Парадигма развития науки

Методологическое обеспечение

А. Е. Кононюк

ДИСКРЕТНО-НЕПРЕРЫВНАЯ МАТЕМАТИКА

Книга 11

Автоматы

Часть 2

Детерминированные автоматы

(Окончание)

Киев
«Освіта України»
2017

УДК 51 (075.8)
ББК В161.я7
К213

Рецензенты:

В. В. Довгай — к-т физ.-мат. наук, доц. (Национальный тех— нический университет «КПІ»);

В. В. Гавриленко — д-р физ.-мат. наук, проф.,

О. П. Будя — к-т техн. наук, доц. (Киевский университет эко— номики, туризма и права);

Н. К. Печурин — д-р техн. наук, проф. (Национальный ави— ационный университет).

Кононюк А. Е.
К213 Дискретно-непрерывная математика. (Автоматы). — В 12-и кн. Кн. 11,Ч.2— К.: 2017. —578 с.

ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-10 (книга 11, ч.2)

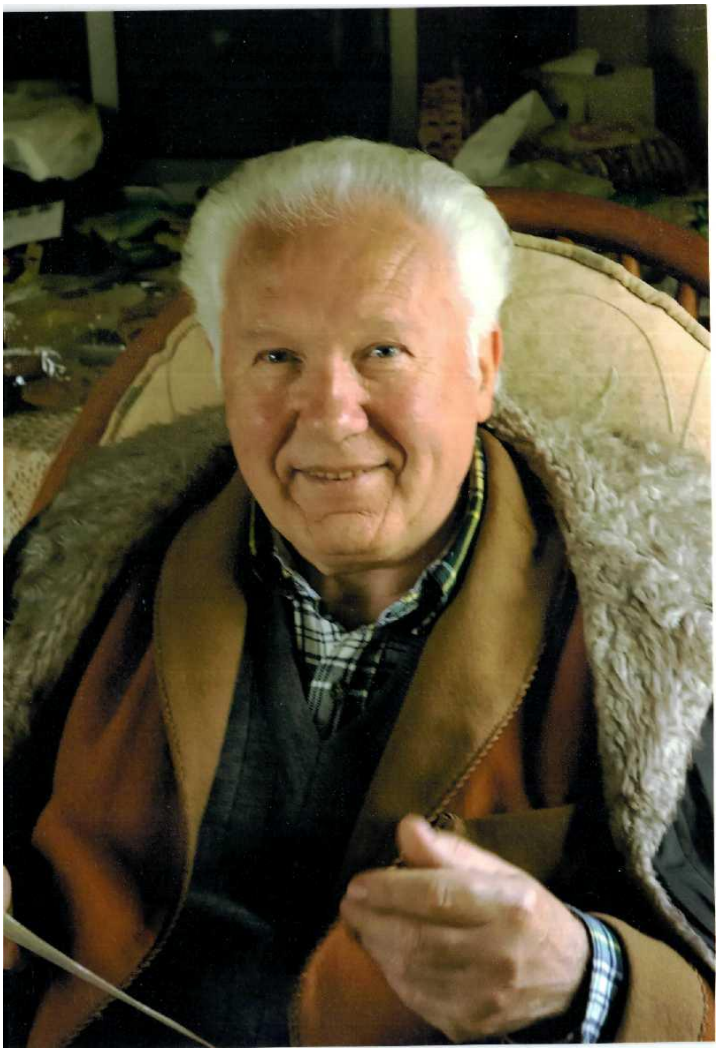
Многотомная работа содержит систематическое изложение математических дисциплин, исспользуемых при моделировании и исследованиях математических моделей систем.

В работе излагаются основы теории множеств, отношений, поверхностей, пространств, алгебраических систем, матриц, графов, математической логики, теории вероятностей и массового обслуживания, теории формальных грамматик и автоматов, теории алгоритмов, которые в совокупности образуют единную методолгически взамосвязанную математическую систему «Дискретно-непрерывная математика».

Для бакалавров, специалистов, магистров, аспирантов, докторантов и просто ученых и специалистов всех специальностей.

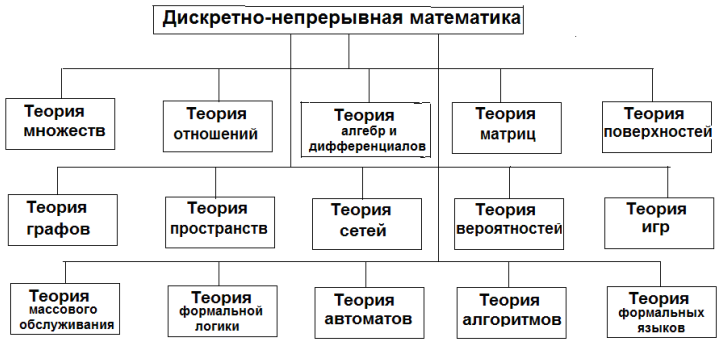
УДК 51 (075.8)
ББК В161.я7

ISBN 978-966-373-693-8 (многотомное издание)
ISBN 978-966-373-694-10 (книга 11, ч. 2)



Кононюк Анатолий Ефимович

Структура
открытой развивающейся панмедийной системы математических наук (дисциплин)
"Дискретно-непрерывная математика"



Оглавление

1. Введение в анализ и синтез математических моделей автоматов

1.1. Основные понятия теории линейных переходных графов

В этом разделе формулируются алгоритмы анализа и синтеза математических моделей автоматов, основанные на методах теории линейных переходных графов или, как говорят, линейных графов сигналов (signal flow grahps), а также на решении систем уравнений в алгебре событий.

Переходным графом сигналов принято называть такой ориентированный граф, каждой вершине i которого соответствует сигнал (весовая функция) a_i, зависящий некоторым образом от сигналов (весовых функций) на других вершинах.

Путь на графе представляет собой последовательность смежных дуг, причем каждой дуге, соединяющей вершины a_i и a_j, соответствует одна из букв алфавита $X = \{x_1, x_2, ..., x_m\}$, через которые обозначены веса дуг. Вес каждой дуги, соединяющей вершины a_i и a_j, соответствует переходу t_{ij}, показывающему, как сигнал a_j относится к сигналу a_i. Если обозначить вершины графа $a_1, a_2, ..., a_n$, то путь, проходящий через вершины с возрастающим номером, называется *прямым* путем, а путь, вдоль которого номера вершин убывают, называется *обратным* путем. Путь на графе может быть *открытым,* если одна и та же вершина встречается не более одного раза, и *замкнутым,* если путь возвращается к начальной вершине. Путь может быть и незамкнутым, и неоткрытым. Последовательность дуг, входящих в замкнутый путь, называется *контуром обратной связи.* Дуги и вершины, не входящие в контур обратной связи, называются *каскадными.* Вершина называется *стоком,* если она имеет только входящие дуги, и *источником,* если она имеет только выходящие дуги. Две вершины или дуги, лежащие в общем контуре обратной связи, называются *спаренными.*

На рис. 1.1 показан переходный граф сигналов.

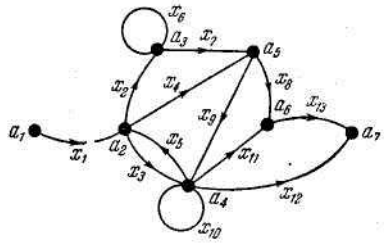


Рис. 1.1.

Пути $x_1x_2\,x_7x_8\,x_{13}$, $x_1x_3\,x_{11}x_{13}$, $x_1x_4x_8\,x_{13}$ являются прямыми, путь x_9x_5 — обратным. Все эти пути являются открытыми, тогда как $x_9x_5x_4$, x_{10}, x_6 — замкнутыми, а пути $x_1x_3\,x_{10}x_{12}$, $x_1x_2x_6x_7$ и не открыты, и не замкнуты. Дуги, соединяющие вершины a_1 и a_2, a_5 и a_6, a_4 и a_6, a_4 и a_7, a_6 и a_7, — каскадные, а остальные — дуги обратной связи.

