в а 22

Конечный автомат. Типы

22.1. Автомат с памятью

До сих пор рассматривались устройства, не обладающие памятью. Такие устройства в любой момент времени на любую комбинацию входных двоичных сигналов реагируют однозначно, т. е. каждой комбинации двоичных сигналов на входе устройства соответствует определенная комбинация двоичных сигналов на выходе независимо от того, какие сигналы приходили на вход раньше. Отсюда происходят названия «*комбинационный автомат*» или «*комбинационная схема*».

Как отмечено выше, общепринятой моделью поведения такого устройства является система булевых функций. Комбинации двоичных сигналов иногда удобно обозначать абстрактными символами или буквами некоторого заданного алфавита. Пусть имеется входной алфавит $A = \{a_1, a_2, \dots, a_\alpha\}$ и выходной алфавит $B = \{b_1, b_2, \dots, b_\beta\}$. Тогда общий вид описания поведения комбинационного автомата можно представить табл. 22.1, где $b_{i_j} \in B$. Эта таблица является аналогом таблицы истинности. Она полностью определяет алгоритм функционирования заданного устройства.

Таблица 22.1

Пример задания комбинационного автомата

Вход	Выход
a_1	b_{i_1}
a_2	b_{i_2}
…	…
a_α	b_{i_α}

Часто алгоритм функционирования устройства бывает таким, что простым соответствием вида табл. 22.1 его задать невозможно. Для его описания необходимо ввести время, и работа устройства рассматривается во времени. Время носит дискретный характер, т. е. течение времени представляется как последовательность моментов. В любой момент времени устройство принимает сигнал на входе и выдает сигнал на выходе. В различные моменты устройство может реагировать на один и тот же входной сигнал по-разному. Пусть, например, на вход устройства с входным и выходным алфавитами $A = \{a_1, a_2, a_3, a_4\}$ и $B = \{b_1, b_2, b_3\}$ пришла последовательность сигналов

$$a_1\ a_1\ a_3\ a_2\ a_2\ a_4\ a_3,$$

а на выходе наблюдается последовательность сигналов

$$b_2\ b_3\ b_2\ b_1\ b_3\ b_2\ b_1.$$

В первый момент времени устройство на входной сигнал a_1 реагировало выходным сигналом b_2, а во второй момент на тот же сигнал – сигналом b_3. На сигналы a_2 и a_3 в разные моменты оно реагировало также по-разному, т. е. поведение устройства нельзя описать тем же способом, что использовался для описания поведения комбинационного автомата.

Эта неоднозначность устраняется, если ввести понятие *внутреннего состояния* или просто *состояния*. Тогда то, что устройство реагирует на один и тот же сигнал в разные моменты по-разному, можно объяснить так, что оно находится при этом в разных состояниях. Приняв входной сигнал, устройство выдает выходной сигнал и может перейти в другое состояние.

Состояния образуют множество $Q = \{q_1, q_2, \dots, q_\gamma\}$. Тогда алгоритм функционирования устройства можно представить совокупностью строк вида

$$(a_i, q_j) \to (q_s, b_t),$$

где $a_i \in A,\ b_t \in B,\ q_j, q_s \in Q$. Устройство, поведение которого можно задать в таком виде, носит название «*автомат с памятью*» или «*последовательностный автомат*». Последний термин заключает в себе смысл отображения одной последовательности в другую.

Соответствие $(a_i, q_j) \to (q_s, b_t)$ можно задать таблицей, подобной табл. 22.1, где фактически представлены две функции, определенные на множестве $A \times Q$. Областью значений одной функции является множество Q, другой функции – множество B.

Моделью описанного устройства является *конечный автомат*, представляющий собой совокупность следующих объектов:

$A = \{a_1, a_2, \dots, a_\alpha\}$ – множество входных символов, или входной алфавит;

$B = \{b_1, b_2, \dots, b_\beta\}$ – множество выходных символов, или выходной алфавит;

$Q = \{q_1, q_2, \dots, q_\gamma\}$ – множество состояний, или внутренний алфавит;

$\Psi : A \times Q \to Q$ – функция переходов;

$\Phi : A \times Q \to B$ – функция выходов.

Для конечного автомата используется обозначение (A, B, Q, Ψ, Φ). Слово «конечный» подчеркивает, что все три множества, входящие в состав данной модели, конечны. В том виде, в каком эта модель здесь представлена, она носит название «*автомат Мили*». Другой разновидностью данной модели является *автомат Мура*, отличающийся от автомата Мили только тем, что функция

выходов не зависит от входного символа, т. е. представляет собой отображение $\Phi : Q \to B$.

Значением функции $\Psi(a, q)$ является состояние q^+, в которое переходит автомат из состояния q, если на вход его подан символ a. Значением функции $\Phi(a, q)$ является выходной символ, выдаваемый автоматом в состоянии q при поступлении на его вход символа a, а значением функции $\Phi(q)$ автомата Мура – выходной символ b, который выдает автомат, находясь в состоянии q. Если значения функций $\Psi(a, q)$ и $\Phi(a, q)$ определены для любой пары значений аргументов a и q, а в модели Мура функция $\Phi(q)$ определена для всех значений q, то автомат является *полностью определенным*, или *полным* автоматом. Иногда приходится иметь дело с *не полностью определенным*, или *частичным* автоматом, у которого эти функции могут быть определены не везде.

Конечный автомат является абстрактной математической моделью дискретного устройства. Поведение, описанное данной моделью, может быть по-разному реализовано в дискретном устройстве. При *синхронной реализации* моменты времени, когда определяется состояние, в которое переходит устройство, а в случае автомата Мили и выходной символ, зафиксированы. Технически это осуществляется введением генератора синхронизирующих сигналов, которые инициируют соответствующие действия. Период следования таких сигналов не должен быть меньше чем время, необходимое для завершения переходных процессов в устройстве. Реализованный таким образом конечный автомат называется *синхронным автоматом*.

При *асинхронной реализации* указанные моменты времени не фиксированы. Они связаны с моментами, в которые происходит изменение входного сигнала. Реализованный таким образом автомат называется *асинхронным автоматом*. Если синхронный автомат различает последовательности разной длины из одинаковых входных символов, т. е. может реагировать на них различными выходными последовательностями, то асинхронный автомат воспринимает последовательность одинаковых входных символов любой длины как один символ. В схеме такого устройства генератор синхронизирующих сигналов отсутствует.

В асинхронном автомате переход из состояния в состояние происходит за некоторый конечный промежуток времени, в течение которого не должен меняться входной сигнал. При действии любого входного сигнала автомат приходит в некоторое устойчивое состояние, из которого он не выходит до конца действия данного сигнала. При этом должно выполняться требование *прямого перехода*, которое формально выражается следующим образом: если $\Psi(a, q_i) = q_j$, то $\Psi(a, q_j) = q_j$. Иногда допускается выполнение следующей цепочки переходов: $\Psi(a, q_i) = q_r$, $\Psi(a, q_r) = q_s$, … , $\Psi(a, q_t) = q_j$, $\Psi(a, q_j) = q_j$.

22.2. Представления автомата

Конечный автомат удобно представлять *таблицей переходов* и *таблицей выходов*. Строкам этих таблиц соответствуют состояния автомата, столбцам – входные символы. На пересечении строки, соответствующей состоянию q, и столбца, соответствующего входному символу a, в таблице переходов записывается значение $\Psi(a, q)$, а в таблице выходов – значение $\Phi(a, q)$. Другими словами, в первом случае в клетке таблицы указывается состояние, в которое автомат переходит из состояния q при поступлении на его вход символа a, а во втором случае – выходной символ, который при этом выдает автомат. Табл. 22.2 и табл. 22.3 представляют собой пример описанного представления автомата Мили, у которого $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2\}$ и $Q = \{q_1, q_2, q_3\}$.

Таблица 22.2

Функция Ψ

	a_1	a_2	a_3	a_4
q_1	q_1	q_2	q_1	q_2
q_2	q_3	q_1	q_3	q_1
q_3	q_3	q_1	q_1	q_1

Таблица 22.3

Функция Ф

	a_1	a_2	a_3	a_4
q_1	b_1	b_1	b_2	b_1
q_2	b_1	b_2	b_2	b_1
q_3	b_2	b_2	b_1	b_1

Зная начальное состояние автомата и входную последовательность, нетрудно получить по этим таблицам соответствующую последовательность выходных символов. Приведем пример такого соответствия для автомата, заданного табл. 22.2 и 22.3, на вход которого поступила последовательность символов $a_1, a_2, a_2, a_1, a_3, a_4, a_1, a_4$ при начальном состоянии q_1. Покажем также состояния, которые проходит автомат:

a_1	a_2	a_2	a_1	a_3	a_4	a_1	a_4
q_1	q_1	q_2	q_1	q_1	q_1	q_2	q_3
b_1	b_1	b_2	b_1	b_2	b_1	b_1	b_1.

Автомат Мура представляется одной таблицей переходов, к которой добавлен один столбец со значениями функции выходов (табл. 22.4).

Можно свести таблицу переходов и таблицу выходов автомата Мили в одну таблицу, которую называют *таблицей переходов и выходов*. Такая таблица для автомата, заданного в виде табл. 22.2 и табл. 22.3, имеет вид табл. 22.5.

Более наглядным при небольшом числе состояний является представление автомата в виде *графа поведения автомата*, который представляет собой ориентированный граф. Его вершины соответствуют состояниям автомата, а дуги – переходам между состояниями. При этом дуга помечается всеми входными символами, которые вызывают соответствующий переход, и выходными символами, сопровождающими данный переход (в случае автомата

Мили). В случае автомата Мура выходными символами помечаются вершины, соответствующие состояниям, в которых находится автомат при выдаче данных символов. На рис. 22.1 изображены графы переходов автоматов, заданных табл. 22.4 и 22.5.

Таблица 22.4

Таблица переходов автомата Мура

	a_1	a_2	a_3	a_4	Ф
q_1	q_1	q_2	q_1	q_2	b_1
q_2	q_3	q_1	q_3	q_1	b_1
q_3	q_3	q_1	q_1	q_1	b_2

Таблица 22.5

Таблица переходов и выходов автомата Мили

	a_1	a_2	a_3	a_4
q_1	q_1,b_1	q_2,b_1	q_1,b_2	q_2,b_1
q_2	q_3,b_1	q_1,b_2	q_3,b_2	q_1,b_1
q_3	q_3,b_2	q_1,b_2	q_1,b_1	q_1,b_1

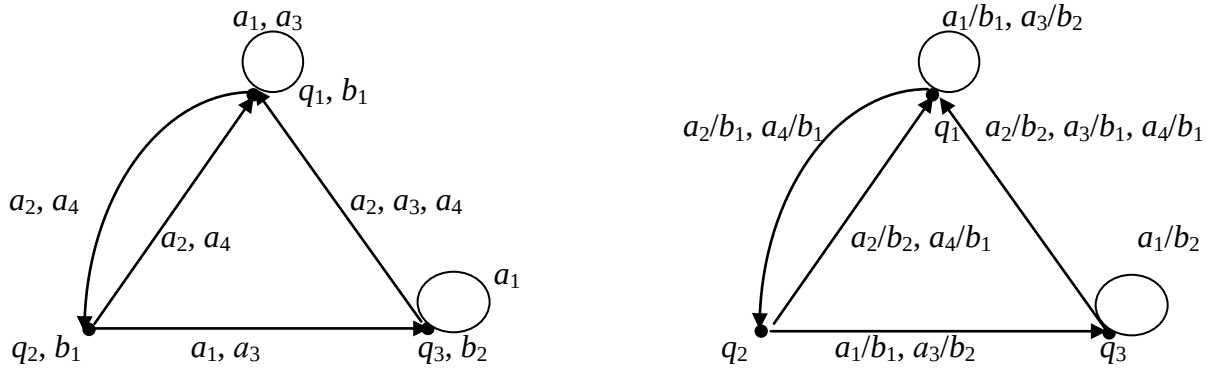


Рис. 22.1. Примеры графов поведения: *а)* автомата Мура; *б)* автомата Мили

Еще одним способом представления автомата является *матрица поведения*, представляющая собой квадратную матрицу, строки и столбцы которой помечаются состояниями автомата. В случае автомата Мура на пересечении строки q_i и столбца q_j матрицы поведения записываются входные символы, переводящие автомат из состояния q_i в состояние q_j, а строки помечаются также и выходными символами. В случае автомата Мили элементы матрицы поведения, кроме входных символов, вызывающих соответствующие переходы, содержат выходные символы, которые сопровождают эти переходы. Если из состояния q_i нет перехода в состояние q_j, то на пересечении строки q_i и столбца q_j ставится прочерк. Для рассмотренных автоматов ниже представлены матрицы поведения, причем первая из них – матрица поведения автомата Мура, вторая – матрица поведения автомата Мили.

$$\begin{array}{c} \\ q_1,b_1 \\ q_2,b_1 \\ q_3,b_2 \end{array} \begin{array}{c} \begin{array}{ccc} q_1 & q_2 & q_3 \end{array} \\ \begin{bmatrix} a_1,a_3 & a_2,a_4 & - \\ a_2,a_4 & - & a_1,a_3 \\ a_2,a_3,a_4 & - & a_1 \end{bmatrix} \end{array},$$

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \end{array} \begin{array}{c} \begin{array}{ccc} q_1 & q_2 & q_3 \end{array} \\ \begin{bmatrix} a_1/b_1,\ a_3/b_2 & a_2/b_1,\ a_4/b_1 & - \\ a_2/b_2,\ a_4/b_1 & - & a_1/b_1, a_3/b_2 \\ a_2/b_3,\ a_3/b_1,\ a_4/b_1 & - & a_1/b_2 \end{bmatrix} \end{array}.$$

22.3. Связь между моделями Мили и Мура

Всякое отображение входных последовательностей в выходные может быть реализовано как с помощью модели Мили, так и с помощью модели Мура. Определим преобразование, переводящее любой автомат Мили в эквивалентный ему автомат Мура, а также преобразование, переводящее любой автомат Мура в эквивалентный ему автомат Мили.

Пусть задан автомат Мура $M = (A, B, Q, \Psi, \Phi)$ и требуется получить эквивалентный ему автомат Мили $M^* = (A^*, B^*, Q^*, \Psi^*, \Phi^*)$.

Очевидно, $A^* = A$ и $B^* = B$. Положим $Q^* = Q$ и $\Psi^* = \Psi$, а Φ^* определим следующим образом. Пусть $\Psi(a, q) = q'$ и $\Phi(q') = b$, где $q, q' \in Q$, $a \in A$ и $b \in B$. Это означает, что автомат, будучи в состоянии q, отвечает на входной символ a выходным символом b, который выдается в следующий момент времени, когда автомат окажется в состоянии q'. Следовательно, можно считать, что $\Phi^*(a, q) = b$. Автомат Мура и эквивалентный ему автомат Мили представлены в табл. 22.6 и 22.7 соответственно.

Пусть теперь задан автомат Мили $M = (A, B, Q, \Psi, \Phi)$ и требуется получить эквивалентный ему автомат Мура $M^* = (A^*, B^*, Q^*, \Psi^*, \Phi^*)$. Как и в предыдущем случае, имеем $A^* = A$ и $B^* = B$. Определим Q^* следующим образом. Рассмотрим все такие пары вида (q, b), где $q \in Q$, $b \in B$, что для каждой (q, b) имеется такая пара (a, q'), что $\Psi(a, q') = q$ и $\Phi(a, q') = b$ $(a \in A,\ q' \in Q)$. Каждой паре (q, b) поставим в соответствие состояние $q^* \in Q^*$ и определим функции Ψ^* и Φ^* следующим образом:

$$\Psi^*(a, q^*) = \Psi^*(a, (q, b)) = (\Psi(a, q), \Phi(a, q));\ \ \Phi^*(q^*) = \Phi^*((q, b)) = b.$$

Таблица 22.6

	a_1	a_2	a_3	Φ
q_1	q_3	q_2	q_2	0
q_2	q_1	q_4	q_3	1
q_3	q_2	q_2	q_1	0
q_4	q_3	q_4	q_4	1

Таблица 22.7

	a_1	a_2	a_3
q_1	q_3,0	q_2,1	q_2,1
q_2	q_1,0	q_4,1	q_3,0
q_3	q_2,1	q_2,1	q_1,0
q_4	q_3,0	q_4,1	q_4,1

Если автомат имеет состояние, в которое он никогда не переходит (это может быть начальное состояние), то всякому такому состоянию ставится в

соответствие состояние автомата Мура, переходы из него определяются аналогично, а выходной символ при нем не определен.