Вершина a_1 является источником, а вершина a_7 — стоком. Вершины a_2,a_3,a_4,a_5 — вершины обратной связи, а вершины a_1, a_6, a_7 — каскадные. Вершины a_2 и a_4 спаренные, так как они лежат в общем контуре обратной связи.

Граф, состоящий только из каскадных вершин, называется *каскадным,* а граф, содержащий одну или более вершин обратной связи, называется *графом обратной связи.* Соединение обратной связи представляет собой подграф, в котором каждая пара вершин спарена. Такой подграф состоит только из вершин и ветвей обратной связи.

Поскольку те вершины графа, которые являются стоком и истоком, всегда каскадные, то любую из вершин соединения обратной связи можно сделать каскадной с помощью операции, называемой *расщеплением вершины*.

Произвольная вершина a_j в этом случае расщепляется на две вершины: a_j', которая является истоком, и a_j'', которая служит стоком. На рис. 1.2 показан пример расщепления вершины a_4 графа, приведенного на рис. 1.1.

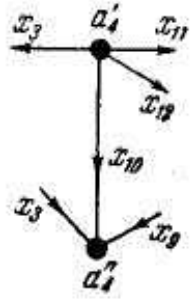


Рис. 1.2.

Минимальное число вершин, которые нужно расщепить, чтобы разбить все контуры обратной связи, называется *индексом соединения* обратной связи. Расщепленные вершины называются *индексными* вершинами.

Переходный граф сигналов можно упростить, устранив ряд вершин и приведя ветвевой переход к значению пути через устраненные вершины. Полученный граф называется *остатком* первоначального графа. Вершина, входящая в остаточный граф, называется *остаточной.* Путь, если он связывает остаточную вершину с собой или с заданной остаточной вершиной и не проходит через любые другие остаточные вершины, кроме указанных, также является *остаточным.* Исключая в исходном графе все вершины, кроме источников, стоков и индексных вершин, получим *индексный остаток* переходного графа. Пример индексного остатка графа, приведенного на рис. рис. 1.1, показан на рис. 1.3, *а*.

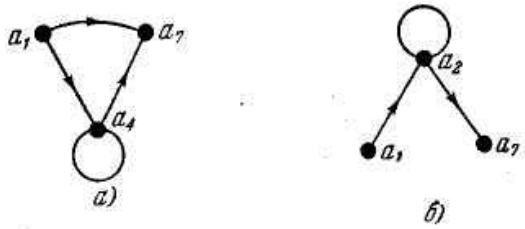


Рис. 1.3.

В качестве индексной вершины выбрана вершина a_4, так как ее расщепление устраняет все контуры обратной связи. Вместо вершины a_4 может быть выбрана вершина a_2. Тогда индексный остаток графа имеет вид рис. 1.3,*б*. Пути $x_1x_2\,x_7x_8\,x_{13}$ и $x_1x_4\,x_8x_{13}$ на графе рис. 1.3, *а* являются примерами остаточных путей. Путь $x_1x_3\,x_5x_4\,x_8\,x_{13}$ не является остаточным, так как он проходит через остаточную вершину a_4.

Если необходимо сохранить вершину a_i, которая не является ни источником, ни стоком, ни индексной вершиной, то это можно сделать путем соединения вершины a_i с новой вершиной a_i' дугой с весом e, а затем устранением вершины a_i. Сигнал x'_i таким образом, равен сигналу x_i, и так как вершина a_i' не имеет других связей, то она является истоком. Аналогично вершина может быть превращена в сток путем подстанорки новой вершины a_i'' с соединением перехода от a_i к a_i'', а затем устранением вершины a_i.

Переходный граф сигналов является *линейным,* если сигнал на каждой вершине a_i является линейной комбинацией сигналов на других вершинах:

$$a_i = \sum_j a_j t_{ji}, \qquad i,\ j \in I = \{1, 2, \ldots, n\}.$$

Нетрудно убедиться, что графоид автомата является линейным переходным графом сигналов, если:

1) *сигнал, соответствующий каждой вершине a_i, является множеством x_{1i} последовательностей, которые переводят автомат из начального состояния a_1 в состояние q_i;*

2) *ветвевой переход t_{ij} дуги, связывающей вершины a_i и a_j (или состояния q_i и q_j), является входным сигналом $x \in X$, который вызывает переход автомата из состояния q_i в состояние q_j;*

3) *к сигналу, соответствующему начальному состоянию q_1, добавляется пустой элемент e для обозначения того, что последовательность нулевой длины оставляет автомат в состоянии q_1.*

Сигнал на каждой вершине q_j, т. е. множество последовательностей x_{1j}, можно записать следующим образом:

$$x_{1j} = \sum_{q_i \in p_j} x_{1i} t_{ij} + \delta(q_j), \qquad (1.1)$$

где t_{ij} — дизъюнкция входных элементов, переводящих автомат из состояния q_i в q_j и $\delta(q_j) = e$, если $j = 1$, и $\delta(q_j) = \varnothing$, если $j \neq 1$, где q_1— начальное состояние автомата. Следовательно, тождественные преобразования, применяемые к линейным переходным графам, естественным образом распространяются и на графоиды автоматов, с учетом лишь того, что умножение здесь не коммутативно.

На рис. 1.4 показаны элементарные тождества для каскадных переходных графов сигналов.

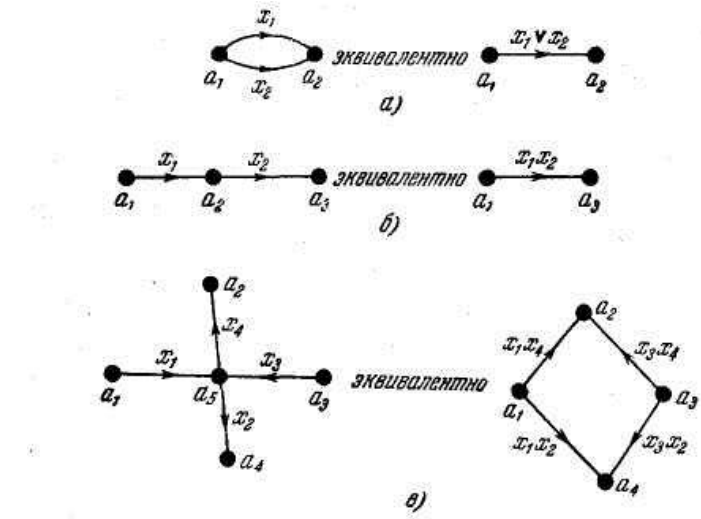


Рис. 1.4.

Из рис. 1.4, *а* следует, что из вершины a_1 можно попасть в a_2 по дуге с весом x_1 или x_2, а из рис. 1.4,*б* видно, что из a_1 в a_3 можно попасть, пройдя последовательно дугу с весом x_1, а затем дугу с весом x_2. На рис. 1.4, *в* показано преобразование узла в четырехугольник.

В теории переходных графов сигналов петля соответствует делению. По формуле (1.1) для графа, показанного на рис. 1.5,

Рис. 1.5.

можно составить систему уравнений

$$\begin{cases} a_1 = e, \\ a_2 = a_1 \cdot 1 + a_2 \cdot x, \\ a_3 = a_2(-1). \end{cases} \qquad (1.2)$$

Решая систему (1.2) относительно a_2, находим

$$a_2 - a_2 x = a_1 \cdot 1,$$

$$a_2(1 - x) = a_1 \cdot 1,$$

$$a_2 = \frac{a_1 \cdot 1}{1 - x}.$$

Известно, что для $|x| < 1$ имеет место

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots \qquad (1.3)$$

В алгебре событий это аналогично итерации

$$\{x\}^* = e \vee x \vee x^2 \vee x^3 \vee \ldots. \qquad (1.4)$$

геометрическая интерпретация которой имеет вид, показанный на рис. 1.6.