Если автомат является частичным, то достаточно ввести новое состояние, соответствующее неопределенному состоянию, и новый выходной символ, соответствующий неопределенному выходному символу, и после описанных преобразований вернуться к неопределенному состоянию и неопределенному выходному символу. Переходы из такого состояния не определены. Автомат Мили и эквивалентный ему автомат Мура представлены в табл. 22.8 и 22.9 соответственно.

Таблица 22.8

	a_1	a_2	a_3	a_4
q_1	$-,b_1$	q_2,b_1	$-,-$	q_2,b_1
q_2	q_3,b_1	$-,b_2$	$q_3,-$	q_2,b_1
q_3	q_3,b_2	$-,-$	q_2,b_1	q_2,b_1

Таблица 22.9

			a_1	a_2	a_3	a_4	Φ^*
q_1	$\to$	q^*_1	q^*_6	q^*_2	$-$	q^*_2	$-$
q_2,b_1	$\to$	q^*_2	q^*_3	q^*_7	q^*_5	q^*_2	b_1
q_3,b_1	$\to$	q^*_3	q^*_4	$-$	q^*_2	q^*_2	b_1
q_3,b_2	$\to$	q^*_4	q^*_4	$-$	q^*_2	q^*_2	b_2
$q_3,-$	$\to$	q^*_5	q^*_4	$-$	q^*_2	q^*_2	$-$
$-,b_1$	$\to$	q^*_6	$-$	$-$	$-$	$-$	b_1
$-,b_2$	$\to$	q^*_7	$-$	$-$	$-$	$-$	b_2

22.4. Автомат с абстрактным состоянием. Булев автомат

Широко распространенным типом автомата является модель, описываемая одной многозначной внутренней переменной q и многими входными и выходными булевыми переменными $x_1, x_2, \dots, x_n$ и $y_1, y_2, \dots, y_m$. Поведение такого автомата задается системой уравнений

$$q^+ = \psi(x_1, x_2, \dots, x_n; q);$$
$$y_1 = \varphi_1(x_1, x_2, \dots, x_n; q);$$
$$y_2 = \varphi_2(x_1, x_2, \dots, x_n; q);$$
$$\dots$$
$$y_m = \varphi_m(x_1, x_2, \dots, x_n; q),$$

более компактно представляемой в векторной форме

$$q^+ = \boldsymbol{\psi}(\boldsymbol{x}, q);$$
$$\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{x}, q).$$

Функции $\boldsymbol{\psi}$ и $\boldsymbol{\varphi}$ отличаются от введенных ранее Ψ и Φ только тем, что многозначные входная и выходная переменные оказались замененными на соответствующие булевы векторы, но внутренняя переменная осталась многозначной.

Описанная модель называется *автоматом с абстрактным состоянием*. Ею удобно пользоваться на начальных этапах логического проектирования дискретных устройств, когда вход и выход устройства описываются как некоторые множества булевых переменных, имеющих конкретную техническую интерпретацию, в то время как множество внутренних переменных представляется пока в простейшей форме, в виде одной многозначной переменной q. Число значений переменной q полагается равным числу различных состояний автомата, при котором он может реализовать заданное функциональное отношение между входом и выходом.

Если заменить внутреннюю переменную q на соответствующий булев вектор $\boldsymbol{z} = (z_1, z_2, \ldots, z_k)$, то получится система уравнений, в которой все переменные и все функции оказываются булевыми:

$$
\begin{gathered}
z_1^+ = \psi_1(x_1, x_2, \ldots, x_n; z_1, z_2, \ldots, z_k);\\
z_2^+ = \psi_2(x_1, x_2, \ldots, x_n; z_1, z_2, \ldots, z_k);\\
\ldots\\
z_k^+ = \psi_k(x_1, x_2, \ldots, x_n; z_1, z_2, \ldots, z_k);\\
y_1 = \varphi_1(x_1, x_2, \ldots, x_n; z_1, z_2, \ldots, z_k);\\
y_2 = \varphi_2(x_1, x_2, \ldots, x_n; z_1, z_2, \ldots, z_k);\\
\ldots\\
y_m = \varphi_m(x_1, x_2, \ldots, x_n; z_1, z_2, \ldots, z_k).
\end{gathered}
$$

Эта модель называется *булевым автоматом*. Ее также можно представить в компактной векторной форме:

$$
\begin{gathered}
\boldsymbol{z}^+ = \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{z});\\
\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{x}, \boldsymbol{z}).
\end{gathered}
$$

Булев автомат в определенном смысле ближе к реальным дискретным устройствам, поскольку его переменные непосредственно реализуются физическими переменными устройства, в частности, на типичных для современной техники элементах с двумя устойчивыми состояниями. Векторы $\boldsymbol{x}$, $\boldsymbol{y}$ и $\boldsymbol{z}$ показывают структуру абстрактных символов a и b и состояния q. Приведенная выше система функций соответствует структуре, изображенной на рис. 22.2, где КС – комбинационная схема, реализующая приведенную выше систему, а П – блок памяти, осуществляющий задержку на период между соседними моментами времени.

Переменная z_i представляет состояние i-го двоичного элемента памяти, а выражение

$$
z_i^+ = \psi_i(x_1, x_2, \ldots, x_n; z_1, z_2, \ldots, z_k)
$$

надо понимать так, что состояние i-го элемента памяти определяется значениями входных символов и состояниями элементов памяти в предыдущий момент времени.

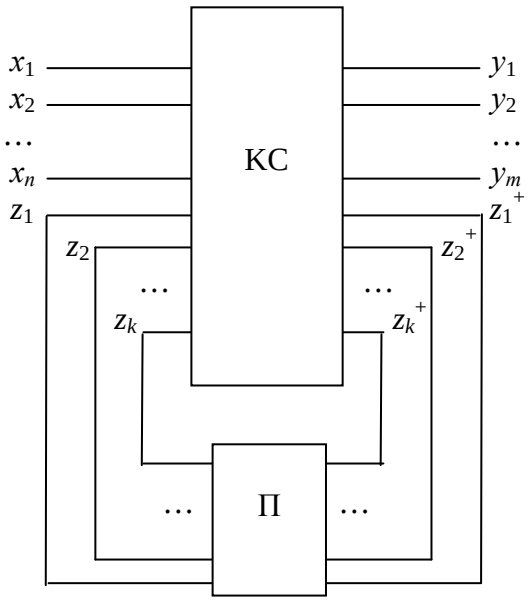


Рис. 22.2. Структура булева автомата

Г л а в а 23

Минимизация полных автоматов

23.1. Эквивалентность состояний. Постановка задачи минимизации

Одно и то же поведение может быть задано автоматами с различным числом состояний. Отсюда возникает задача *минимизации числа состояний автомата*, или просто *минимизации автомата*.

Основным понятием, используемым в данной задаче, является эквивалентность состояний автомата. Состояние q_i автомата M_1 и состояние q_j автомата M_2 *эквивалентны*, если автомат M_1 при начальном состоянии q_i и автомат M_2 при начальном состоянии q_j под воздействием любой входной последовательности выдают одинаковые последовательности на выходе. При этом M_1 и M_2 могут представлять собой один и тот же автомат.

Отношение эквивалентности состояний обладает свойствами рефлексивности (каждое состояние эквивалентно самому себе), симметричности (если состояние q_i эквивалентно состоянию q_j, то q_j эквивалентно q_i) и транзитивности (если состояние q_i эквивалентно состоянию q_k, а q_k эквивалентно состоянию q_j, то q_i эквивалентно q_j). Как известно, бинарное отношение на некотором множестве, обладающее такими свойствами, определяет разбиение данного множества на классы эквивалентности.

Автомат M_1 и автомат M_2 *эквивалентны*, если для каждого состояния одного из них имеется хотя бы одно эквивалентное ему состояние другого.

Задача минимизации полного автомата ставится следующим образом: для заданного автомата M найти эквивалентный ему автомат с минимальным числом состояний.

Пусть задан автомат $M = (A, B, Q, \Psi, \Phi)$ и требуется найти эквивалентный ему автомат $M' = (A, B, Q', \Psi', \Phi')$, обладающий минимальным числом состояний. Допустим, установлено отношение эквивалентности на множестве состояний автомата M и определены классы эквивалентности $S_1, S_2, \ldots, S_m$. Обозначим символом $q^{(i)}$ любое состояние, принадлежащее классу S_i. Тогда минимальный автомат строится следующим образом.

Формируется множество состояний $Q' = \{q'_1, q'_2, \ldots, q'_m\}$, где каждому q'_i поставлен во взаимно однозначное соответствие класс эквивалентности S_i.

Если для некоторых $q^{(i)} \in S_i$ и $a \in A$ имеет место $\Phi(a, q^{(i)}) = b$, где $b \in B$, то $\Phi'(a, q'_i) = b$.

Если для некоторых $q^{(i)} \in S_i$ и $a \in A$ имеет место $\Psi(a, q^{(i)}) = q^{(j)}$, где $q^{(j)} \in S_j$, то $\Psi'(a, q'_i) = q'_j$.

23.2. Установление эквивалентности состояний

В следующих трех случаях эквивалентность или неэквивалентность состояний определяется непосредственно по таблицам переходов и выходов.

1. Найдется такой входной сигнал $a \in A$, что $\Phi(a, q_i) \neq \Phi(a, q_j)$. Тогда состояния q_i и q_j *явно неэквивалентны*. В таблице выходов это соответствует столбцу, где элементы в строках q_i и q_j различны.

2. Для всех $a \in A$ имеют место $\Phi(a, q_i) = \Phi(a, q_j)$ и $\Psi(a, q_i) = \Psi(a, q_j)$. Здесь состояния q_i и q_j *явно эквивалентны*. В таблице переходов и таблице выходов строки q_i и q_j совпадают.

3. Строки q_i и q_j таблицы выходов совпадают, а строки q_i и q_j таблицы переходов совпадут после замены каждого значения q_i на q_j или каждого значения q_j на q_i (если, конечно, такие значения присутствуют в этих строках). В этом случае состояния q_i и q_j также являются явно эквивалентными.

Если состояния не являются явно эквивалентными или явно неэквивалентными, то для решения вопроса об их эквивалентности приходится применять более глубокий анализ.

Пусть $q_i \neq q_j$. *Цепью, порождаемой парой состояний* $\langle q_i, q_j \rangle$ полного автомата M, назовем множество C, элементами которого являются следующие пары: 1) сама пара $\langle q_i, q_j \rangle$; 2) если $\langle q_k, q_l \rangle \in C$, то все пары вида $\langle \Psi(a, q_k), \Psi(a, q_l) \rangle$, где $\Psi(a, q_k)$ и $\Psi(a, q_l)$ различны. Другие пары не входят в C.

Таким образом, цепь C, порождаемая парой состояний $\langle q_i, q_j \rangle$, содержит все пары состояний, в которые автомат переходит из состояний q_i и q_j под воздействием всевозможных входных последовательностей (конечных и бесконечных). Само название «цепь» связано с характером образования этого множества: реализуется процесс, подобный цепной реакции (появление одного элемента в цепи вызывает вовлечение множества других элементов). Справедливо следующее утверждение.

У т в е р ж д е н и е 23.1. Состояния q_i и q_j автомата M являются эквивалентными, если и только если в цепи, порождаемой парой состояний $\langle q_i, q_j \rangle$, нет ни одной пары явно неэквивалентных состояний. В этом случае все пары, принадлежащие данной цепи, являются парами эквивалентных состояний.

Действительно, пусть состояние q_i автомата M эквивалентно состоянию q_j, а цепь C, порождаемая парой $\langle q_i, q_j \rangle$, содержит пару явно неэквивалентных состояний, скажем, $\langle q_k, q_l \rangle$. Тогда существует входная последовательность, переводящая автомат M из состояний q_i и q_j в состояния q_k и q_l. Она вызывает различные выходные последовательности, что противоречит условию. С другой стороны, поскольку в C нет ни одной пары явно неэквивалентных состояний, выходные последовательности при одной и той же входной последовательности и при начальных состояниях q_i и q_j одинаковы.

Эквивалентность состояний можно представить *матрицей эквивалентности* – булевой матрицей, строки и столбцы которой соответствуют состояниям автомата. Элемент, расположенный на пересечении *i*-й строки и *j*-го столбца, имеет значение 1, если и только если состояния q_i и q_j эквивалентны.

Рассмотрим процесс минимизации полного автомата на примере (табл. 23.1).

Таблица 23.1

Таблица переходов и выходов минимизируемого автомата

	a_1	a_2	a_3
1	2,1	2,0	5,0
2	1,0	4,1	4,1
3	2,1	2,0	5,0
4	3,0	2,1	2,1
5	6,1	4,0	3,0
6	8,0	9,1	6,1
7	6,1	2,0	8,0
8	4,1	4,0	7,0
9	7,0	9,1	7,1

Сначала находим явно эквивалентные и явно неэквивалентные состояния. В силу рефлексивности и симметричности отношения эквивалентности достаточно иметь только часть матрицы эквивалентности, расположенную выше (или ниже) главной диагонали. Получим следующую матрицу, пустые элементы в верхней части которой соответствуют парам. Для этих пар непосредственно из таблицы переходов и выходов не видно, эквивалентны в них состояния или нет, т. е. не являются ни явно эквивалентными, ни явно неэквивалентными:

$$
\begin{array}{cccccccc}
2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\left[\begin{array}{cccccccc}
0 & 1 & 0 & & 0 & & & 0 \\
 & 0 & & 0 & & 0 & 0 & \\
 & & 0 & & 0 & & & 0 \\
 & & & 0 & & 0 & 0 & \\
 & & & & 0 & & & 0 \\
 & & & & & 0 & 0 & \\
 & & & & & & & 0 \\
 & & & & & & & 0
\end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array}
\end{array}.
$$

Для пар состояний, которым соответствуют пустые места в этой матрице, надо строить порождаемые ими цепи, которые удобно представлять ориентированными графами. Вершинам такого графа соответствуют пары,

принадлежащие формируемой цепи. Из вершины $\langle q_i, q_j \rangle$ в вершину $\langle q_k, q_l \rangle$ направлена дуга, если $\langle q_k, q_l \rangle = \langle \Psi(a, q_i), \Psi(a, q_j) \rangle$. Не обязательно строить всю цепь, если порождающая ее пара не является эквивалентной. Можно прекратить процесс после появления первой пары, которой соответствует нулевое значение элемента матрицы эквивалентности. Тогда неэквивалентными объявляются все пары, лежащие на пути к данной паре от порождающей пары.

Часть цепи, порождаемой парой ⟨1, 5⟩, изображена на рис. 23.1. Она содержит пару явно неэквивалентных состояний ⟨2, 7⟩. Дальше цепь можно не строить, так как ясно, что пара ⟨1, 5⟩ оказалась неэквивалентной и, кроме нее, неэквивалентными оказались пары ⟨4, 9⟩ и ⟨2, 6⟩, лежащие на пути из вершины ⟨1, 5⟩ к вершине ⟨2, 7⟩.