Рис. 1.6.

Переходный граф с контуром обратной связи показан на рис. 1.7.

Рис. 1.7.

Нетрудно установить, что множество путей, ведущих из вершины a_1 в вершину a_3, является бесконечной суммой

$R = x_1x_2 + x_1x_2x_3x_2 + x_1x_2\,(x_3x_2)^2 + x_1x_2\,(x_3x_2)^2 + \ldots =$

$= x_1x_2\,[1 + x_3x_2 + (x_3x_2)^2 + \ldots\,]$.

Учитывая (1.3.) и (1.4), можно записать

$R = x_1x_2[e \vee x_3x_2 \vee (x_3x_2)^2 \vee \ldots] = x_1x_2\{x_3x_2\}^*$.

Поэтому описанные тождественные преобразования можно использовать для получения регулярных выражений по графоиду автомата.

1.2. Алгоритм анализа автоматов

Одной из важных проблем математической теории автоматов является проблема анализа, состоящая в отыскании регулярного выражения, представимого в конечном автомате. Вопросы анализа конечных автоматов рассмотрены в работах В. М. Глушкова, Бжозовского и Мак-Класки, В. Г. Боднарчука, М. А. Спивака и др. Было предложено несколько подходов и методов решения данной проблемы.

Существующие методы анализа можно разделить на графические и аналитические. Впервые алгоритм получения регулярных выражений графическим методом по графоиду автомата был дан Мак-Нотоном и Ямадой, однако он довольно громоздок, и в дальнейшем Бжозовским и Мак-Класки был предложен более простой алгоритм, использующий методы теории переходных графов сигналов. В. М. Глушков построил алгоритм анализа конечных автоматов, состоящий в последовательном применении операции расширения. В других работах проблема анализа автоматов сводится к решению ряда линейных уравнений в алгебре событий.

Ниже приводятся алгоритмы графического и аналитического методов анализа автоматов Мура. Заметим, что указанные методы можно распространить на автоматы Мили.

Рассмотрим графический метод анализа конечных автоматов. Отыскание регулярного выражения, означающего множество слов, переводящих графоид автомата из состояния q_i в q_j, сводится к нахождению ветвевого перехода из вершины q_i в q_j. В этом случае графоид автомата приводится к переходному графу, имеющему только два состояния q_i и q_j. Если в начале приведения вершина q_i является источником, а q_j — стоком, то полученный граф будет иметь только один непустой переход от q_i к q_j, и вес этого перехода как раз и является искомым регулярным выражением.

При сведении графоида автомата к переходному графу с двумя вершинами остальные вершины должны быть удалены.

Пусть необходимо устранить в графе вершину q_k с петлевым переходом t_{kk}. Причем вершины q_p и q_s соответственно являются предшествующей и последующей по отношению к вершине q_k. Очевидно, что регулярное выражение, переводящее графоид из q_p в q_s, будет иметь вид

$$R_1 = t_{pk} \cdot \{t_{kk}\}^* \cdot t_{ks},$$

а если устраним вершину q_k, то регулярное выражение запишется следующим образом:

$$R = t_{ps} V\, t_{pk} \cdot \{t_{kk}\}^* \cdot t_{ks}. \qquad (1.5)$$

Таким образом, любой графоид автомата можно свести к линейному переходному графу с двумя вершинами q_i и q_j, ветвевой переход между которыми является искомым регулярным выражением, представимым в автомате состоянием q_i при условии, что q_i — начальное состояние автомата.

При анализе графоид автомата, как правило, необходимо приводить к индексному остатку. Любой графоид автомата можно привести к индексному остатку с помощью элементарных преобразований, показанных на рис. 1.4. Однако на практике используют приведение к

индексному остатку с проверкой по правилу, которое заключается в том, что между остаточными вершинами q_i и q_j в переходном графе должна быть дуга, если в первоначальном графоиде есть, по крайней мере, один остаточный путь между q_i и q_j. Вес такой дуги равен дизъюнкции всех приращений остаточных путей от q_i к q_j. Прирост пути от вершины q_i к q_j определяется произведением приращений дуг, образующих этот путь.

Взвешенные индексные остатки графа (рис. 1.1) показаны на рис. 1.8.

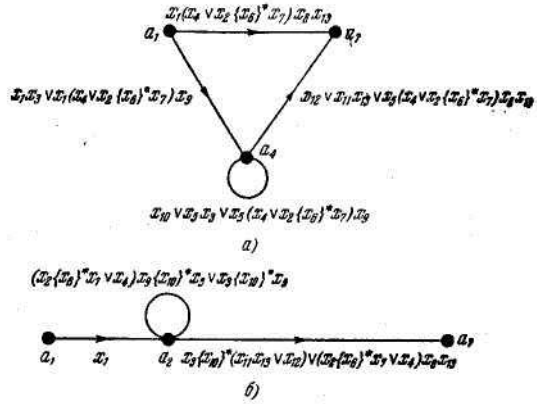


Рис. 1.8.

Остаточные пути от вершины q_4 к вершине q_7 на рис. 1.8,*а* суть x_{12}, $x_{11}x_{13}$, $x_5(x_4 \vee x_2\{x_6\}^* x_7)\, x_8x_{13}$. Следовательно, приращение новой ветви будет

$$x_{12} \vee x_{11}x_{13} \vee x_5(x_4 \vee x_2\{x_6\}^* x_7)\, x_8x_{13}.$$

Пути $x_5x_4\, x_9x_{12}$ и $x_5x_4\, x_9x_{11}x_{13}$ не являются остаточными, так как они проходят через вершину q_4. Легко видеть, что приращение ветви от вершины q_2 к вершине q_7 (рис. 1.8, *б)* имеет вид

$$x_3\{\, x_{10}\}^*(x_{11}x_{13} \vee x_{12}) \vee (x_2\{x_6\}^* x_7 \vee x_4)x_8x_{13}.$$

В случае, когда индексный остаток включает сложный контур обратной связи, устраняемые вершины могут быть исключены путем расщепления вершин. Для удаления вершины q_k все другие вершины расщепляются на источники и стоки; а переходы источник — сток вычисляются и заменяются единичными дугами. Затем расщепленные вершины соединяются вновь. Если вершина q_k должна быть устранена, то приращение новой дуги между вершинами q_p и q_s определяется по

формуле, аналогичной (1.5), т. е. состоит из дизъюнкции приращений прямого пути от q_p к q_s и приращения пути от q_p к q_s через вершину q_k. На рис. 1.9 показан пример расщепления и удаления вершин для определения регулярного выражения, представимого в автомате A состоянием q_2.

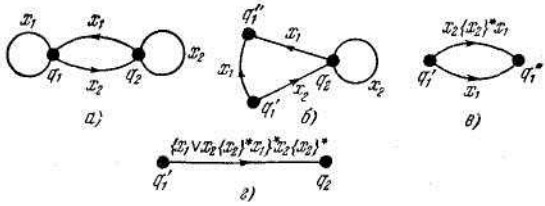


Рис. 1.9.

Сформулируем теперь алгоритм анализа математических моделей автоматов, основанный на методах теории переходных графов.

1°. *По графоиду автомата находим источник и сток. Если их нет, то начальная вершина q_i соединяется с вершиной q'_i дугой с весом е, и конечная вершина q_j соединяется с вершиной q'_j дугой с весом е. После этого q'_i берется как источник, а q'_j — как сток. Переходим к* 2°.

2°. *Все параллельные пути приводятся к форме $x_k \vee x_s$ и все последовательные пути к форме $x_k \cdot x_s$. Переходим к* 3°.

3°. *Получаем индексный остаток графоида, отмечая вершины q_i и q_j и индексные вершины. Находим приращения остаточных дуг. Переходим к* 4°.

4°. *Устраняем последовательно индексные вершины индексного остатка графоида автомата. Приращение пути от вершины q_i к вершине q_j есть регулярное выражение, представимое в автомате состоянием q_j, при условии, что q_i — начальное состояние автомата.*

Работу алгоритма покажем на примере.