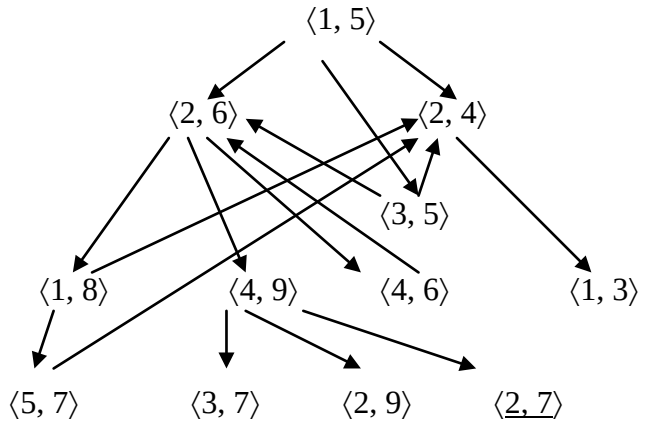


Рис. 23.1. Часть цепи, порождаемой парой ⟨1, 5⟩

Таким образом, матрица дополняется нулями в виде значений элементов, соответствующих парам ⟨1, 5⟩, ⟨2, 6⟩ и ⟨4, 9⟩:

$$\begin{array}{c} \begin{array}{cccccccc} 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \left[\begin{array}{cccccccc} 0 & 1 & 0 & 0 & 0 & & & 0 \\ & 0 & & 0 & 0 & 0 & 0 & \\ & & 0 & & 0 & & & 0 \\ & & & 0 & & 0 & 0 & 0 \\ & & & & 0 & & & 0 \\ & & & & & 0 & 0 & \\ & & & & & & & 0 \\ & & & & & & & 0 \end{array}\right] \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \end{array}.$$

Состояния, принадлежащие паре ⟨1, 7⟩, также оказались неэквивалентными. Цепь, порождаемая парой ⟨1, 8⟩ (рис. 23.2), не содержит пар

неэквивалентных состояний. Поэтому любая из пар, принадлежащих этому множеству, является парой эквивалентных состояний. Окончательно получим следующую матрицу эквивалентности:

$$\begin{array}{cccccccc} 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \left[\begin{array}{cccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 & 1 & 0 \\ & & & 0 & 0 & 0 & 0 & 0 \\ & & & & 0 & 1 & 0 & 0 \\ & & & & & 0 & 0 & 0 \\ & & & & & & 0 & 0 \\ & & & & & & & 0 \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \end{array}.$$

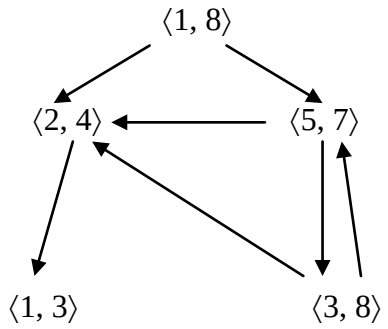


Рис. 23.2. Цепь, порождаемая парой ⟨1, 8⟩

Классы {1, 3, 8}, {2, 4}, {5, 7}, {6} и {9} определяются по строкам матрицы эквивалентности с учетом свойства транзитивности и симметричности отношения эквивалентности. Полученным классам эквивалентности ставим в соответствие состояния нового автомата 1, 2, 3, 4 и 5, в результате чего получим табл. 23.2, представляющую минимальный автомат, эквивалентный заданному.

Таблица 23.2

Таблица переходов и выходов минимального автомата

	a_1	a_2	a_3
1	2,1	2,0	3,0
2	1,0	2,1	2,1
3	4,1	2,0	1,0
4	1,0	5,1	4,1
5	3,0	5,1	3,1

Г л а в а 24

Минимизация частичных автоматов

24.1. Отношение реализации. Постановка задачи минимизации

В случае полного автомата задача минимизации сводится к установлению эквивалентности состояний и требуется найти автомат, поведение которого совпадает с поведением исходного автомата. В случае частичного автомата поведение искомого автомата с минимальным числом состояний должно совпадать в тех ситуациях, где оно определено для исходного автомата. Здесь понятие эквивалентности не подходит. Далее будет показано, что задача минимизации частичного автомата не сводится к задаче минимизации полного автомата. Для строгой формулировки задачи введем ряд понятий.

Пусть задан частичный автомат $M = (A, B, Q, \Psi, \Phi)$.

Входная последовательность $a_{i_1} a_{i_2} ... a_{i_p}$ называется *допустимой* для состояния q_{i_1} автомата M, если существует последовательность состояний $q_{i_1} q_{i_2} ... q_{i_p}$, такая, что значение $\Phi(a_{i_p}, q_{i_p})$ и значения $\Psi(a_{i_j}, q_{i_j})$ для $1 \le j \le p-1$ определены и $\Psi(a_{i_j}, q_{i_j}) = q_{i_{j+1}}$.

При допустимой для некоторого состояния входной последовательности автомат, выйдя из данного состояния, проходит вполне определенную последовательность состояний. Выход автомата должен быть определен для заключительного символа входной последовательности. Для промежуточных символов это не обязательно. Если при какой-то входной последовательности встретился неопределенный переход, то данная последовательность не является допустимой для соответствующего начального состояния.

Состояние q_j автомата M_2 *реализует* состояние q_i автомата M_1, если любая входная последовательность, допустимая для q_i, допустима и для q_j, а отвечающие ей выходные последовательности, полученные от автомата M_1 при начальном состоянии q_i и от автомата M_2 при начальном состоянии q_j, совпадают везде, где выходы автомата M_1 определены.

Отношение реализации, очевидно, является рефлексивным (любое состояние реализует само себя) и транзитивным (из того, что q_i реализует q_j, а q_j реализует q_k, следует, что q_i реализует q_k). Это отношение не является симметричным, т. е. из того, что q_i реализует q_j, не следует, что q_j реализует q_i. Симметричность не обеспечивается из-за того, что реализующее состояние может иметь более широкую область допустимых последовательностей, чем реализуемое состояние.

Автомат M_2 *реализует* автомат M_1, если для каждого состояния q_i автомата M_1 имеется по крайней мере одно состояние q_j автомата M_2, реализующее

состояние q_i. Поведение автомата M_2, реализующего автомат M_1, совпадает с поведением автомата M_1 везде, где поведение автомата M_1 определено.

Задача минимизации частичного автомата ставится следующим образом: для заданного автомата $M = (A, B, Q, \Psi, \Phi)$ найти реализующий его автомат $M' = (A, B, Q', \Psi', \Phi')$ с минимальным числом состояний.

Сформулированная задача сводится к группированию состояний исходного автомата в некоторые подмножества, в общем случае пересекающиеся, и сопоставлению каждого такого подмножества с состоянием нового автомата, реализующим любое состояние из этого подмножества. При этом необходимо, чтобы число таких подмножеств было минимальным, а объединение их составляло множество всех состояний исходного автомата.

Пусть имеется некоторая совокупность S подмножеств множества Q состояний автомата M. Эти подмножества могут пересекаться, но ни одно из них не содержится в другом. Совокупность S называется *группировкой* автомата M, если каждое его состояние входит хотя бы в одно из подмножеств данной совокупности.

Некоторое множество состояний $\{q_r, q_s, \dots, q_t\} \subseteq Q$ назовем *непосредственно производным по входному символу* $a \in A$ *множеством* от множества состояний $\{q_i, q_j, \dots, q_k\} \subseteq Q$, если значения $\Psi(a, q_j), \Psi(a, q_j), \dots, \Psi(a, q_k)$ составляют множество $\{q_r, q_s, \dots, q_t\}$.

Для множества состояний $\{q_r, q_s, \dots, q_t\}$ непосредственно производным от него по входному символу a является множество тех состояний, в которые автомат переходит из состояний $q_r, q_s, \dots, q_t$ при поступлении на его вход символа a.

Множество состояний $Q_i \subseteq Q$ называется *непосредственно производным* от $Q_j \subseteq Q$, если найдется такой входной символ $a \in A$, что Q_i является непосредственно производным по a от Q_j.

Группировка S называется *правильной*, если для каждого ее элемента S_i справедливо следующее:

– любое непосредственно производное от него множество является подмножеством какого-то из элементов S;

– для любых $q_j, q_k \in S_i$ и для любого $a \in A$ справедливо $\Phi(a, q_j) = \Phi(a, q_k)$ всегда, когда эти значения оба определены.

Для любого элемента S_i правильной группировки автомата M и любого входного символа $a \in A$ можно найти в этой же группировке элемент, содержащий все состояния, в которые переходит автомат M из состояний, принадлежащих S_i, при поступлении на его вход символа a.

Правильная группировка автомата M, имеющая минимальное число элементов среди всех правильных группировок автомата M, называется *минимальной*.

От любой правильной группировки автомата M можно перейти к автомату M', реализующему M, путем совмещения состояний, входящих в один и тот же элемент группировки. Если $\{q_i, q_j, \dots, q_k\}$ – элемент правильной группировки S

автомата M, то в автомате M' ему соответствует состояние, реализующее любое из состояний $q_i, q_j, \ldots, q_k$. Если S – минимальная правильная группировка, то построенный по ней автомат M' будет обладать минимальным числом состояний среди всех автоматов, реализующих автомат M.

Чтобы получить множество состояний Q' автомата M', надо каждому элементу $S_i \in S$ поставить в соответствие состояние $q'_i \in Q'$. Функции Φ' и Ψ' получаются следующим образом.

Пусть $q^{(i)}$ – некоторое (любое) состояние автомата M, принадлежащее элементу $S_i \in S$. Если $\Phi(a, q^{(i)}) = b$, то $\Phi'(a, q'_i) = b$. Если для всех $q^{(i)}$ из S_i значение $\Phi(a, q^{(i)})$ не определено, то значение $\Phi'(a, q'_i)$ считается неопределенным.

Если значение $\Psi(a, q^{(i)})$ не определено для всех $q^{(i)} \in S_i$, то $\Psi(a, q'_i)$ считается неопределенным. Обозначим символом $\Psi(a, S_i)$ множество, непосредственно производное от множества $S_i \in S$ по входному символу a (если значение $\Psi(a, q^{(i)})$ не определено для всех $q^{(i)} \in S_i$, то $\Psi(a, S_i) = \varnothing$). Тогда $\Psi(a, q'_i) = q'_j$, где q'_j соответствует любому $S_j \in S$, для которого $\Psi(a, S_i) \subseteq S_j$.

Рассмотрим заимствованный из работы [20] пример построения автомата по правильной группировке, на котором продемонстрируем, что минимизация частичного автомата не сводится к минимизации полного автомата. Пусть табл. 24.1 представляет таблицу переходов и выходов заданного частичного автомата. Все два варианта доопределения представлены в табл. 24.2 и табл. 24.3.

Таблица 24.1

Минимизируемый автомат

	a_1	a_2
1	1,–	2,0
2	3,0	1,0
3	2,1	1,0

Таблица 24.2

Вариант доопределения

	a_1	a_2
1	1,0	2,0
2	3,0	1,0
3	2,1	1,0

Нетрудно убедиться, что число состояний любого из этих полных автоматов не может быть уменьшено. В то же время множество $S = \{\{1, 2\}, \{1, 3\}\}$ является правильной группировкой заданного автомата и табл.24.4 представляет таблицу переходов и выходов построенного по данной группировке автомата, реализующего заданный автомат. Число состояний полученного автомата меньше числа состояний исходного автомата.

Таблица 24.3

Второй вариант доопределения

	a_1	a_2
1	1,1	2,0
2	3,0	1,0
3	2,1	1,0

Таблица 24.4

Результат минимизации

	a_1	a_2
1	2,0	1,0
2	1,1	1,0

24.2. Совместимость состояний

Состояния q_i и q_j автомата M *несовместимы*, если существует такая входная последовательность, допустимая для q_i и q_j, что заключительные выходные символы, вызываемые этой последовательностью при начальных состояниях q_i и q_j, не совпадают. Состояния q_i и q_j автомата M *совместимы*, если они не являются несовместимыми.

Отношение совместимости на множестве состояний автомата рефлексивно, симметрично, но не транзитивно. Из этих свойств отношение несовместимости обладает только симметричностью.

Очевидно, что любые два состояния, принадлежащие одному и тому же элементу правильной группировки, совместимы. Отношение совместимости можно использовать при нахождении минимальной правильной группировки.

В некоторых случаях совместимость или несовместимость состояний устанавливается непосредственно. Пусть q_i и q_j – состояния некоторого автомата M. Если существует столбец таблицы выходов, в котором элементы строк q_i и q_j определены и различны, то состояния q_i и q_j несовместимы. Это *явно несовместимые* состояния.

Если строки q_i и q_j таблицы переходов совпадают везде, где их элементы определены, и строки q_i и q_j таблицы выходов также совпадают везде, где их элементы определены, то состояния q_i и q_j совместимы. Это *явно совместимые* состояния.

Совместимость состояний q_i и q_j, которые не являются ни явно совместимыми, ни явно несовместимыми, определяется с помощью цепи, порождаемой парой состояний $\langle q_i, q_j \rangle$, которая находится так же, как цепь для полного автомата.

Цепью, порождаемой парой состояний $\langle q_i, q_j \rangle$ частичного автомата M, назовем множество C, элементами которого являются следующие пары состояний: сама пара $\langle q_i, q_j \rangle$; все пары вида $\langle \Psi(a, q_k), \Psi(a, q_l) \rangle$, где $\Psi(a, q_k)$ и $\Psi(a, q_l)$ определены и различны, если $\langle q_k, q_l \rangle \in C$. Другие пары не входят в C.

Справедливо следующее утверждение, которое доказывается точно так же, как утверждение 5.1.

У т в е р ж д е н и е 24.1. Состояния q_i и q_j автомата M являются совместимыми, если и только если в цепи, порождаемой парой состояний $\langle q_i, q_j \rangle$, нет ни одной пары явно несовместимых состояний. В этом случае все пары, принадлежащие данной цепи, являются парами совместимых состояний.

Совместимость удобно представлять булевой *матрицей совместимости*, строкам и столбцам которой соответствуют состояния автомата и элемент на пересечении i-й строки и j-го столбца имеет значение 1, если и только если состояния q_i и q_j совместимы. Процесс установления совместимости состояний

частичного автомата не отличается от процесса установления эквивалентности состояний полного автомата, описанного в разд. 5.2.

Пусть задан автомат, таблицу переходов и выходов которого представляет табл. 24.5.

Таблица 24.5
Таблица переходов и выходов минимизируемого автомата

	a_1	a_2	a_3
1	2,1	–,–	–,–
2	3,–	–,1	–,1
3	1,–	5,–	2,–
4	1,–	5,–	5,–
5	–,0	6,–	–,1
6	–,0	4,0	–,1

Явно несовместимыми являются пары ⟨1, 5⟩, ⟨1, 6⟩ и ⟨2, 6⟩. Пара ⟨2, 5⟩ является явно совместимой. Части цепей, порождаемых парами ⟨1, 2⟩, ⟨1, 4⟩, ⟨2, 4⟩, ⟨3, 5⟩ и ⟨3, 6⟩, которые необходимы для построения матрицы совместимости, показаны на рис. 24.1.

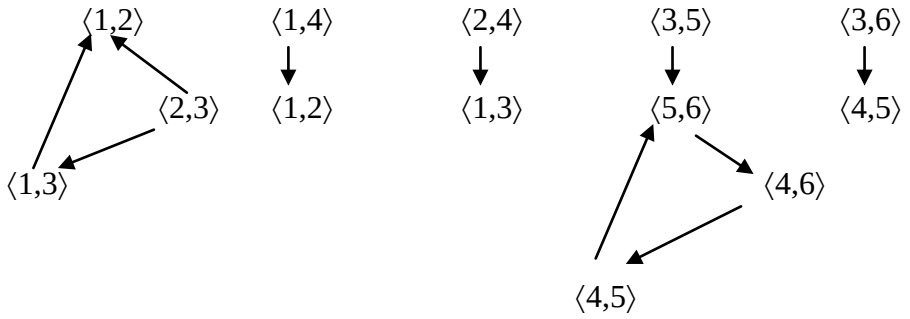


Рис. 24.1. Части цепей для установления совместимости состояний

Формирование матрицы совместимости показано на примере следующей последовательности матриц:

$$\begin{array}{c} 2\ 3\ 4\ 5\ 6 \\ \left[\begin{array}{ccccc} & & 0 & 0 & \\ & & 1 & 0 & \\ & & & & \\ & & & & \\ & & & & \end{array}\right] \begin{array}{c} 1 \\ 2 \\ 3' \\ 4 \\ 5 \end{array} \end{array} \quad \begin{array}{c} 2\ 3\ 4\ 5\ 6 \\ \left[\begin{array}{ccccc} 1 & 1 & 0 & 0 & \\ & 1 & 1 & 0 & \\ & & & & \\ & & & & \\ & & & & \end{array}\right] \begin{array}{c} 1 \\ 2 \\ 3' \\ 4 \\ 5 \end{array} \end{array} \quad \begin{array}{c} 2\ 3\ 4\ 5\ 6 \\ \left[\begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ & 1 & & 1 & 0 \\ & & & & \\ & & & & \\ & & & & \end{array}\right] \begin{array}{c} 1 \\ 2 \\ 3' \\ 4 \\ 5 \end{array} \end{array} \quad \begin{array}{c} 2\ 3\ 4\ 5\ 6 \\ \left[\begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ & 1 & 1 & 1 & 0 \\ & & & & \\ & & & & \\ & & & & \end{array}\right] \begin{array}{c} 1 \\ 2 \\ 3' \\ 4 \\ 5 \end{array} \end{array}$$

$$
\begin{array}{c|ccccc}
 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 1 & 1 & 1 & 0 & 0 \\
2 & & 1 & 1 & 1 & 0 \\
3' & & & 1 & & \\
4 & & & & & \\
5 & & & & &
\end{array}
\qquad
\begin{array}{c|ccccc}
 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 1 & 1 & 1 & 0 & 0 \\
2 & & 1 & 1 & 1 & 0 \\
3' & & & 1 & 1 & \\
4 & & & & 1 & 1 \\
5 & & & & & 1
\end{array}
\qquad
\begin{array}{c|ccccc}
 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 1 & 1 & 1 & 0 & 0 \\
2 & & 1 & 1 & 1 & 0 \\
3' & & & 1 & 1 & 1 \\
4 & & & & 1 & 1 \\
5 & & & & & 1
\end{array}
$$

Множество S_i называется *совместимым множеством*, если все состояния в нем попарно совместимы. Совместимое множество S_i называется максимальным совместимым множеством, если оно не содержится ни в каком другом совместимом множестве в качестве подмножества. К совместимым множествам относятся также все одноэлементные подмножества множества состояний.

Отношение совместимости на множестве состояний частичного автомата может быть представлено в виде неориентированного графа, вершинам которого соответствуют состояния и две вершины связаны ребром, если и только если соответствующие состояния совместимы. Этот граф назовем *графом совместимости* состояний. Дополнительный по отношению к нему граф является *графом несовместимости* состояний. Поиск всех максимальных совместимых множеств состояний любого автомата сводится к поиску всех максимальных независимых множеств в графе несовместимости, построенном для заданного автомата. Для решения этой задачи можно использовать метод, описанный в гл. 6, в результате применения которого получим множества {1, 2, 3, 4}, {2, 3, 4, 5} и {3, 4, 5, 6}.

Достижимая верхняя граница m числа всех максимальных совместимых множеств для автомата с числом состояний γ так же, как и наибольшее число всех максимальных независимых множеств в графе, приведенное в гл. 6, выражается следующими формулами: где k – некоторое целое положительное число:

$$m = 2 \cdot 3^{k-1}, \text{ если } \gamma = 3k - 1;$$

$$m = 3 \cdot 3^{k-1}, \text{ если } \gamma = 3k;$$

$$m = 4 \cdot 3^{k-1}, \text{ если } \gamma = 3k + 1.$$

24.3. Нахождение минимальной правильной группировки

Рассмотрим некоторые утверждения, позволяющие обосновать описываемый здесь метод.

У т в е р ж д е н и е 24.2. Непосредственно производное от совместимого множества есть совместимое множество.

Действительно, допустим обратное: пусть для некоторого автомата M подмножество S_i множества состояний является совместимым множеством, а непосредственно производное от него S_j – несовместимое множество. Тогда существует последовательность, допустимая для некоторых состояний $q_k, q_l \in S_i$, которая вызывает различные заключительные выходные символы у автомата M при начальных состояниях соответственно q_k и q_l, что противоречит определению совместимости состояний.

Из утверждения 24.2 вытекает следующее.

У т в е р ж д е н и е 24.3. Совокупность всех максимальных совместимых множеств есть правильная группировка.

Действительно, пусть $S = \{S_1, S_2, \ldots, S_m\}$ – совокупность всех максимальных совместимых множеств некоторого автомата M. Если бы S не была правильной группировкой, то существовало бы некоторое совместимое множество S_p, непосредственно производное от некоторого $S_i \in S$ и не являющееся подмножеством никакого $S_j \in S$, но тогда множество S содержало бы не все максимальные совместимые множества.

Из утверждения 24.3 следует, что правильную группировку можно формировать как некоторую совокупность совместимых множеств. Иногда совокупность всех максимальных совместимых множеств является минимальной правильной группировкой, но чаще всего это не так. Например, автомат, таблицей переходов и выходов которого является табл. 24.5, имеет три максимальных совместимых множества: {1, 2, 3, 4}, {2, 3, 4, 5} и {3, 4, 5, 6}, тогда как минимальной правильной группировкой для него является {{1, 2, 3}, {4, 5, 6}} и ни одна совокупность двух максимальных совместимых множеств не является правильной группировкой. Таблицей переходов и выходов минимального автомата, реализующего данный автомат, является табл. 24.6.

Таблица 24.6

Результат минимизации автомата, заданного табл. 24.5

	a_1	a_2	a_3
1	1,1	2,1	1,1
2	1,0	2,0	2,1

Минимальное число состояний γ' автомата, реализующего заданный автомат M с числом состояний γ, находится в следующих границах:

$$\alpha(G) \le \gamma' \le \min(\gamma, m), \qquad (24.1)$$

где $\alpha(G)$ – мощность наибольшего независимого множества графа G совместимости состояний; m – число всех максимальных совместимых множеств автомата M. Эта оценка полезна при поиске минимальной правильной группировки.

Для получения минимальной правильной группировки требуется осуществить комбинаторный поиск в пространстве максимальных и немаксимальных совместимых множеств. При этом некоторые из немаксимальных совместимых множеств можно исключить из рассмотрения, учитывая следующие соображения.

Для каждого совместимого множества S_i (максимального или немаксимального) построим множество $T_i = \{t_{i_1}, t_{i_2}, ..., t_{i_k}\}$, элементами которого являются подмножества множества состояний Q, обладающие следующими свойствами:

1) t_{i_j} ($j = 1, 2, \ldots, k$) является непосредственно производным множеством от S_i;

2) никакое t_{i_j} ($j = 1, 2, \ldots, k$) не содержится ни в S_i, ни в каком другом $t_{i_l} \in T_i$ в качестве подмножества;

3) $|t_{i_j}| > 1$ для всех $j = 1, 2, \ldots, k$.

Если $S_g \subset S_h$, а $T_g \supseteq T_h$, то S_g можно исключить из рассмотрения. Действительно, пусть получена правильная группировка, содержащая S_g. Заменив S_g на S_h, получим правильную группировку с тем же числом элементов.

Процесс нахождения минимальной правильной группировки состоит из двух этапов. На первом этапе находятся совместимые множества, максимальные и немаксимальные, подлежащие рассмотрению. На втором этапе находится совокупность совместимых множеств, обладающая свойствами правильной группировки, с помощью метода комбинаторного поиска, подобного тому, который используется при решении задачи о кратчайшем покрытии (см. гл. 11).

Выполняя первый этап, сначала находим все максимальные совместимые множества. При этом, как уже было сказано, можно использовать описанный в гл. 6 метод нахождения всех максимальных независимых множеств графа, имея в виду граф несовместимости состояний. Затем находим все собственные подмножества совместимых множеств. После получения очередного совместимого множества S_i сразу решаем вопрос об удалении или сохранении S_i, построив соответствующее T_i и сравнив его с другими такими множествами, построенными ранее.

Продемонстрируем выполнение первого этапа на примере автомата, таблицей переходов и выходов которого является табл. 24.7.

Максимальными совместимыми множествами для этого автомата являются $\{1, 2, 3, 5\}$, $\{2, 4\}$, $\{3, 6\}$ и $\{4, 6\}$. Все совместимые множества S_i с соответствующими T_i представлены в табл. 24.8. Из этой таблицы видно, что для сокращения перебора надо удалить все одноэлементные множества S_i, у которых по определению соответствующие T_i всегда пусты, а в данном случае для каждого одноэлементного множества S_i найдется двухэлементное

множество $S_j \supset S_i$, для которого $T_j = \varnothing$. Кроме того, удаляются множества $S_{11} = \{1,5\}$ и $S_{12} = \{2,3\}$, поскольку $S_{11} \subset S_6$, $T_{11} = T_6$ и $S_{12} \subset S_8$, $T_{12} = T_8$.

Таблица 24.7

Таблица переходов и выходов автомата, подлежащего минимизации

	a_1	a_2	a_3	a_4
1	–,–	3,1	4,1	2,1
2	4,0	–,–	–,–	–,–
3	6,0	6,1	–,–	–,–
4	–,–	6,0	1,0	5,1
5	–,–	–,–	2,1	–,–
6	3,0	–,–	2,0	3,1

Таблица 24.8

Совместимые множества

i	S_i	T_i
1	{1, 2, 3, 5}	{4, 6}, {3, 6}, {2, 4}
2	{2, 4}	∅
3	{3, 6}	∅
4	{4, 6}	{1, 2}, {3, 5}
5	{1, 2, 3}	{4, 6}, {3, 6}
6	{1, 2, 5}	{2, 4}
7	{1, 3, 5}	{3, 6}, {2, 4}
8	{2, 3, 5}	{4, 6}
9	{1, 2}	∅
10	{1, 3}	{3, 6}
11	{1, 5}	{2, 4}
12	{2, 3}	{4, 6}
13	{2, 5}	∅
14	{3, 5}	∅
15	{1}	∅
16	{2}	∅
17	{3}	∅
18	{4}	∅
19	{5}	∅
6	{6}	∅

Для выполнения второго этапа, осуществляющего комбинаторный поиск, строится *обобщенная таблица покрытия*. Пусть в результате выполнения первого этапа получена совокупность совместимых множеств $S_1, S_2, \ldots, S_m$ и соответствующие им множества $T_1, T_2, \ldots, T_m$. Строкам обобщенной таблицы покрытия соответствуют совместимые множества, а множество столбцов делится на две части: часть А и часть В. Столбцам части А соответствуют состояния заданного автомата, а столбцам части В – элементы множества

$T_1 \cup T_2 \cup \ldots \cup T_m = = \{t_1, t_2, \ldots, t_p\}$. Клетка таблицы пуста или содержит один из двух знаков: × или •. Часть А содержит только знаки ×, а часть В – и те, и другие. На пересечении строки S_i и столбца q_j в части А имеется знак ×, если и только если состояние q_j принадлежит множеству S_i. На пересечении строки S_i и столбца t_k имеется знак ×, если и только если $t_k \subseteq S_i$. На пересечении строки S_i и столбца t_l имеется знак •, если и только если t_l принадлежит T_i, соответствующему S_i.

Табл. 24.9 представляет обобщенную таблицу покрытий для автомата, таблицей переходов и выходов которого является табл. 24.7.

Таблица 24.9

Обобщенная таблица покрытия

Совместимые множества	A						B				
	1	2	3	4	5	6	{1, 2}	{2, 4}	{3, 5}	{3, 6}	{4, 6}
{1, 2, 3, 5}	×	×	×		×		×	•	×	•	•
{2, 4}		×		×				×			
{3, 6}			×			×				×	
{4, 6}				×		×	•		•		×
{1, 2, 3}	×	×	×				×			•	•
{1, 2, 5}	×	×			×		×	•			
{1, 3, 5}	×		×		×			•	×	•	
{2, 3, 5}		×	×		×				×		•
{1, 2}	×	×					×				
{1, 3}	×		×							•	
{2, 5}		×			×						
{3, 5}			×		×				×		

Задача заключается в том, чтобы найти минимальную совокупность строк, обладающую следующими двумя свойствами.

1. Каждый столбец части А имеет знак × хотя бы в одной из этих строк, т. е. искомая совокупность подмножеств должна покрывать все множество состояний.

2. Если какая-то строка из данной совокупности имеет знак • в части В, то столбец, содержащий этот знак, имеет знак × хотя бы в одной из строк данной совокупности, т. е. искомая совокупность должна обладать свойством правильной группировки, которое заключается в том, что любое непосредственно производное множество от любого элемента правильной группировки должно быть подмножеством некоторого элемента этой же группировки.

Процесс решения представляется как обход дерева поиска. Вершинам дерева соответствуют ситуации, связанные с выбором очередного столбца для покрытия. Ветви дерева соответствуют вариантам покрытия данного столбца. При рассмотрении очередной ситуации в процессе поиска решения применяются следующие правила редукции.

1. Если в части А имеется столбец с единственным знаком ×, то строка, содержащая в данном столбце этот знак, вносится в текущее решение и удаляется из таблицы вместе со всеми столбцами, где она имеет знак ×.

2. Если *i*-я строка имеет знак × везде, где такой знак имеет *j*-я строка, а *j*-я строка имеет знак • везде, где знак • имеет *i*-я строка, то *j*-я строка удаляется. Нетрудно видеть, что это действие представляет собой то же самое, что и описанное выше удаление некоторых совместимых множеств перед построением таблицы.

3. Если *i*-й столбец имеет знак × везде, где имеет такой знак *j*-й столбец из части А, то *i*-й столбец удаляется.

4. Если из какого-то столбца части В при включении строки в решение исчез хотя бы один знак •, то этот столбец переводится в часть А и из него удаляются все знаки •.

5. Если в результате удаления строк в некотором столбце части В остались только знаки •, то строки, содержащие эти знаки, удаляются.

Если ни одно из перечисленных условий не выполняется, то выбирается столбец в части А с минимальным числом знаков × и при формировании текущего решения в первую очередь выбирается строка с максимальным числом знаков ×. Если таких строк несколько, выбирается та, которая имеет наименьшее число знаков •. Выбранная строка удаляется из таблицы и удаляются столбцы, имеющие знак × в этой строке.

Пусть исходной таблицей является табл. 24.9. Выбираем столбец 4 как содержащий минимум знаков ×. Строки {2, 4} и {4, 6} имеют одинаковое число знаков ×, но строка {2, 4} не содержит знаков •, и поэтому в текущее решение включаем {2, 4}. После соответствующего преобразования согласно правилам редукции получаем табл. 24.10. В этой таблице выбираем столбец 6 и покрывающую его строку {3, 6}. Табл. 24.10 преобразуется в табл. 24.11.

Таблица 24.10
Результат первого шага преобразования табл. 24.9

Совместимые множества	A			B			
	1	3	6	{1, 2}	{3, 5}	{3, 6}	{4, 6}
{1, 2, 3, 5}	×	×		×	×	•	•
{3, 6}		×	×			×	
{4, 6}			×	•	•		×
{1, 2, 5}	×			×			
{1, 3, 5}	×	×			×	•	
{3, 5}		×			×		

В части А табл. 24.11 остался один столбец, который покрывается тремя строками. Видно, что для его покрытия не стоит выбирать строку {1, 2, 3, 5}, так как согласно правилу 4 в части А появляется новый столбец, соответствующий множеству {4, 6}, который тоже должен быть покрыт.