Пример 1. Пусть дан автомат Мура A, графоид которого показан на рис. 1.10, *а*.

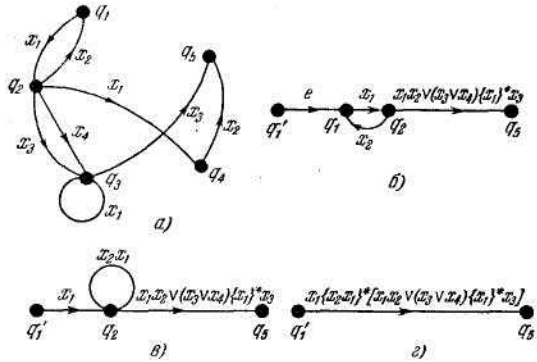


Рис. 1.10.

Найти событие, представимое состоянием q_5.

Так как графоид имеет сток, но не имеет источника, то добавляем вершину q'_1 из которой в q_1 проводим дугу, нагруженную пустым элементом *е.* Затем в соответствии с п. 2° алгоритма преобразуем графоид к виду, показанному на рис. 1.10,*б*. Индексный остаток графа приведен на рис. 1.10,*в*. Исключая индексную вершину q_2, получаем ветвевой переход (рис. 1.10,*г*), который определяет искомое регулярное выражение

$R = x_1 \bullet \{x_2 x_1\}^* \bullet [x_1 x_2 \vee (x_3 \vee x_4) \bullet \{x_1\}^* x_3]$.

Аналитический метод анализа математических моделей автоматов состоит в нахождении регулярного выражения события, представимого в автомате, путем решения системы линейных уравнений в алгебре событий.

Известно, что любой графоид автомата может быть представлен системой линейных уравнений следующего вида:

$$\begin{cases} R_1 = R_1 \vee x_1 \vee R_2 \vee x_2 \vee \ldots \vee R_n \vee x_n \vee e, \\ R_2 = R_1 \vee y_1 \vee R_2 \vee y_2 \vee \ldots \vee R_n \vee y_n, \\ R_n = R_1 \vee z_1 \vee R_2 \vee z_2 \vee \ldots \vee R_n \vee z_n, \end{cases} \qquad (1.6)$$

где R_1 ($i \in I$=l,2.....n) - событие, представимое в автомате состоянием q_i, т. е. множество путей, ведущих из состояния q_1 в q_i, где q_1 — начальное состояние; $\vee x_i$ — дизъюнкция весов дуг, ведущих из вершины q_i в q_1; $\vee y_i$ — дизъюнкция весов дуг, ведущих из вершины q_i в q_2, и т. д.; $\vee z_i$ — дизъюнкция весов дуг, ведущих из q_i в q_n.

Приведенная система сводится к решению уравнения

$R = RX \vee C$,

где X— матрица соединений автомата, $R = \{R_1, R_2, \ldots .... R_n\}$, а

$$C = \begin{cases} e, & \text{если} \quad R = \{R_1\}, \\ \emptyset, & \text{если} \quad R = \{R_2, R_3, \ldots, R_n\}. \end{cases}$$

Если графоид автомата не содержит ни одного цикла, то система уравнений (1.6) имеет единственное решение
$R = C \bullet \{X\}^*$.
В противном случае, т. е. когда существуют циклы, решение системы (1.6) неоднозначно и имеет вид
$R = (C \vee T)\{X\}^*$,
где T —любое событие. В результате решения системы (1.6) можно найти каноническое множество событий данного автомата.

Пример 2. Пусть дан автомат A, графическая интерпретация которого показана на рис. 1.11**.**

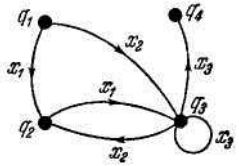


Рис. 1.11**.**
Найти событие, представимое в автомате A состоянием q_4, полагая, что q_1 — начальное состояние.
По графоиду автомата запишем следующую систему уравнений:

$$\begin{cases} R_1 = e, \\ R_2 = R_1 x_1 \vee R_3 x_2, \\ R_3 = R_1 x_2 \vee R_2 x_1 \vee R_3 x_3, \\ R_4 = R_3 x_3. \end{cases}$$

Систему уравнений решаем методом последовательного исключения неизвестных. Исключим вначале неизвестное R_2**.** Получим
$R_2 = x_1 \vee R_2 x_2$,
$R_3 = x_2 \vee (x_1 \vee R_3 x_2)\, x_1 \vee R_3 x_3$.
Затем находим
$R_4 = (x_2 \vee x_1 x_1)\{x_2 x_1 \vee x_3\}^* x_2$,
которое является регулярным выражением события, представимого в автомате A состоянием q_4.

1. 3. Минимальная форма регулярного выражения

Известно, что в результате анализа метематических моделей автоматов графическим или аналитическим методами получается некоторое множество эквивалентных форм регулярных выражений

события, представимого в автомате. Эти регулярные выражения различны по своей сложности. Поэтому возникает проблема получения минимальной формы регулярного выражения.

Регулярное выражение называется *минимальным,* если оно имеет наименьшую циклическую глубину *s*. При равной циклической глубине нескольких регулярных выражений минимальным будет выражение, обладающее меньшей длиной *l*, где *l* — количество букв, входящих в регулярное выражение.

Нетрудно убедиться, что при аналитическом и графическом методах анализа конечного автомата получается конечное число эквивалентных форм регулярных выражений, задающих работу автомата. Поэтому минимальная форма регулярного выражения одного и того же события может быть получена за конечное число раз применения алгоритма анализа к графоиду автомата.

Для того чтобы получить упрощенную (близкую к минимальной) форму регулярного выражения в процессе однократного применения алгоритма анализа, ниже предлагается ряд правил. Введем предварительно некоторые понятия.

Графоид называется *типичным,* если он имеет источник и сток, *слабо типичным,* если он имеет только источник, и *нетипичным,* если он не содержит ни источника, ни стока или имеет лишь сток.

Следует иметь в виду, что полустепень захода вершины q_i равна количеству вершин, из которых в q_i ведет хотя бы одна дуга.

Правило 1. *Для типичного графоида, имеющего только прямые пути, минимальная форма регулярного выражения определяется последовательным исключением неизвестных (вершин) в системе уравнений в любом порядке.*

Для такого автомата существует всего одно регулярное выражение.

Правило 2. *Для типичного графоида, имеющего контуры обратной связи, исключение неизвестных (вершин) начинается с неизвестного R_i, соответствующего версиине q_i, имеющей наименьшую полустепень захода. Исключение нельзя начинать с неизвестного R_i, соответствующего конечной вершине q_i. Дальнейшее исключение следует проводить по возрастающей величине полустепени захода.*

Правило 3. *Для слаботипичного графоида исключение неизвестных (вершин) следует начинать с неизвестного R_i, соответствующего вершине q_i, имеющей наименьшую полустепень захода.*

Правило 4. *Для нетипичного графоида, отмеченная вершина которого является стоком, исключение неизвестных (вершин) производится, как и для типичного графоида, по правилу* 2. *Для нетипичного графоида, отмечен мая вершина которого не является*

стоком, исключение следует производить в соответствии с правилом 3.

В общем случае может оказаться, что в несколько уравнений системы входит одинаковое число неизвестных (на графоиде это соответствует наличию нескольких вершин с одинаковыми полустепенями захода). Тогда неясно, с какого неизвестного (вершины) следует начинать исключение. Всего таких случаев может быть три, которые показаны на рис. 1.12, *а, б, в*.

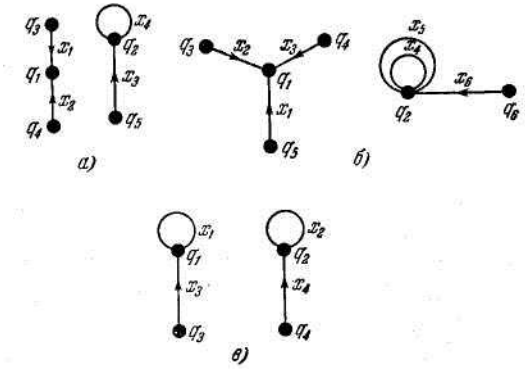


Рис. 1.12.

В зависимости от того, какая из вершин является начальной, получим три следующих варианта.

1) На рис. 1.12, *а, б,в* показаны части некоторых автоматов, ни одна вершина которых не является начальным состоянием.

Система уравнений для графоидов, показанных на рис. 1.12, *а*, имеет вид

$$\begin{cases} R_1 = R_3 x_1 \vee R_4 x_2, \\ R_2 = R_5 x_3 \vee R_2 x_4, \end{cases} \qquad (1.7)$$

откуда

$$R_2 = R_5 x_3 \{ x_4 \}^*.$$

Следовательно, исключение нужно начинать с неизвестного R_2.

Система уравнений для графоидов, показанных на рис. 1.12,*б*, запишется в форме

$$\begin{cases} R_1 = R_5 x_1 \vee R_3 x_2 \vee R_4 x_3, \\ R_2 = R_2 x_4 \vee R_2 x_5 \vee R_6 x_6, \end{cases} \qquad (1.8)$$

Откуда

$R_2 = R_6 x_6 \{ x_4 \vee x_5 \}^*$.

Поэтому исключение следует начинать с неизвестного R_2. Наконец, для графоидов, показанных на рис. 1.12, *в*, система уравнений которых запишется в виде

$$\begin{cases} R_1 = R_1x_1 \vee R_3x_3, \\ R_2 = R_2x_2 \vee R_4x_4, \end{cases} \quad (1.9)$$

исключение неизвестных можно проводить в любом порядке.

2) Предположим, что q_1 — начальное состояние некоторых частей автоматов, показанных на рис. 1.12, *а, б, в*. Тогда в системы уравнений (1.7), (1.8) и (1.9) для R_1 добавляется элемент e. Это усложняет уравнение R_1. Поэтому исключение неизвестных (вершин на графоиде) во всех случаях следует начинать с R_2.

3) Допустим теперь, что начальным состоянием автоматов служит вершина q_2 (рис. 1.12, *а, б, в*).

Тогда системы уравнений, описывающих графоиды, показанные на рис. 1.12, *а* и *б*, принимают вид

$$\begin{cases} R_1 = R_3x_1 \vee R_4x_2, \\ R_2 = R_5x_3 \vee R_2x_4 \vee e \end{cases}$$

и

$$\begin{cases} R_1 = R_5x_1 \vee R_3x_2 \vee R_4x_3, \\ R_2 = R_2x_4 \vee R_2x_5 \vee R_6x_6 \vee e, \end{cases}$$

а их решениями будут соответственно

$$R_2 = (R_5x_3 \vee e)\{x_4\}^*,$$

$$R_2 = (R_6x_6 \vee e)\{x_4 \vee x_5\}^*,$$

с которых следует начинать исключение неизвестного. Для графоида, показанного на рис. 1.12, *в*, исключение следует начинать с неизвестного R_1, так как уравнение (1.9) для R_2 усложняется добавлением буквы e.

Пример 3. Пусть дан автомат A, графоид которого показан на рис. 1.13.

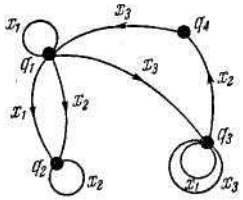


Рис. 1.13.

Необходимо найти минимальное регулярное выражение события, представимого в автомате состоянием q_4.

Составим систему уравнений

$$\begin{cases} R_1 = R_1x_1 \vee R_2x_2 \vee R_4x_3 \vee e, \\ R_2 = R_1x_1 \vee R_2x_2, \\ R_3 = R_2x_3 \vee R_3x_3 \vee R_3x_1, \\ R_4 = R_3x_2. \end{cases}$$

Исключая последовательно неизвестные R_2, R_1, R_3 определим R_4.

$$R_2 = R_1x_1\{x_2\}^*,$$

$$R_1 = R_1x_1 \vee R_1x_1\{x_2\}^*x_2 \vee R_4x_3 \vee e,$$

$$R_1 = (R_4x_3 \vee e)\{x_1 \vee x_1\{x_2\}^*x_2\}^*,$$

$$R_3 = (R_4x_3 \vee e)\{x_1 \vee x_1\{x_2\}^*x_2\}^*x_1\{x_2\}^*x_3 \vee R_3(x_3 \vee x_1),$$

$$R_4 = R_4x_3\{x_1 \vee x_1\{x_2\}^*x_2\}^*x_1\{x_2\}^*x_3\{x_3 \vee x_1\}^*x_2 \vee \vee \{x_1 \vee x_1\{x_2\}^*x_2\}^*x_1\{x_2\}^*x_3\{x_3 \vee x_1\}^*x_2,$$

$$R_4 = \{x_1 \vee x_1\{x_2\}^*x_2\}^*x_1\{x_2\}^*\{x_3 \vee x_1\}^*x_2 \cdot \{x_3\{x_1 \vee x_1\{x_2\}^*\}^* \cdot \cdot x_1\{x_2\}^*x_3\{x_3 \vee x_1\}\}^*x_2.$$

Искомое регулярное выражение имеет длину $l = 19$ и циклическую глубину $s = 3$.

Найдем минимальное регулярное выражение того же события. Используя правило 4, исключим вначале неизвестное R_4, а затем R_2. Далее по правилу 2, исключая R_3 и, наконец, R_1, находим R_4:

$$R_2 = R_1x_1\{x_2\}^*,$$

$$R_3 = R_1x_1\{x_2\}^* \vee R_3x_3 \vee R_3x_1,$$

$$R_3 = R_1x_1\{x_2\}^*\{x_3 \vee x_1\}^*,$$

$$R_1 = R_1x_1 \vee R_1x_1\{x_2\}^* \vee R_1x_1\{x_2\}^*\{x_3 \vee x_1\}^*x_2x_3 \vee e,$$

$$R_1 = \{x_1 \vee x_1\{x_2\}^* \vee x_1\{x_2\}^*\{x_3 \vee x_1\}^*x_2x_3\}^*,$$

$$R_4 = \{x_1 \vee x_1\{x_2\}^* \vee x_1\{x_2\}^*\{x_3 \vee x_1\}^*x_2x_3\}^*x_1\{x_2\}^*\{x_3 \vee x_1\}^*x_2.$$

Как видно, данное регулярное выражение имеет $l = 14$ и $s = 2$.

В заключение отметим, что в общем случае при анализе математических моделей автоматов целесообразно комбинировать графический и аналитический методы анализа, применяя вначале к графоиду автомата тождественные преобразования, а затем составляя и решая систему уравнений.

1.4. Задание регулярных выражений в форме графов

Любое регулярное выражение можно представить в виде графа. Графы элементарных регулярных выражений, соответствующих дизъюнкции $x_i \vee x_j$, умножению $x_i \cdot x_j$ и итерации $\{ x_i \}^*$, показаны на рис. 1.14,*а,б,в.*

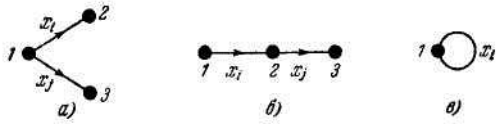


Рис. 1.14.

Вершинам графа обычно приписываются номера из множества 1, 2, 3,... В некоторых случаях они вовсе опускаются. В каждом графе регулярного выражения фиксируются вершина, которая является началом (обычно номер один), и вершины (одна или несколько), которые служат концом.