Поэтому выбираем строку $\{1, 2, 5\}$. Полученная правильная группировка $\{1, 2, 5\}$, $\{2, 4\}$, $\{3, 6\}$ является минимальной. Действительно, обратившись к дереву поиска, изображенному на рис. 24.2, видим, что вернувшись в вершину 6 и заменив множество $\{3, 6\}$ на множество $\{4, 6\}$, не получим группировки с двумя элементами. Вернувшись же в начальную вершину 4, можно получить единственную двухэлементную группировку $\{1, 2, 3, 5\}$, $\{4, 6\}$, которая не является правильной.

Таблица 24.11

Результат преобразования табл. 24.10

Совместимые множества	A	B		
	1	{1, 2}	{3, 5}	{4, 6}
{1, 2, 3, 5}	×	×	×	•
{4, 6}		•	•	×
{1, 2, 5}	×	×		
{1, 3, 5}	×		×	

В результате минимизации автомата, представленного табл. 24.7, получаем автомат с тремя состояниями, таблицей переходов и выходов которого является табл. 24.12.

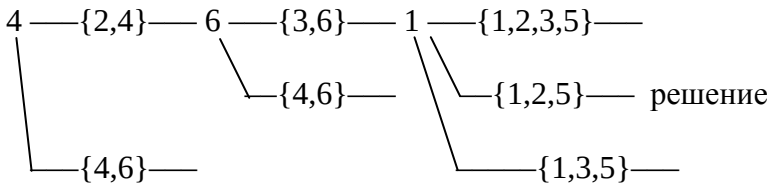


Рис. 24.2. Дерево поиска минимальной правильной группировки

Таблица 24.12

Таблица переходов и выходов минимального автомата

	a_1	a_2	a_3	a_4
1	2,0	3,1	2,1	1,1
2	2,0	3,0	1,0	1,1
3	3,0	3,1	1,0	3,1

Г л а в а 25

Кодирование состояний синхронного автомата

25.1. Задача кодирования состояний

В абстрактной модели автомата $M = (A, B, Q, \Psi, \Phi)$ элементами множеств A, B и Q являются абстрактные символы. Как было сказано выше, для реализации заданного поведения в виде логической сети надо перейти от функций Ф и Ψ к системе булевых функций, т. е. от абстрактной модели автомата надо перейти к структурной модели. При этом переменные a, b и q заменяются векторными переменными:

$$a \rightarrow \boldsymbol{x} = (x_1, x_2, \ldots, x_n);$$
$$b \rightarrow \boldsymbol{y} = (y_1, y_2, \ldots, y_m);$$
$$q \rightarrow \boldsymbol{z} = (z_1, z_2, \ldots, z_k).$$

Различным значениям многозначных переменных a, b и q должны быть поставлены в соответствие различные значения векторных переменных $\boldsymbol{x}$, $\boldsymbol{y}$ и $\boldsymbol{z}$. Векторы $\boldsymbol{x}$, $\boldsymbol{y}$ и $\boldsymbol{z}$ показывают структуру абстрактных символов a и b и состояния q. Элементами этой структуры являются соответственно двоичные сигналы и состояния двоичных элементов памяти. Функции Ф и Ψ преобразуются соответственно в векторные функции $\boldsymbol{\varphi}(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{y}$ и $\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{z}^+$, а те, в свою очередь, – в систему булевых функций, число которых $m + k$:

$$y_i = \varphi_i(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_k),\ i = 1, 2, \ldots, m;$$
$$z_j^+ = \psi_i(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_k),\ j = 1, 2, \ldots, k.$$

Числа n, m и k должны удовлетворять соотношениям $\alpha \le 2^n$, $\beta \le 2^m$ и $\gamma \le 2^k$, где α, β и γ – числа абстрактных входных и выходных символов и состояний соответственно. Минимальные значения этих величин определяются как $n = \lceil \log_2 \alpha \rceil$, $m = \lceil \log_2 \beta \rceil$ и $k = \lceil \log_2 \gamma \rceil$, где $\lceil a \rceil$ означает минимальное целое число, не меньшее a.

Векторы, приписываемые различным абстрактным символам и состояниям в случае синхронного автомата, могут быть выбраны произвольно с одним лишь ограничением на число компонент, указанным выше, но если принимать во внимание простоту комбинационной схемы, реализующей заданное поведение, то различные варианты кодирования неравнозначны. Часто на практике используется модель автомата с абстрактным состоянием, где векторы $\boldsymbol{x}$ и $\boldsymbol{y}$ заданы и остается только закодировать многозначную переменную q. В дальнейшем будем считать, что структура входных и выходных сигналов

задана, т. е. векторы $\boldsymbol{x}$ и $\boldsymbol{y}$ заданы, и будем рассматривать только кодирование состояний, т. е. замену значений многозначной переменной q значениями вектора $\boldsymbol{z}$.

Чтобы показать неравнозначность выбора вариантов кодирования, приведем следующий простой пример из работы [20]. Пусть табл. 25.1 представляет собой таблицу переходов и выходов заданного автомата, а табл. 25.2 – два варианта кодирования его состояний.

Таблица 25.1

Таблица переходов и выходов

	0	1
q_1	q_1, 0	q_2, 0
q_2	q_1, 0	q_3, 1
q_3	q_4, 1	q_1, 0
q_4	q_1, 1	q_1, 1

Таблица 25.2

Варианты кодирования состояний

	Вариант 1	Вариант 2
q_1	1 1	0 0
q_2	0 0	1 0
q_3	1 0	1 1
q_4	0 1	0 1

Соответствующие системы булевых функций представлены матрицами $\boldsymbol{U}_1$, $\boldsymbol{V}_1$ для варианта 1 и $\boldsymbol{U}_2$, $\boldsymbol{V}_2$ для варианта 2:

$$\boldsymbol{U}_1 = \begin{matrix} x & z_1 & z_2 \end{matrix}\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{V}_1 = \begin{matrix} z_1^+ & z_2^+ & y \end{matrix}\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix};$$

$$\boldsymbol{U}_2 = \begin{matrix} x & z_1 & z_2 \end{matrix}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{V}_2 = \begin{matrix} z_1^+ & z_2^+ & y \end{matrix}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

После минимизации данных систем булевых функций получим следующие системы ДНФ в матричном представлении:

$$\boldsymbol{U}_1 = \begin{matrix} x & z_1 & z_2 \end{matrix}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & - \\ 1 & 0 & - \\ 0 & - & 1 \\ - & 0 & 1 \end{bmatrix}, \quad \boldsymbol{V}_1 = \begin{matrix} z_1^+ & z_2^+ & y \end{matrix}\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix};$$

$$\boldsymbol{U}_2 = \begin{matrix} x & z_1 & z_2 \end{matrix}\begin{bmatrix} 1 & - & 0 \\ - & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad \boldsymbol{V}_2 = \begin{matrix} z_1^+ & z_2^+ & y \end{matrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Отсюда ясно видно, что, выбрав вариант 2 для кодов состояний заданного автомата, получим более простую систему ДНФ.

Можно подсчитать, сколько всего различных и неравнозначных вариантов кодирования существует для конкретного автомата. Пусть γ – число состояний автомата и k – минимальная длина булева вектора, кодирующего состояние. Тогда число различных вариантов кодирования равно числу размещений 2^k элементов по γ позициям, т. е. $\frac{2^k!}{(2^k-\gamma)!}$. Если учесть то, что перестановка столбцов дает равнозначные варианты, то получим $\frac{2^k!}{(2^k-\gamma)!k!}$. Можно считать равнозначными варианты, получаемые друг из друга путем инвертирования значений внутренних переменных. Тогда получим $\frac{2^k!}{(2^k-\gamma)!k!2^k}$ или $\frac{(2^k-1)!}{(2^k-\gamma)!k!}$. Ясно, что перебрать все варианты реально только при небольшом числе состояний автомата. Если же число состояний достаточно велико, то для выбора не самого плохого варианта кодирования необходимо применять более изощренный способ, чем полный перебор.

25.2. Метод «желательных соседств»

Рассмотрим задачу упрощения только функций состояний памяти при кодировании состояний автомата, хотя при некотором усложнении используемого критерия можно добиваться упрощения и выходных функций.

Каждой паре состояний q_i, q_j автомата с множеством состояний $Q = \{q_1, q_2, \dots, q_\gamma\}$ припишем вес $w_{ij} = w'_{ij} + w''_{ij}$, где w''_{ij} – число столбцов таблицы переходов, в которых строки q_i и q_j имеют одинаковые элементы, т. е. число значений переменной a, при которых $\Psi(a, q_i) = \Psi(a, q_j)$, а w'_{ij} определяется следующим образом. Пусть w_p – число пар вида $\langle \Psi(a_s, q_p), \Psi(a_t, q_p) \rangle$, где входные символы a_s и a_t имеют соседние коды, $\Psi(a_s, q_p) = q_i$ и $\Psi(a_t, q_p) = q_j$. Тогда $w'_{ij} = \sum_{p=1}^{\gamma} w_p$.

Желательно, чтобы коды состояний q_i и q_j были тем ближе друг к другу, чем больше величина w_{ij}. Здесь имеется в виду расстояние в гиперкубе, представляющем пространство кодирующих переменных, между вершинами, соответствующими данным кодам. Для объяснения такого правила приведем следующие соображения.

Общий вид матриц $\boldsymbol{U}$ и $\boldsymbol{V}$, задающих систему булевых функций z_j^+ ($j = 1, 2, \dots, k$), представим следующим образом:

$$\boldsymbol{U}=\begin{array}{c}\begin{matrix} x_1 & x_2 & \dots & x_n & z_1 & z_2 & \dots & z_k \end{matrix}\\ \begin{bmatrix} & \dots & & & \dots & & \\ x_{s1} & x_{s2} & \dots & x_{sn} & z_{p1} & z_{p2} & \dots & z_{pk} \\ & \dots & & & \dots & & \\ x_{t1} & x_{t2} & \dots & x_{tn} & z_{p1} & z_{p2} & \dots & z_{pk} \\ & \dots & & & \dots & & \end{bmatrix}\end{array}, \quad \boldsymbol{V}=\begin{array}{c}\begin{matrix} z_1^+ & z_2^+ & \dots & z_k^+ \end{matrix}\\ \begin{bmatrix} & \dots & \\ z_{i1} & z_{i2} & \dots & z_{ik} \\ & \dots & \\ z_{j1} & z_{j2} & \dots & z_{jk} \\ & \dots & \end{bmatrix}\end{array},$$

где $(x_{s1}, x_{s2}, \dots, x_{sn})$ и $(x_{t1}, x_{t2}, \dots, x_{tn})$ – соседние векторы, являющиеся кодами входных символов a_s и a_t и представляющие совокупности входных двоичных сигналов, а $(z_{p1}, z_{p2}, \dots, z_{pk})$, $(z_{i1}, z_{i2}, \dots, z_{ik})$ и $(z_{j1}, z_{j2}, \dots, z_{jk})$ – коды состояний q_p, q_i и q_j соответственно.

Здесь отражена ситуация, которая учитывается при подсчете значения w'_{ij}. Пара вектор-строк $(x_{s1}, x_{s2}, \dots, x_{sn}, z_{p1}, z_{p2}, \dots, z_{pk})$ и $(z_{i1}, z_{i2}, \dots, z_{ik})$ матриц $\boldsymbol{U}$ и $\boldsymbol{V}$ представляет переход автомата из состояния q_p в состояние q_i при входном символе a_s, выражаемый формулой $\Psi(a_s, q_p) = q_i$. Точно так же пара строк $(x_{t1}, x_{t2}, \dots, x_{tn}, z_{p1}, z_{p2}, \dots, z_{pk})$, $(z_{j1}, z_{j2}, \dots, z_{jk})$ представляет переход, выражаемый формулой $\Psi(a_t, q_p) = q_j$. Отсюда видно, что чем больше одноименных компонент векторов $(z_{i1}, z_{i2}, \dots, z_{ik})$ и $(z_{j1}, z_{j2}, \dots, z_{jk})$, являющихся кодами состояний q_i и q_j, совпадают и равны единице, тем больше, возможно, будет условий для простого склеивания элементарных конъюнкций в получаемой системе ДНФ.

Ситуация, учитываемая при подсчете значения w''_{ij}, представляется следующими матрицами:

$$\boldsymbol{U}=\begin{array}{c}\begin{matrix} x_1 & x_2 & \dots & x_n & z_1 & z_2 & \dots & z_k \end{matrix}\\ \begin{bmatrix} & \dots & & & \dots & & \\ x_{u1} & x_{u2} & \dots & x_{un} & z_{i1} & z_{i2} & \dots & z_{ik} \\ & \dots & & & \dots & & \\ x_{u1} & x_{u2} & \dots & x_{un} & z_{j1} & z_{j2} & \dots & z_{jk} \\ & \dots & & & \dots & & \end{bmatrix}\end{array}, \quad \boldsymbol{V}=\begin{array}{c}\begin{matrix} z_1^+ & z_2^+ & \dots & z_k^+ \end{matrix}\\ \begin{bmatrix} & \dots & \\ z_{v1} & z_{v2} & \dots & z_{vk} \\ & \dots & \\ z_{v1} & z_{v2} & \dots & z_{vk} \\ & \dots & \end{bmatrix}\end{array}.$$

Здесь пары строк $(x_{u1}, x_{u2}, \dots, x_{un}, z_{i1}, z_{i2}, \dots, z_{ik})$, $(z_{v1}, z_{v2}, \dots, z_{vk})$ и $(x_{u1}, x_{u2}, \dots, x_{un}, z_{j1}, z_{j2}, \dots, z_{jk})$, $(z_{v1}, z_{v2}, \dots, z_{vk})$ представляют переходы в одно и то же состояние q_v при одном и том же значении a_u переменной a, т. е. $\Psi(a_u, q_i) = \Psi(a_u, q_j) = q_v$. Ясно, что желательно иметь соседними коды тех состояний q_i и q_j, для которых переменная a принимает много значений a_u, удовлетворяющих $\Psi(a_u, q_i) = \Psi(a_u, q_j)$. Тогда возможно простое склеивание элементарных конъюнкций, представленных показанными строками матрицы $\boldsymbol{U}$.

Одна из реализаций метода «желательных соседств» представляется как процесс построение k-мерного гиперкуба, напоминающий сборку некоторой простой механической конструкции. При этом вершины гиперкуба, являющиеся первоначально вершинами некоторого пустого графа (без ребер), заранее поставлены в соответствие состояниям автомата и на парах этих вершин заданы величины w_{ij}.

Исходными данными для построения k-мерного гиперкуба являются величины w_{ij} и число состояний автомата γ. Предполагается, что это число минимально или по каким-то причинам не подлежит минимизации. Если γ не представляет собой целой степени двойки, то оно увеличивается до $\gamma' = 2^k$ и считается, что $w_{rs} = 0$, если хотя бы одно из q_r и q_s является дополнительно введенным таким образом состоянием.

Построение k-мерного гиперкуба представляется как последовательность k шагов. На p-м шаге рассматривается множество $(p-1)$-мерных гиперкубов, они объединяются в пары, и из каждой пары получается один p-мерный гиперкуб путем соответствующего добавления ребер. При этом по возможности для соединения ребрами выбираются те вершины, которым соответствуют наибольшие из оставшихся значения w_{ij}. Вершинам полученного гиперкуба приписываем k-компонентные булевы векторы с соблюдением отношения соседства, представленного ребрами гиперкуба.

На первом шаге из γ' изолированных вершин, или 0-мерных гиперкубов, строятся одномерные гиперкубы в виде $\gamma'/2$ попарно несмежных ребер. На последнем k-м шаге из двух $(k-1)$-мерных гиперкубов собирается один k-мерный гиперкуб путем добавления 2^{k-1} ребер.

Продемонстрируем этот процесс на примере автомата, таблицей переходов которого является табл. 25.3. Введем два фиктивных состояния q_7 и q_8, чтобы довести число состояний до $8 = 2^3$. Значения w_{ij} удобно задать в виде табл. 25.4, где строки и столбцы соответствуют состояниям автомата.

На первом шаге получаем четыре одномерных гиперкуба, изображенных на рис. 25.1. Максимальное значение имеет вес $w_{23} = 3$. Поэтому в первую очередь ребром соединяются вершины q_2 и q_3. Затем ребрами соединяются вершины q_1 с q_6, q_4 с q_5 и q_7 с q_8.

Чтобы из четырех одномерных гиперкубов получить два двухмерных, надо добавить четыре ребра. Для этого выбираются такие ребра, чтобы сумма их весов была максимальна. Сначала строится один гиперкуб, для которого выбираются два ребра с максимальной суммой весов. Затем точно так же собирается второй гиперкуб. На рис. 25.2 показаны варианты выбора ребер для получения первого двухмерного гиперкуба.

Вершины q_7 и q_8 здесь не участвуют, так как все инцидентные им ребра имеют нулевой вес и сумма их весов заведомо не максимальна. Максимальной суммой весов обладает четвертый вариант. Для второго гиперкуба все варианты одинаковы.

На рис. 25.3 изображены два двумерных гиперкуба, из которых надо собрать один трехмерный гиперкуб, добавив четыре ребра. Варианты такой сборки представлены на рис. 25.4. Сумма весов ребер показана ниже каждого изображения варианта сборки.

Таблица 25.3

Таблица переходов

q \ $x_1 x_2$	00	01	10
q_1	q_1	q_2	q_6
q_2	q_3	q_2	q_1
q_3	q_2	q_3	q_5
q_4	q_4	q_5	q_2
q_5	q_3	q_5	q_4
q_6	q_3	q_2	q_4

Таблица 25.4

Значения w_{ij}

q_2	q_3	q_4	q_5	q_6	q_7	q_8	
2	1	0	0	2	0	0	q_1
	3	1	2	2	0	0	q_2
		2	1	0	0	0	q_3
			2	0	0	0	q_4
				2	0	0	q_5
					0	0	q_6
						0	q_7

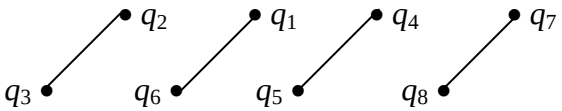


Рис. 25.1. Результат первого шага построения гиперкуба

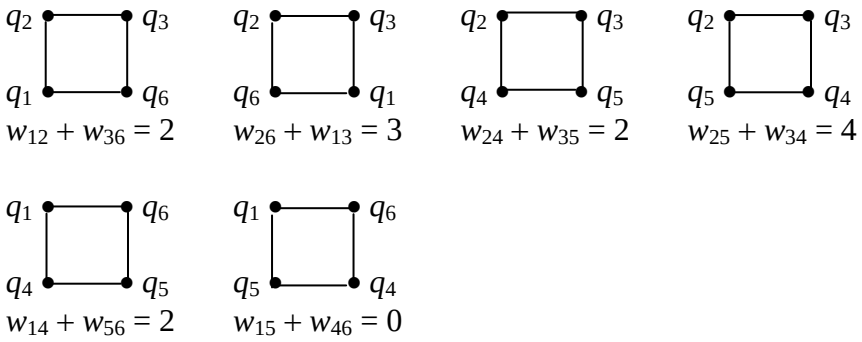


Рис. 25.2. Варианты сборки двухмерного гиперкуба

Для окончательного решения нашего примера выбран вариант сборки с максимальной суммой весов добавляемых ребер, равной 4. Этот гиперкуб показан отдельно на рис. 25.5, где возле каждой вершины представлен код соответствующего состояния.

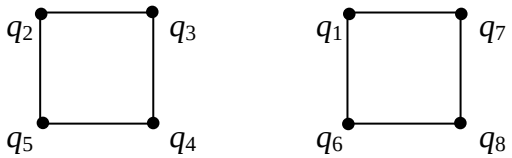


Рис. 25.3. Результат сборки двумерных гиперкубов

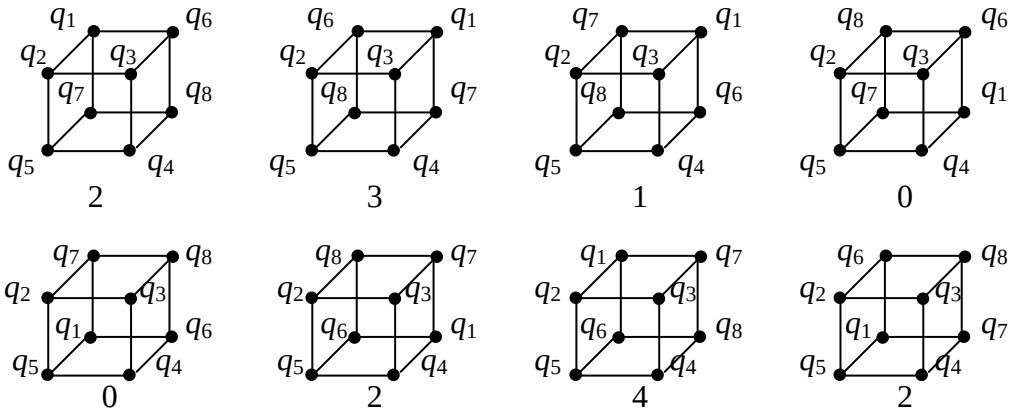


Рис. 25.4. Варианты сборки трехмерного гиперкуба

Булев автомат, соответствующий данному варианту кодирования, представим тремя картами Карно, которые задают не полностью определенные функции z_1^+, z_2^+ и z_3^+ и строкам которых соответствуют состояния заданного автомата (рис. 25.6).

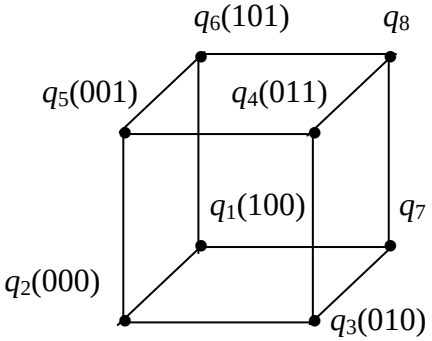


Рис. 25.5. Результат сборки гиперкуба с кодами состояний

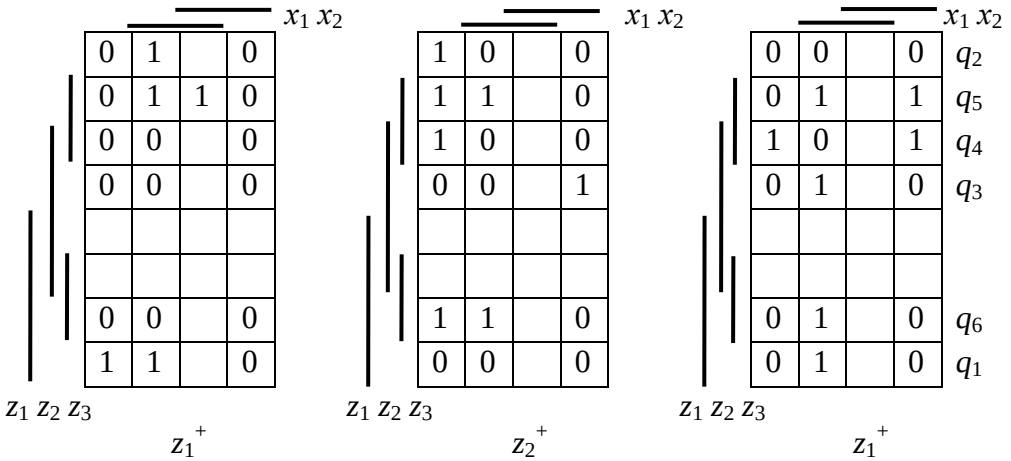


Рис. 25.6. Представление функций z_1^+, z_2^+ и z_3^+ с помощью карт Карно

Минимизированная система булевых функций, описывающая заданное поведение, представляется следующими матрицами:

$$\boldsymbol{U} = \begin{array}{c} \begin{matrix} x_1 & x_2 & z_1 & z_2 & z_3 \end{matrix} \\ \begin{bmatrix} 1 & - & - & 0 & 0 \\ - & 0 & 1 & - & 0 \\ 0 & 0 & 0 & 0 & - \\ 0 & 0 & - & - & 1 \\ 1 & - & - & 0 & 1 \\ 1 & - & - & 1 & 0 \\ - & 1 & - & 1 & 0 \\ 0 & - & - & 1 & 1 \\ - & 1 & 0 & - & 1 \\ 1 & - & 1 & - & - \end{bmatrix} \end{array}, \quad \boldsymbol{V} = \begin{array}{c} \begin{matrix} z_1^+ & z_2^+ & z_3^+ \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array}.$$

Для сравнения приведем минимизированную систему булевых функций, получаемую при произвольном кодировании, например, путем приписывания состояниям чисел от 0 до 5 в двоичной системе счисления согласно порядку номеров состояний. Матрица кодирования и матрицы, представляющие данную систему булевых функций, имеют следующий вид:

$$\boldsymbol{C} = \begin{array}{c} \begin{matrix} z_1 & z_2 & z_3 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \end{array}, \quad \boldsymbol{U} = \begin{array}{c} \begin{matrix} x_1 & x_2 & z_1 & z_2 & z_3 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & - & 1 \\ 0 & 0 & - & 1 & - \\ - & 1 & - & 0 & 1 \\ - & 1 & - & 1 & 1 \\ - & 1 & 1 & - & 0 \\ 1 & - & 0 & - & 0 \\ - & 0 & 1 & - & - \\ - & 1 & - & 1 & 0 \\ 1 & - & - & 0 & 0 \\ 1 & - & - & 1 & 1 \\ 1 & - & 1 & - & - \\ - & 1 & 0 & - & - \end{bmatrix} \end{array}, \quad \boldsymbol{V} = \begin{array}{c} \begin{matrix} z_1^+ & z_2^+ & z_3^+ \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array}.$$

Последняя система ДНФ оказалась сложнее – число различных элементарных конъюнкций в ней на две больше, чем в предыдущей системе.

Г л а в а 26

Кодирование состояний асинхронного автомата

26.1. Явление состязаний элементов памяти

Асинхронный автомат отличается от синхронного тем, что промежуток между моментами времени, когда автомат меняет свое состояние, у него не фиксирован, а определяется изменением входного сигнала. В связи с этим на функцию переходов Ψ накладывается следующее ограничение: если $\Psi(a, q_i) = q_j$ для некоторых $a \in A$ и $q_i \in Q$, то $\Psi(a, q_j) = q_j$, т. е. всякий переход должен всегда вести в некоторое состояние, устойчивое при действующем в данный момент входном сигнале a. Естественно, что входной сигнал не должен меняться до тех пор, пока автомат не придет в устойчивое состояние. Без этих ограничений невозможно построить автомат, реализующий заданное поведение.

В реальных схемах не может быть одновременного изменения различных сигналов, допускаемого в абстрактной модели синхронного автомата. Реальные логические элементы обладают инерционностью, приводящей к некоторым задержкам их «срабатывания», причем разброс этих задержек для различных элементов носит случайный характер. Кроме того, сигналы, переключающие различные элементы памяти, могут проходить цепочки логических элементов, имеющие различную длину. Задержки на элементах в цепочках суммируются. Сами элементы памяти имеют различное время переключения. В схемах, работающих в синхронном режиме, такими задержками можно пренебречь, поскольку там процесс переключения элементов памяти и процесс формирования переключающих сигналов разделены во времени. В асинхронных схемах эти процессы происходят одновременно и указанное свойство реальных элементов должно учитываться.

Рассмотрим процесс перехода из состояния в состояние для асинхронного автомата, поведение которого представлено в табл. 26.1 с выделенными устойчивыми состояниями, где крайний правый столбец представляет произвольно выполненное кодирование состояний. Переход из одного состояния в другое в реальной схеме реализуется как смена набора состояний элементов памяти. Пусть сначала действует входной сигнал a_1 и автомат находится в устойчивом состоянии q_1 (код 000). Затем входной сигнал меняется на a_3, и автомат согласно заданному поведению должен пойти в состояние q_4 (код 011). В зависимости от того, какой из двух элементов памяти, z_2 или z_3, меняет свое состояние первым, автомат может оказаться на какое-то время в промежуточном состоянии, представленном набором состояний элементов памяти 010 либо 001. Если первым меняет свое состояние элемент z_3, то автомат окажется в состоянии q_2 (код 001), которое является устойчивым для

входного сигнала a_3, т. е. вместо того, чтобы идти в состояние q_4, автомат остается в состоянии q_2.

Таблица 26.1

Таблица переходов асинхронного автомата

	a_1	a_2	a_3	$z_1\,z_2\,z_3$
q_1	$\boldsymbol{q}_1$	q_3	q_4	0 0 0
q_2	q_1	q_3	$\boldsymbol{q}_2$	0 0 1
q_3	q_1	$\boldsymbol{q}_3$	–	0 1 0
q_4	q_5	–	$\boldsymbol{q}_4$	0 1 1
q_5	$\boldsymbol{q}_5$	q_3	q_2	1 0 0

Рассмотренное явление носит название *состязаний* или *гонок* элементов памяти. Принято называть состязания *неопасными*, если все промежуточные состояния, в которых автомат может оказаться при переходе из одного состояния в другое под воздействием некоторого входного сигнала a, являются неустойчивыми для сигнала a, т. е. при любом порядке переключений элементов памяти автомат из некоторого состояния q_i под воздействием входного сигнала a переходит всегда в состояние $q_j = \Psi(a, q_i)$. Если же при этом автомат может оказаться в некотором устойчивом состоянии q_k, отличном от q_j, то состязания называются *опасными*.

Состязания могут быть также и между входными двоичными сигналами. Чтобы устранить их влияние на работу автомата, вводят обычно ограничение на изменение входных сигналов. Например, можно наложить запрет на изменение одновременно более чем одного двоичного сигнала.

26.2. Условие отсутствия опасных состязаний

Реализацию заданного в автомате перехода из состояния q_i в состояние q_j (примем для такого перехода обозначение $q_i \to q_j$) можно обеспечить, придав функции переходов Ψ значение q_j на всех возможных промежуточных состояниях, в которые автомат может попасть из состояния q_i при заданном фиксированном входном сигнале a, т. е. $\Psi(a, q_l) = q_j$ при любом q_l, в котором может оказаться автомат при переходе из q_i. В этом случае элементы памяти, которые должны изменить свое состояние, будут подвергаться надлежащему постоянному воздействию на всем протяжении рассматриваемого перехода, независимо от того, в каком порядке они «сработают».

Обозначим $U(q_i, q_j)$ множество всех возможных промежуточных состояний (включая исходное и конечное), в которые автомат может попасть при реализации перехода $q_i \to q_j$, если элементы памяти, меняющие свое состояние, делают это в произвольном порядке. Для приведенного выше примера $U(q_1, q_4) = \{q_1, q_2, q_3, q_4\}$, $U(q_5, q_2) = \{q_1, q_2, q_5, q\}$, где q – некоторое состояние,

не предусмотренное в задании поведения автомата, соответствующее набору состояний элементов памяти 101.