В следующем подразделе излагается алгоритм синтеза математических моделей автоматов по графам регулярных выражений (Точнее говоря, по нагруженным графам регулярных выражений, т. е. по графоидам. Поскольку этот термин используется для задания автомата, нагруженные графы регулярных выражений будем называть просто графами). Однако в графах регулярных выражений возможны случаи образования ложных путей, которые соответствуют входным словам, не принадлежащим исходным регулярным выражениям. Поэтому ниже будет доказано утверждение о полноте системы правил, устраняющих ложные пути в графах регулярных выражений с помощью введения *пустых стрелок.*

Используя графы элементарных регулярных выражений, можно индуктивно построить граф сколь угодно сложного регулярного выражения. Например, граф регулярного выражения

$R = x_1 x_3 \cdot \{x_2 x_1\}^* \vee x_4 \{x_1\}^* x_2 \vee \{x_3\}^*$ имеет вид, показанный на рис. 1.15.

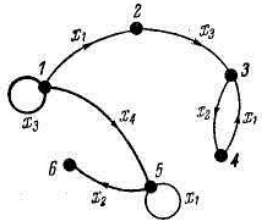


Рис. 1.15.

Начальной вершиной графа является вершина 1, а конечными — вершины 3, 6 и 1.

Каждому пути в графе регулярного выражения от начала к любому из концов должна соответствовать последовательность входных букв, принадлежащая данному регулярному выражению и, наоборот, каждая последовательность, задаваемая регулярным выражением, определяет некоторый путь в графе регулярного выражения. Необходимо отметить, что в графах, построенных по регулярным выражениям определенного вида, могут возникать ложные последовательности, не задаваемые регулярным выражением. Так, путь на графе рис. 1.15, выделенный жирными дугами, соответствует группе $x_3x_4x_2$, которое не принадлежит регулярному выражению R.

Возникновение ложных последовательностей можно избежать введением в графе пустых стрелок, обозначающих мгновенный переход из одной вершины графа в другую.

О. П. Кузнецовым рассмотрены случаи возникновения и устранения ложных последовательностей в графах регулярных выражений. Например, исправленный граф регулярного выражения R показан на рис. 1.16.

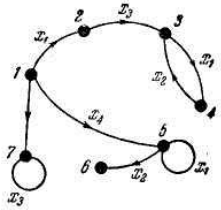


Рис. 1.6.

Приведем систему правил, определяющих типы регулярных выражений, графы которых должны содержать пустые стрелки, и докажем утверждение о ее полноте.

Правило 1. Пустые стрелки на графе регулярного выражения S вводятся в случае произведения двух или более итераций

$$S = \prod_{i \in I} \{R_i\}^*,$$

где $i \in I = \{1,2,..., n\}$, а R_i — произвольное регулярное выражение. Геометрическая интерпретация правила 1 при $I = \{1, 2, 3\}$ показана на рис. 1.17.

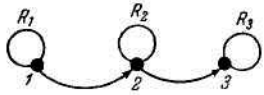


Рис. 1.17.

Правило 2. Пустые стрелки на графе регулярного выражения *S*, которое начинается и заканчивается итерационными скобками, вводятся в случаях

а) $S = \{\{P\}^* \cdot R\}^*$;

б) $S = \{R \cdot \{N\}^*\}^*$;

в) $S = \{\{P\}^* \cdot R \cdot \{N\}^*\}^*$,

где *P, R, N*— любые регулярные выражения.

Применение правила 2 показано на рис. 1.18, *а, б, в*.

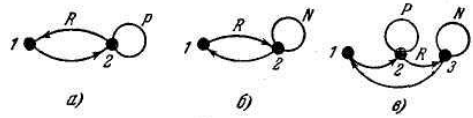


Рис. 1.18.

Правило 3. Пустые стрелки на графе регулярного выражения *S* вводятся в случае дизъюнкции, если хотя бы один из дизъюнктивных членов начинается с итерации

$S = \{R\}^* \cdot Q \vee \{P\}^* \vee \ \ldots \ \vee Q$,

где *Q* — регулярное выражение, не содержащее итерационных скобок. Правилу 3 соответствует граф регулярного выражения, показанный на рис. 1.19.

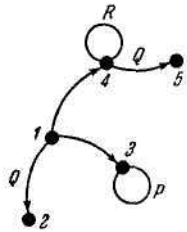


Рис. 1.19.

Правило 4. Пустые стрелки на графе регулярного выражения *S* вводятся при умножении слева на дизъюнкцию, если хотя бы один из дизъюнктивных членов заканчивается итерацией

$S = (Q \bullet \{P\}^* \vee \{R\}^* \vee \ \ldots \ \vee Q) \bullet N$.

Граф регулярного выражения *S* показан на рис. 1.20.

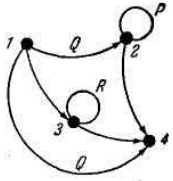


Рис. 1.20.

Следует отметить, что в частном случае регулярные выражения *P, R, N, Q* могут состоять из одного элемента.

Справедливо следующее **утверждение.**

Система правил 1—4 *введения пустых стрелок для исключения ложных путей в графах регулярных выражений является полной.*

Доказательство. Любое регулярное выражение в алгебре событий образуется в результате применения конечного числа операций дизъюнкции ($\vee$), умножения (•) и итерации ($\{\}^*$). Прежде, чем рассматривать различные сочетания из этих трех операций, заметим, что необходимость в использовании пустых стрелок появляется лишь тогда, когда регулярное выражение содержит итерацию. Одна лишь операция итерации как при однократном, так и при многократном применении не приводит к появлению пустых стрелок в графе регулярного выражения. Поэтому можно утверждать, что при построении графа регулярного выражения необходимость в использовании пустых стрелок может возникнуть только в случае сочетания следующих операций (•, $\{\}^*$), ($\vee$, $\{\}^*$), (•, $\vee$, $\{\}^*$). Однако не всегда наличие итерации в регулярном выражении приводит к появлению пустой стрелки. Определим, в каких случаях необходимо использование пустых стрелок в графах регулярных выражений.

Рассмотрим регулярные выражения, полученные применением всех трех операций. Обозначим регулярные выражения в базисе (•, $\vee$) через *P*, (•, {}*) — через *Q*, ($\vee$, {}*) — через *R* и в базисе (•, $\vee$, {}*) — через *V*.

Нетрудно видеть, что итерация над *P* не требует использования пустой стрелки. Применение итерации к *R, Q, V* приводит к появлению пустых стрелок. Эти случаи описываются правилом 2.

Рассматривая попарно дизъюнкции всех регулярных выражений, видим, что $P \vee P$ не требует использования пустой стрелки. Все остальные случаи описываются правилом 3.

Осталось разобрать все возможные попарные произведения из *P, R, Q* и *V*. Ясно, что $P \bullet P$, $P \bullet Q$, $P \bullet R$, $P \bullet V$ и $Q \bullet P$ не требуют использования пустых стрелок. Введение пустых стрелок в графах $Q \bullet Q$, $Q \bullet R$ и $Q \bullet V$ описывается правилом 1. Произведения $R \bullet P$, $V \bullet P$ описываются правилом 4, а $R \bullet Q$,$R \bullet R$, $R \bullet V$, $V \bullet Q$, $V \bullet R$ и $V \bullet V$ описываются правилами 1 и 4.

Таким образом, нами рассмотрены все возможные виды регулярных выражений и показано, что сочетания операций (•, $\vee$, {}*), которые приводят к появлению пустых стрелок в графах регулярных выражений, полностью описываются правилами 1—4. Этим наше утверждение доказано.

1.5. Алгоритм синтеза автоматов

Известно, что методы теории переходных графов сигналов, разработанные для анализа электрических и электронных схем, могут успешно применяться для анализа математических моделей конечных автоматов. С точки зрения практического применения наибольшее значение имеют алгоритмы синтеза математических моделей автоматов, среди которых широкое распространение получили методы, предложенные В. М. Глушковым. Однако алгоритмы синтеза математических моделей автоматов предполагают использование правил подчинения мест в регулярном выражении, которые, по сути дела, сводятся к перебору всех возможных слов, входящих в данное регулярное выражение. Применение правил подчинения мест не всегда удобно и затрудняет использование алгоритма.