Два перехода $q_i \to q_j$ и $q_k \to q_l$, совершаемые автоматом при одном и том же входном сигнале в различные состояния ($q_j \neq q_l$), назовем *парой переходов*. Тогда условие отсутствия опасных состязаний формулируется следующим образом.

У т в е р ж д е н и е 26.1. При одновременном возбуждении элементов памяти в процессе перехода опасные состязания отсутствуют тогда и только тогда, когда для каждой пары переходов $q_i \to q_j$, $q_k \to q_l$ имеет место $U(q_i, q_j) \cap U(q_k, q_l) = \varnothing$.

Множества $U(q_i, q_j)$ и $U(q_k, q_l)$ можно формально представить как множества кодов соответствующих состояний, а те, в свою очередь, представляются троичными векторами, полученными из кодов состояний q_i, q_j, q_k и q_l. Обозначим эти векторы $t(q_i, q_j)$ и $t(q_k, q_l)$. Компоненты вектора $t(q_i, q_j)$ принимают значения одноименных компонент кодов q_i и q_j, если они совпадают, и принимают значение «–» в противном случае. Например, если коды состояний q_i и q_j соответственно (0 0 0 1) и (0 1 0 1), то $t(q_i, q_j) = (0 – 0\ 1)$.

Необходимым и достаточным условием непересечения множеств $U(q_i, q_j)$ и $U(q_k, q_l)$ является ортогональность векторов $t(q_i, q_j)$ и $t(q_k, q_l)$. К этому условию надо добавить еще необходимое требование того, что для реализации различных состояний их коды должны быть ортогональными.

Для автомата, таблицей переходов которого является табл. 26.1, имеются следующие пары переходов:

для a_1	для a_3
$q_1 \to q_1$, $q_4 \to q_5$;	$q_1 \to q_4$, $q_2 \to q_2$;
$q_1 \to q_1$, $q_5 \to q_5$;	$q_1 \to q_4$, $q_5 \to q_2$;
$q_2 \to q_1$, $q_4 \to q_5$;	$q_2 \to q_2$, $q_4 \to q_4$;
$q_2 \to q_1$, $q_5 \to q_5$;	$q_4 \to q_4$, $q_5 \to q_2$.
$q_3 \to q_1$, $q_4 \to q_5$;	
$q_3 \to q_1$, $q_5 \to q_5$;	

Для входного сигнала a_2 пары переходов согласно определению этого понятия отсутствуют. Легко проверить, что кодирование состояний, представленное следующей матрицей, обеспечивает отсутствие опасных состязаний:

$$\boldsymbol{C} = \begin{array}{c} \begin{matrix} z_1 & z_2 & z_3 \end{matrix} \\ \begin{bmatrix} 0 & 0 & - \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & - \\ 1 & 1 & - \end{bmatrix} \end{array} \begin{matrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{matrix}.$$

Функции возбуждения элементов памяти, описывающие сигналы, которые устанавливают элементы памяти в нужные состояния, при фиксированных значениях двоичных входных сигналов определяются на интервалах, которым соответствуют множества $U(q_i, q_j)$. Для любого $U(q_i, q_j)$ такой интервал представляется троичным вектором $t(q_i, q_j)$. Пусть, например, коды состояний автомата, поведение которого описывается с помощью табл. 26.1, представлены приведенной выше матрицей $\boldsymbol{C}$. Для кодирования входных сигналов используются переменные x_1, x_2, значениями которых являются соответственно 0, 0 для a_1, 0, 1 – для a_2 и 1, 0 – для a_3. Тогда булевы функции, описывающие рассматриваемый автомат, можно задать следующими матрицами, в которых удалены строки, соответствующие поглощаемым интервалам:

$$\boldsymbol{U} = \begin{array}{c} \begin{matrix} x_1 & x_2 & z_1 & z_2 & z_3 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & - & - \\ 0 & 0 & 1 & - & - \\ 0 & 1 & - & - & - \\ 1 & 0 & - & 0 & - \\ 1 & 0 & - & 1 & - \end{bmatrix} \end{array}, \quad \boldsymbol{V} = \begin{array}{c} \begin{matrix} z_1^+ & z_2^+ & z_3^+ \end{matrix} \\ \begin{bmatrix} 0 & 0 & - \\ 1 & 1 & - \\ 0 & 1 & 1 \\ 1 & 0 & - \\ 0 & 1 & 0 \end{bmatrix} \end{array}.$$

26.3. Минимизация длины кода

Кодирование состояний, обеспечивающее отсутствие опасных состязаний (гонок), называется *противогоночным*. Естественно, здесь возникает задача минимизации длины кода состояния, приводящая к наименьшему числу элементов памяти в реальной схеме.

Условия отсутствия опасных состязаний можно выразить троичной матрицей, в которой столбцы соответствуют состояниям автомата и для каждой пары переходов $q_i \to q_j$, $q_k \to q_l$ имеется хотя бы одна строка, где компоненты i и j имеют одно значение, 0 или 1, а компоненты k и l – противоположное ему значение. Остальные компоненты имеют значение «–». Эта матрица может быть избыточна, если некоторые строки находятся в отношении импликации.

Троичный вектор $\boldsymbol{a}$ *имплицирует* троичный вектор $\boldsymbol{b}$, если $\boldsymbol{b}$ получается из $\boldsymbol{a}$ заменой некоторых нулей или единиц значением «–» и, возможно, инвертированием полученного результата. Например, вектор (1 0 – – 1 0 1) имплицирует (1 0 – – – 0 1) и (0 1 – – – 1 –). Смысл этого отношения в том, что

условие, представленное вектором $\boldsymbol{b}$, автоматически выполняется при соблюдении условия, представленного вектором $\boldsymbol{a}$. Данное отношение обладает свойствами рефлексивности и транзитивности.

Представим все условия отсутствия опасных состязаний в виде троичной матрицы, удалим из нее имплицируемые строки и назовем полученную матрицу *матрицей условий*.

Для автомата, таблицей переходов которого является табл. 26.1, условия отсутствия опасных состязаний, определяемых парами переходов $q_1 \to q_1$, $q_4 \to q_5$; $q_1 \to q_1$, $q_5 \to q_5$ и $q_2 \to q_1$, $q_4 \to q_5$, могут быть представлены следующими строками:

$$\begin{array}{ccccc} q_1 & q_2 & q_3 & q_4 & q_5 \\ 0 & - & - & 1 & 1 \\ 0 & - & - & - & 1 \\ 0 & 0 & - & 1 & 1. \end{array}$$

Здесь только последняя строка останется в матрице условий. Остальные строки имплицируются последней строкой и в матрицу условий не включаются.

Будем говорить, что троичная матрица $\boldsymbol{R}$ *имплицирует* троичную матрицу $\boldsymbol{S}$, если для каждой строки матрицы $\boldsymbol{S}$ в матрице $\boldsymbol{R}$ найдется имплицирующая ее строка.

Задача противогоночного кодирования сводится к нахождению матрицы с минимальным числом строк, имплицирующей матрицу условий. Столбцы этой матрицы будут представлять искомые коды состояний, т. е., чтобы получить матрицу кодирования $\boldsymbol{C}$, надо транспонировать *кратчайшую имплицирующую форму* матрицы условий.

Множество строк матрицы условий называется *совместимым*, если существует вектор, имплицирующий каждую строку этого множества. Заметим, что множество строк, в котором каждая пара строк является совместимой, не всегда является совместимым множеством. Действительно, пусть имеется три троичных вектора: $\boldsymbol{u}$ = (0 1 – – 0 1), $\boldsymbol{v}$ = (– 1 0 – 0 –) и $\boldsymbol{w}$ = (– –1 – – 1). Для $\boldsymbol{u}$ и $\boldsymbol{v}$ общим имплицирующим вектором является (0 1 0 – 0 1), для $\boldsymbol{u}$ и $\boldsymbol{w}$ – вектор (0 1 1 – 0 1), а для $\boldsymbol{v}$ и $\boldsymbol{w}$ – вектор (– 0 1 – 1 1). Любой имплицирующий вектор для двух векторов не является таковым для третьего. Совместимое множество называется *максимальным*, если оно не является собственным подмножеством другого совместимого множества.

Чтобы получить кратчайшую имплицирующую форму для троичной матрицы, надо найти все максимальные совместимые множества ее строк, а затем получить кратчайшее покрытие строк этими множествами.

Пусть табл. 26.2 представляет таблицу переходов заданного асинхронного автомата. Для того чтобы учесть требование ортогональности кодов для различных состояний, можно ввести в таблицу переходов еще один столбец, где присутствуют все состояния, которые являются устойчивыми. Тогда выполнение данного требования обеспечивается введением в матрицу условий строк, не имплицируемых другими строками и выражающих отсутствие

опасных состязаний, которые соответствуют данному столбцу. При этом табл. 26.2 примет вид табл. 26.3. Заметим, что неортогональные коды могут оказаться у состояний, которым соответствуют одинаковые строки таблицы переходов, и введение дополнительного столбца делает все строки различными.

Таблица 26.2

Таблица переходов асинхронного автомата

	a_1	a_2	a_3	a_4
q_1	$\boldsymbol{q}_1$	$\boldsymbol{q}_1$	$\boldsymbol{q}_1$	q_5
q_2	q_1	$\boldsymbol{q}_2$	$\boldsymbol{q}_2$	$\boldsymbol{q}_2$
q_3	$\boldsymbol{q}_3$	q_2	q_4	q_5
q_4	q_3	q_2	$\boldsymbol{q}_4$	q_5
q_5	–	$\boldsymbol{q}_5$	q_4	$\boldsymbol{q}_5$

Таблица 26.3

Таблица переходов с дополнительным столбцом

	a_1	a_2	a_3	a_4	a_5
q_1	$\boldsymbol{q}_1$	$\boldsymbol{q}_1$	$\boldsymbol{q}_1$	q_5	$\boldsymbol{q}_1$
q_2	q_1	$\boldsymbol{q}_2$	$\boldsymbol{q}_2$	$\boldsymbol{q}_2$	$\boldsymbol{q}_2$
q_3	$\boldsymbol{q}_3$	q_2	q_4	q_5	$\boldsymbol{q}_3$
q_4	q_3	q_2	$\boldsymbol{q}_4$	q_5	$\boldsymbol{q}_4$
q_5	–	$\boldsymbol{q}_5$	q_4	$\boldsymbol{q}_5$	$\boldsymbol{q}_5$

Анализируя табл. 26.3, нетрудно найти, в частности, что условия отсутствия опасных состязаний при входном сигнала a_1 выражаются следующими троичными векторами (справа показаны рассматриваемые при этом пары переходов)

$q_1\ q_2\ q_3\ q_4\ q_5$

$0\ -\ 1\ -\ -$ $\quad q_1 \to q_1,\ q_3 \to q_3;$

$0\ 0\ 1\ -\ -$ $\quad q_2 \to q_1,\ q_3 \to q_3;$

$0\ -\ 1\ 1\ -$ $\quad q_1 \to q_1,\ q_4 \to q_3;$

$0\ 0\ 1\ 1\ -$ $\quad q_2 \to q_1,\ q_4 \to q_3.$

В данном случае строка $(0\ 0\ 1\ 1\ -)$ имплицирует каждую из трех остальных, которые, следовательно, не надо включать в матрицу условий. Рассматривая остальные пары переходов и формируя для них соответствующие троичные векторы, получим следующую матрицу условий, которая для удобства обозрения разбита на секции, соответствующие различным столбцам табл. 26.3:

$$\boldsymbol{S} = \begin{array}{c} \begin{matrix} q_1 & q_2 & q_3 & q_4 & q_5 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1 & 1 & - \\ 0 & 1 & 1 & - & - \\ 0 & 1 & - & 1 & - \\ - & 0 & 0 & - & 1 \\ - & 0 & - & 0 & 1 \\ 0 & - & - & 1 & 1 \\ - & 0 & - & 1 & 1 \\ 0 & 1 & - & - & 0 \\ - & 0 & 1 & - & 1 \\ - & - & 0 & 1 & - \end{bmatrix} \end{array} \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{l} \} \; a_1 \\ \\ \} \; a_2 \\ \\ \\ \} \; a_3 \\ \\ \} \; a_4 \\ \\ \} \; a_5 \end{array} .$$

Каждому совместимому множеству строк троичной матрицы согласно его определению соответствует вектор, имплицирующий все строки, являющиеся элементами этого множества. Для данной троичной матрицы ***S*** получим следующие максимальные совместимые множества и соответствующие им векторы:

{1, 6, 7, 9}	(0 0 1 1 1);
{2, 3, 4, 5, 8}	(0 1 1 1 0);
{2, 3, 6}	(0 1 1 1 1);
{2, 4, 7, 8, 10}	(0 1 1 0 0);
{3, 5, 8, 9, 10}	(0 1 0 1 0);
{3, 6, 10}	(0 1 0 1 1);
{4, 6, 7, 10}	(0 0 0 1 1);
{7, 8, 9}	(0 1 0 0 0).

Кратчайшее покрытие множества строк матрицы ***S*** составляют множества {1, 6, 7, 9}, {2, 3, 4, 5, 8} и {2, 4, 7, 8, 10}. Соответствующие им векторы являются вектор-столбцами матрицы кодирования заданного асинхронного автомата:

$$\boldsymbol{C} = \begin{array}{c} \begin{matrix} z_1 & z_2 & z_3 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{l} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{array} .$$

26.4. Рассмотрение *K*-множеств

Для ускорения процесса противогоночного кодирования состояний асинхронного автомата вместо рассмотрения пар переходов можно рассматривать пары так называемых *K*-множеств.

K-множеством называется множество состояний асинхронного автомата, переходы из которых при некотором фиксированном входном сигнале ведут в одно и то же состояние (также принадлежащее данному множеству). Пусть таблицу переходов автомата представляет табл. 26.4. *K*-множествами при входном сигнале a_1 являются $\{q_1, q_3\}$, $\{q_2, q_4, q_5\}$ и $\{q_6, q_7\}$.

Таблица 26.4

Таблица переходов асинхронного автомата

	a_1	a_2	a_3	a_4	a_5
q_1	$\boldsymbol{q}_1$	q_5	$\boldsymbol{q}_1$	q_5	q_2
q_2	$\boldsymbol{q}_2$	q_6	$\boldsymbol{q}_2$	–	$\boldsymbol{q}_2$
q_3	q_1	–	q_4	$\boldsymbol{q}_3$	–
q_4	q_2	q_5	$\boldsymbol{q}_4$	–	q_6
q_5	q_2	$\boldsymbol{q}_5$	q_1	$\boldsymbol{q}_5$	–
q_6	q_7	$\boldsymbol{q}_6$	q_2	q_3	$\boldsymbol{q}_6$
q_7	$\boldsymbol{q}_7$	q_5	–	q_3	q_6

Парой K-множеств в дальнейшем будем называть два различных *K*-множества, построенных для одного и того же входного сигнала. Так же, как для пары переходов, для каждой пары *K*-множеств строится троичный вектор, компоненты которого соответствуют состояниям автомата. Компоненты, соответствующие состояниям, которые принадлежат одному из этих *K*-множеств, имеют значение 0; компоненты, соответствующие состояниям из другого *K*-множества, – значение 1, а компоненты, соответствующие состояниям, не принадлежащим ни одному их этих *K*-множеств, – значение «–». Множество построенных таким образом векторов для всех пар *K*-множеств, из которого удалены имплицируемые векторы, образует матрицу условий ***S****, имплицирующую матрицу ***S***.

Действительно, каждой из строк матрицы ***S***, полученной при рассмотрении пары переходов $q_i \to q_j$, $q_k \to q_l$, соответствует некоторая строка матрицы ***S****, полученная при рассмотрении пары *K*-множеств, одному из которых принадлежат состояния q_i и q_j, а другому – состояния q_k и q_l. При этом строка матрицы ***S**** имплицирует строку матрицы ***S***. Благодаря транзитивности отношения импликации кратчайшая имплицирующая форма матрицы ***S**** имплицирует также и матрицу ***S***.