Геометрической интерпретацией основного аглоритма синтеза математических моделей автоматов В. М. Глушкова является метод построения таблицы переходов (графоида автомата) по графам регулярных выражений, предложенный О. П. Кузнецовым. Ниже приводится алгоритм синтеза математических моделей автоматов по графам регулярных выражений, который приводит к такому же результату, как и усовершенствованный алгоритм синтеза В. М. Глушкова. Предлагаемый алгоритм обеспечивает построение автоматов с частично минимизированным числом состояний и обладает меньшей трудоемкостью по сравнению с предложенным О. П. Кузнецовым, так как упрощается индексация вершин на графе регулярного выражения. В результате применения указанного алгоритма отождествляются подобные и квазиподобные места в регулярных выражениях, хотя, вообще говоря, не требуется разметки мест и использования правил их подчинения.

Прежде, чем строить граф регулярного выражения, необходимо каждый из членов конечного множества регулярных выражений, представимых в автомате разными выходными сигналами y_i, преобразовать следующим образом. Используя закон коммутативности дизъюнкции, сгруппируем дизъюнктивные члены с одинаковыми конечными последовательностями входных букв и вынесем общие множители вправо за скобки (при вынесении за скобки общих множителей нельзя допускать, чтобы дизъюнктивный член в скобках превращался в пустое слово). На графе регулярного выражения это приводит к объединению вершин i и j таких, что

а) любая последовательность стрелок, выходящая из i, имеет аналогичную последовательность, выходящую из j, и наоборот;

б) конечные вершины этих последовательностей должны быть отмечены одинаковыми выходными индексами.

Из определения квазиподобных мест следует, что объединение вершин *i* и *j* на графе регулярного выражения соответствует частичному отождествлению квазиподобных мест в регулярном выражении. Ясно, что в результате такого преобразования регулярного выражения получаем эквивалентную форму того же события.

В процессе построения графа регулярного выражения все начальные вершины графов, представляющих каждое из регулярных выражений, объединяются в одну вершину, которой присваивается индекс 1. Далее, всем остальным вершинам графа регулярного выражения присваиваем по одному неповторяющемуся индексу из множества 2,3,... Подобная индексация вершин графа позволяет присвоить один индекс *n* подобным и один индекс *m* квазиподобным местам в регулярном выражении.

Индекс вершины графа, из которой выходит пустая стрелка, приписывается слева к индексу вершины, в которую эта пустая стрелка входит. Такая индексация вершин, связанных между собой пустыми стрелками, определяет правильный порядок следования по графу от начальной вершины к конечной. Это обеспечивает построение автомата, представляющего множеством выходных элементов заданное регулярное выражение.

Таким образом, в результате построения графа регулярного выражения с учетом правил 1—4 и указанного порядка присвоения индексов вершинам происходит отождествление подобных и квазиподобных мест в исходном регулярном выражении.

Сформулируем теперь алгоритм абстрактного синтеза автоматов.

1°. *Преобразовываем каждое регулярное выражение, вынося общие множители вправо за скобку, и объединяя их знаками дизъюнкции. Получаем регулярное выражение R.*

2°. *Используя правила* 1—4*, строим граф регулярного выражения R.*

3°. *Присваиваем вершинам графа индексы из множества* $I = \{1,2,3,...\}$.

4°. *Строим таблицу переходов автомата. Строки таблицы соответствуют различным буквам* x_i *входного алфавита, содержащимся в R. Первый столбец таблицы обозначим символом* «1» *начальной вершины графа и с нее начинаем построение таблицы переходов автомата. В клетку таблицы, стоящую на пересечении* q_j *столбца и* x_i *строки, записывается дизъюнкция индексов тех вершин графа, в которые входят стрелки с буквами* x_i*, выходящие из любой вершины, индексы которой включены в множество индексов состояния* q_j*. В начале построения таблицы в состояние* q_j *входит*

один индекс начальной вершины. Если таких стрелок нет, в клетку вписывается «—» — символ пустого состояния автомата. После запалнения клеток столбца таблицы содержимое каждой клетки, если оно полностью не совпадает с содержимым состояний, отмечающих столбцы, выписывается как новое отмечающее состояние. Процесс построения таблицы переходов считается законченным, если все содержимое каждой клетки таблицы выписано как отмечающее состояние. Если вершина графа имеет несколько индексов, то в клетку таблицы переходов записывается только крайний справа индекс.

5°. *Состояние q_i, в множество индексов которого входит индекс конечной вершины графа, отмечается выходным сигналом y_i, соответствующим данной конечной вершине графа. Если в множество индексов входят индексы нескольких конечных вершин, то такое состояние отмечается дизъюнкцией выходных сигналов.*

6°. *Производя перекодировку состояний и выходных букв, получаем отмеченную таблицу переходов автомата.*

Рассмотрим работу алгоритма на примерах.

Пример 4. Пусть задано событие, регулярное выражение которого имеет вид

$R = \{ x_1 \vee x_2\}^* \bullet (x_1\{ x_1\}^* \bullet \{x_3\}^* x_2 \vee x_1 \{x_3\}^* x_2 \vee x_3 \{x_3\}^* x_2)$.

Построить автомат Мура *A,* представляющий это событие выходным элементом *y*.

Вынося общие множители за скобки, запишем выражение *R* в форме

$R = \{ x_1 \vee x_2\}^* \bullet (x_1\{ x_1\}^* \vee x_1 \vee x_3) \bullet \{x\}_3 x_2$.

Учитывая правило 4, строим граф регулярного выражения *R,* который показан на рис. 1.21.

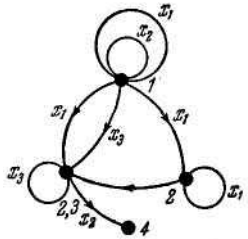


Рис. 1.21.

Приписываем номера вершинам графа. Начальная вершина графа помечена номером 1, а конечная — номером 4.

По графу регулярного выражения строим отмеченную таблицу переходов, которая имеет вид, показанный на табл. 1.1.

Таблица 1.1

Y	–	–	–	y	y
I / X	1	1∨2∨3	3	1∨4	4
x_1	1∨2∨3	1∨2∨3	–	1∨2∨3	–
x_2	1	1∨4	4	1	–
x_3	3	3	3	3	–

Производя перекодировку состояний, окончательно получаем отмеченную таблицу переходов (табл. 1.2) автомата Мура *A,* графоид которого показан на рис. 1.22.

Т а б л и ц а 1.2

Y	–	–	–	y	y
Q / X	q_1	q_2	q_3	q_4	q_5
x_1	q_2	q_2	–	q_2	–
x_2	q_1	q_4	q_5	q_1	–
x_3	q_3	q_3	q_3	q_3	–

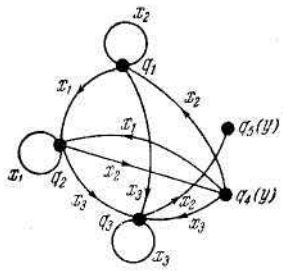


Рис. 1.22.

Пример 5. Требуется синтезировать автомат, моделирующий выработку условного рефлекса с забыванием. Входной алфавит

автомата обозначим через $X = \{x_1, x_2, x_3\}$, где x_1 — наличие условного раздражителя и отсутствие безусловного, x_2 — наличие безусловного раздражителя и отсутствие условного, x_3 — одновременная подача условного и безусловного раздражителей. Входной алфавит автомата состоит из двух букв $Y = \{ y_1, y_2\}$. Автомат вырабатывает условный рефлекс после некоторого этапа обучения и реагирует выходной буквой y_2 на действие условного раздражителя x_1 так же, как и на действие безусловного x_2 или совместное действие обоих раздражителей x_3. В остальных случаях на выходе появляется буква y_1. Этап обучения заключается в одновременном действии условного и безусловного раздражителей. Таких совпадений за время обучения автомата должно быть не меньше n. Если в процессе обучения между двумя последовательными совпадениями раздражителей произошло более чем k несовпадений, то процесс обучения нарушается и его нужно начинать заново. Если после выработки автоматом условного рефлекса произойдет более чем m последовательных несовпадений действия условного и безусловного раздражителей, то рефлекс забывается.