Процесс получения кратчайшей имплицирующей формы матрицы ***S**** ничем не отличается от описанного ранее процесса получения кратчайшей имплицирующей формы матрицы ***S***, т. е. так же строятся максимальные

совместимые множества строк и отыскивается кратчайшее покрытие строк матрицы $\boldsymbol{S}^*$ этими множествами.

В общем случае матрица $\boldsymbol{S}^*$ неэквивалентна матрице $\boldsymbol{S}$, т. е. не всегда матрица $\boldsymbol{S}$ имплицирует $\boldsymbol{S}^*$. Поэтому для одного и того же автомата кратчайшая имплицирующая форма матрицы $\boldsymbol{S}^*$ может иметь большее число строк, чем кратчайшая имплицирующая форма матрицы $\boldsymbol{S}$. Однако рассмотрение пар K-множеств менее трудоемко по сравнению с рассмотрением пар переходов.

Проиллюстрируем данный метод кодирования на примере табл. 26.4. Здесь нет необходимости вводить дополнительный столбец, как это сделано в предыдущем примере, поскольку нет одинаковых строк. Для данного автомата получаем матрицу условий

$$\boldsymbol{S}^* = \begin{array}{c} \begin{matrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & - & - \\ 0 & - & 0 & - & - & 1 & 1 \\ - & 0 & - & 0 & 0 & 1 & 1 \\ 0 & 1 & - & 0 & 0 & 1 & 0 \\ 0 & - & 1 & 1 & 0 & - & - \\ - & 0 & 1 & 1 & - & 0 & - \\ 0 & - & 1 & - & 0 & 1 & 1 \\ 0 & 0 & - & 1 & - & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix} .$$

Для троичной матрицы $\boldsymbol{S}^*$ получим следующие максимальные совместимые множества и соответствующие им векторы:

$\{1, 2\}$ (0 1 0 1 1 1 1);
$\{1, 3\}$ (0 1 0 1 1 0 0);
$\{2, 3\}$ (0 0 0 0 0 1 1);
$\{2, 6\}$ (0 1 0 0 – 1 1);
$\{2, 8\}$ (0 0 0 1 – 1 1);
$\{3, 7\}$ (0 0 1 0 0 1 1),
$\{4, 6\}$ (0 1 0 0 0 1 0);
$\{5, 6\}$ (0 0 1 1 0 0 –);
$\{5, 7, 8\}$ (0 0 1 1 0 1 1).

Совокупность множеств $\{1, 2\}$, $\{1, 3\}$, $\{4, 6\}$ и $\{5, 7, 8\}$ является кратчайшим покрытием множества строк матрицы $\boldsymbol{S}^*$. Матрица кодирования для рассматриваемого автомата имеет вид

$$\boldsymbol{C} = \begin{array}{c} \begin{matrix} z_1 & z_2 & z_3 & z_4 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{matrix}.$$

Функции возбуждения строятся на интервалах, каждый из которых соответствует *K*-множеству. Для этого представим входной сигнал многозначной переменной a, принимающей значения a_1, a_2, a_3, a_4, a_5, и получим следующие матрицы:

$$\boldsymbol{U} = \begin{array}{c} \begin{matrix} a & z_1 & z_2 & z_3 & z_4 \end{matrix} \\ \begin{bmatrix} a_1 & 0 & 0 & 0 & - \\ a_1 & 1 & 1 & - & - \\ a_1 & 1 & 0 & - & 1 \\ a_2 & - & - & 0 & - \\ a_2 & 1 & - & 1 & - \\ a_3 & - & - & 0 & 0 \\ a_3 & 1 & - & 1 & - \\ a_3 & - & - & 0 & 1 \\ a_4 & - & - & 0 & 0 \\ a_4 & - & 0 & - & 1 \\ a_5 & - & - & - & 0 \\ a_5 & 1 & - & - & 1 \end{bmatrix} \end{array}, \quad \boldsymbol{V} = \begin{array}{c} \begin{matrix} z_1^+ & z_2^+ & z_3^+ & z_4^+ \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \end{array}.$$

Полученные функции можно упростить за счет расширения интервалов:

$$\boldsymbol{U} = \begin{array}{c} \begin{matrix} a & z_1 & z_2 & z_3 & z_4 \end{matrix} \\ \begin{bmatrix} a_1 & 0 & 0 & - & - \\ a_1 & - & 1 & - & - \\ a_1 & 1 & 0 & - & - \\ a_2 & - & - & 0 & - \\ a_2 & - & - & 1 & - \\ a_3 & - & - & 0 & 0 \\ a_3 & - & - & 1 & - \\ a_3 & - & - & 0 & 1 \\ a_4 & - & - & - & 0 \\ a_4 & - & - & - & 1 \\ a_5 & - & - & - & 0 \\ a_5 & - & - & - & 1 \end{bmatrix} \end{array}, \quad \boldsymbol{V} = \begin{array}{c} \begin{matrix} z_1^+ & z_2^+ & z_3^+ & z_4^+ \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \end{array}.$$

26.5. Соседнее кодирование состояний

До сих пор рассматривались прямые переходы, когда элементы памяти одновременно меняют свои состояния. Можно избавиться от опасных состязаний, устранив вообще состязания между элементами памяти. Каждый из прямых переходов можно заменить последовательностью *элементарных переходов*, т. е. таких, в которых меняется состояние только одного элемента памяти. В этом случае удобно представить коды состояний как элементы булева пространства внутренних переменных, а переход из состояния в состояние – как движение по ребрам полного булева графа, или гиперкуба, из вершины в вершину. Упомянутая последовательность элементарных переходов представится тогда последовательностью пройденных ребер.

Кодирование состояний сводится к размещению состояний на вершинах гиперкуба. Можно использовать также процесс сборки гиперкуба, применяемый при кодировании синхронного автомата методом «желательных соседств». Рассмотрим использование этого процесса для соседнего кодирования состояний асинхронного автомата.

В данном случае также вводится величина w_{ij}, задаваемая на парах состояний автомата, только теперь она представляет коэффициент связи – число *K*-множеств, содержащих пару состояний q_i, q_j. Дальнейшая сборка гиперкуба ничем не отличается от той, которая применяется при кодировании состояний синхронного автомата. Заметим только, что при подсчете значения w_{ij} одно и то же *K*-множество учитывается столько раз, сколько раз оно определяется различными входными сигналами.

После того как гиперкуб собран, в нем выделяются подграфы, порожденные *K*-множествами. Эти подграфы должны быть связными, и на них накладывается следующее ограничение. Подграфы, которые соответствуют *K*-множествам, определяемым одним и тем же входным сигналом, не должны пересекаться.

В *K*-множестве устойчивым является только одно состояние. В подграфе, соответствующем *K*-множеству, можно сориентировать ребра согласно переходам, заканчивающимся всегда в устойчивом состоянии. Состояния, связанные элементарным переходом, располагаются на соседних вершинах. Если не удается выделить связные подграфы в гиперкубе, непересекающиеся и соответствующие *K*-множествам, которые определяются одним и тем же входным сигналом, надо увеличить размерность гиперкуба на единицу и проложить цепочку элементарных переходов, по крайней мере, для одного прямого перехода через другой подкуб.

Обозначим *K*-множество, определяемое входным сигналом a_i и устойчивым состоянием q_j, символом K_{ij} и рассмотрим автомат, заданный табл. 26.4. Для него получим все *K*-множества:

$K_{11} = \{q_1, q_3\}$, $K_{12} = \{q_2, q_4, q_5\}$, $K_{17} = \{q_6, q_7\}$;
$K_{25} = \{q_1, q_4, q_5, q_7\}$, $K_{26} = \{q_2, q_6\}$;
$K_{31} = \{q_1, q_5\}$, $K_{32} = \{q_2, q_6\}$, $K_{34} = \{q_3, q_4\}$;
$K_{43} = \{q_3, q_6, q_7\}$, $K_{45} = \{q_1, q_5\}$;
$K_{52} = \{q_1, q_2\}$, $K_{56} = \{q_4, q_6, q_7\}$.

Значения w_{ij} представим с помощью табл. 26.5. Результат процесса сборки гиперкуба показан на рис. 26.1.

Таблица 26.5

Значения w_{ij}

q_2	q_3	q_4	q_5	q_6	q_7	q_8	
1	1	1	3	0	1	0	q_1
	0	1	1	2	0	0	q_2
		1	0	1	1	0	q_3
			2	1	2	0	q_4
				0	1	0	q_5
					3	0	q_6
						0	q_7

Разместим теперь в этом гиперкубе связные подграфы, соответствующие K-множествам, и снабдим ребра этих подграфов ориентацией, соответствующей направлениям переходов (рис. 26.2). В данном примере удалось без увеличения размерности гиперкуба разместить их так, чтобы любые два из них, соответствующие одному и тому же входному сигналу, не пересекались.

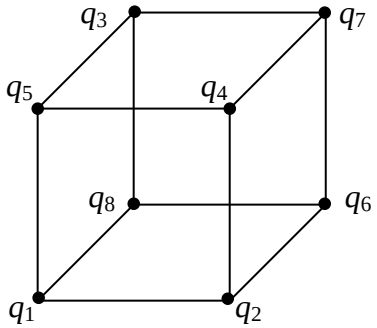


Рис. 26.1. Гиперкуб, представляющий соседство состояний

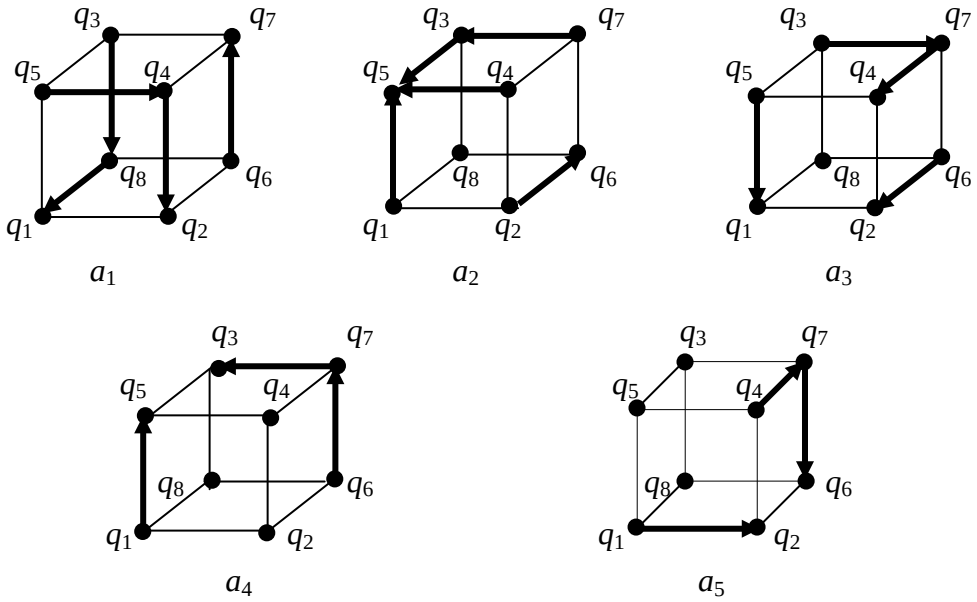


Рис. 26.2. Цепочки элементарных переходов

Преобразуем теперь исходную таблицу в *таблицу элементарных переходов*, которая представлена в виде табл. 26.6.

Таблица 26.6

Таблица элементарных переходов

	a_1	a_2	a_3	a_4	a_5
q_1	$\boldsymbol{q}_1$	q_5	$\boldsymbol{q}_1$	q_5	q_2
q_2	$\boldsymbol{q}_2$	q_6	$\boldsymbol{q}_2$	–	$\boldsymbol{q}_2$
q_3	q_8	q_5	q_7	$\boldsymbol{q}_3$	–
q_4	q_2	q_5	$\boldsymbol{q}_4$	–	q_7
q_5	q_4	$\boldsymbol{q}_5$	q_1	$\boldsymbol{q}_5$	–
q_6	q_7	$\boldsymbol{q}_6$	q_2	q_7	$\boldsymbol{q}_6$
q_7	$\boldsymbol{q}_7$	q_3	q_4	q_3	q_6
q_8	q_1	–	–	–	–

Функции возбуждения элементов памяти строятся здесь так же, как для синхронного автомата.

Литература

1. Андерсон Д.А. Дискретная математика и комбинаторика. – М.: Издательский дом «Вильямс», 2003. – 960 с.

2. Ангер С. Асинхронные последовательностные схемы. – М.: Наука, 1977. – 400 с.

3. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 536 с.

4. Баранов С.И. Синтез микропрограммных автоматов. – Л.: Энергия, 1979. – 232 с.

5. Быкова С.В., Буркатовская Ю.Б. Булевы функции: Учебное пособие. – Томск: Томский государственный университет, 2010. – 192 с.

6. Вавилов Е.Н., Портной Г.П. Синтез схем электронных цифровых машин. – М.: Советское радио, 1964. - 440 с.

7. Глушков В.М. Синтез цифровых автоматов. – М.: ГИФМЛ, 1962. – 476 с.

8. Закревский А.Д. Логический синтез каскадных схем. – М.: Наука. 1981. – 414 с.

9. Закревский А.Д., Поттосин Ю.В., Черемисинова Л.Д. Основы логического проектирования. В 3 кн. Кн.1. Комбинаторные алгоритмы дискретной математики. – Мн.: ОИПИ НАН Беларуси, 2004. – 226 с.

10. Закревский А.Д., Поттосин Ю.В., Черемисинова Л.Д. Основы логического проектирования. В 3 кн. Кн.2. Оптимизация в булевом пространстве. – Мн.: ОИПИ НАН Беларуси, 2004. – 240 с.

11. Закревский А.Д., Поттосин Ю.В., Черемисинова Л.Д. Основы логического проектирования. В 3 кн. Кн.3. Проектирование устройств логического управления. – Мн.: ОИПИ НАН Беларуси, 2006. – 254 с.

12. Закревский А.Д., Поттосин Ю.В., Черемисинова Л.Д. Логические основы проектирования дискретных устройств. – М.: ФИЗМАТЛИТ, 2007. – 592 с.

13. Зыков А.А. Основы теории графов. – М.: Наука, 1987. – 384 с.

14. Кнут Д.Э. Искусство программирования, том 4, А. Комбинаторные алгоритмы, часть 1. – М.: ООО «И. Д. Вильямс», 2013. – 960 с.

15. Кристофидес Н. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 с.

16. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженеров. – М.: Энергия, 1988. – 480 с.

17. Лазарев В.Г., Пийль Е.И. Синтез управляющих автоматов. – М.: Энергоатомиздат, 1989. – 328 с.

18. Лекции по теории графов / Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. – М.: Наука, 1990. – 384 с.

19. Миллер Р. Теория переключательных схем, т.I. – М.: Наука, 1970. – 416 с.

20. Миллер Р. Теория переключательных схем, т.II. – М.: Наука, 1971. – 304 с.

21. Новиков Ф.А. Дискретная математика для программистов. (2-е изд.). – М., С.-Пб.: Питер. – 2005. – 364 с.

22. Поспелов Д.А. Логические методы анализа и синтеза схем. – М.; Л.: Энергия, 1964. – 320 с.

23. Поттосин Ю.В. Основы теории проектирования цифровых устройств. – Saarbrücken: LAP LAMBERT Academic Publishing, 2011. – 336 с.

24. Скляров В.А. Синтез автоматов на матричных БИС. – Минск: Наука и техника, 1984. – 287 с.

25. Фридман А., Менон П. Теория и проектирование переключательных схем. – М.: Мир, 1978. – 580 с.

26. Яблонский С.В. Введение в дискретную математику. – М.: Наука, 1986. – 384 с.

Предметный указатель