Считаем, что условный рефлекс с забыванием вырабатывается при следующих параметрах: $k \leq 1$, $n \geq 2$, $m > 2$.

Переходя от описательной формы работы автомата к заданию алгоритма на языке регулярных выражений, получим

$$R = \{x_1 \vee x_2 \vee x_3\}^* [(x_2 \vee x_3) \vee x_3 ((x_1 \vee x_2) x_3 \vee x_3) \cdot$$
$$\cdot \{x_3\}^* \{ (x_1 \vee x_2) \cdot x_3\}^* \cdot ((x_1 \vee x_2) \vee (x_1 \vee x_2) \cdot (x_1 \vee x_2))].$$

Граф регулярного выражения R показан на рис. 1.23.

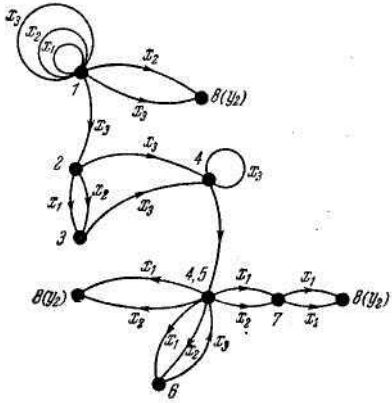


Рис. 1.23.

По графу строим отмеченную таблицу переходов

(табл. 1.3). Состояния $(1 \vee 2 \vee 8 \vee 4)$ и $(1 \vee 2 \vee 8 \vee 5)$

Таблица 1.3

Y	y_1	y_2	y_2	y_1
I / X	1	1∨8	1∨2∨8	1∨3
x_1	1	1	1∨3	1
x_2	1∨8	1∨8	1∨8∨3	1∨8
x_3	1∨2∨8	1∨2∨8	1∨2∨8∨4	1∨2∨8∨4

Y	y_1	y_2	y_2	y_2
I / X	1∨8∨3	1∨2∨8∨4	1∨2∨8∨6∨7	1∨2∨8∨5
x_1	1	1∨3∨8∨6∨7	1∨8	1∨3∨6∨8∨7
x_2	1∨8	1∨8∨3∨6∨7	1∨8	1∨8∨3∨6∨7
x_3	1∨2∨8∨4	1∨2∨8∨4	1∨2∨8∨5	1∨2∨8∨4

отождествляем, так как им соответствуют одинаковые столбцы в табл. 1.3. Производим перекодировку состояний следующим образом:

$$(1) \to q_1, \quad (1 \vee 8) \to q_2, \quad (1 \vee 2 \vee 8) \to q_3, \quad (1 \vee 3) \to q_4,$$
$$(1 \vee 8 \vee 3) \to q_5, \quad (1 \vee 2 \vee 8 \vee 4) \to q_6,$$
$$(1 \vee 3 \vee 8 \vee 6 \vee 7) \to q_7, \quad (1 \vee 2 \vee 8 \vee 5) \to q_6.$$

Окончательный вариант отмеченной таблицы переходов автомата имеет вид табл. 1.4, а графоид автомата показан на рис. 1.4.

Таблица 1.4

Y	y_1	y_2	y_2	y_1	y_2	y_2	y_2
Q / X	q_1	q_2	q_3	q_4	q_5	q_6	q_7
x_1	q_1	q_1	q_4	q_1	q_1	q_7	q_2
x_2	q_2	q_2	q_5	q_2	q_2	q_7	q_2
x_3	q_3	q_3	q_6	q_6	q_6	q_6	q_6

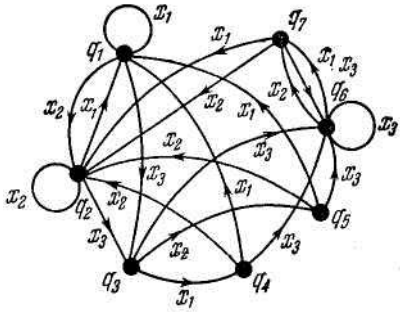


Рис. 1.24.

Рассмотрим теперь алгебраический метод синтеза абстрактных автоматов, состоящий в нахождении по регулярному выражению системы уравнений, которая определяет графоид автомата, представляющий заданное регулярное выражение. М. А. Спивак ввел понятие *базиса* в языке регулярных выражений. Конечную систему регулярных выражений $R_1, ..., R_m$ в алфавите $X = \{x_1, x_2,..., x_n\}$ назовем базисом, если имеет место равенство

$$R_i = R_1x_1 \vee R_2x_2 \vee \ ... \ \vee R_nx_n \vee C, \qquad (1.10)$$

где

$$C = \begin{cases} \emptyset, & \text{если} \quad e \notin R_i, \\ e, & \text{если} \quad e \in R_i, \end{cases}$$

а $i \in I = \{1,2,..., m\}$. В результате построения уравнений базиса получаем систему уравнений, определяющую графоид автомата.

Рассмотрим примеры построения систем уравнений для простейших регулярных выражений.

1) Пусть $R = x_1$. Используя (1.10), запишем систему уравнений, которая является базисом

$$\begin{cases} R_2 = R_1x_1, \\ R_1 = e. \end{cases}$$

Геометрическая интерпретация системы уравнений показана на рис. 1.25.

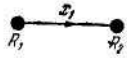


Рис. 1.25.

2) Пусть $R = x_1 \vee x_2$. Система уравнений имеет вид

$$\begin{cases} R_2 = R_1(x_1 \vee x_2), \\ R_1 = e. \end{cases}$$

3) Пусть $R = x_1 \cdot x_2$. Тогда система уравнений запишется в форме

$$\begin{cases} R_3 = R_1 x_1 x_2 = R_2 x_2, \\ R_2 = R_1 x_1, \\ R_1 = e. \end{cases}$$

Геометрическая интерпретация системы показана на рис. 1.26.

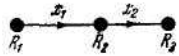


Рис. 1.26.

4) Пусть $R = \{S\}^*$. Система уравнений сводится к одному уравнению:

$R_1 = R_1 S \vee e,$

где S может принимать, например, следующие значения:

а) $S = x_1;$

б) $S = x_1 \vee x_2;$

в) $S = x_1 \bullet x_2.$

Тогда системы уравнений имеют соответственно вид

а) $R_1 = R_1 \cdot x_1 \vee e;$

б) $R_1 = R_1(x_1 \vee x_2) \vee e;$

в) $$\begin{cases} R_2 = R_2 x_1 x_2 \vee e = R_1 \cdot x_2 \vee e, \\ R_1 = R_2 x_1, \end{cases}$$

а их геометрическая интерпретация показана на рис. 1.27, *а, б, в.*

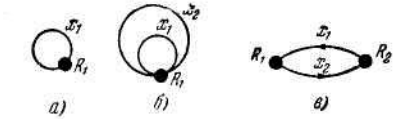


Рис. 1.27.

Заметим, что при синтезе автомата по регулярному выражению, граф которого содержит ложные последовательности входных слов, возникают некоторые трудности.

Рассмотрим, например, регулярное выражение $R = \{P\}^* \bullet \{S\}^*$, граф которого (см. 1.4) имеет ложный путь. Запишем систему уравнений, определяющую базис

$$\begin{cases} R_2 = R_1 \vee R_2 \cdot S, \\ R_1 = R_1 \cdot P \vee e, \end{cases}$$

интерпретация которой показана на рис. 1.28.

Рис. 1.28.

Из него видно, что мы получили граф регулярного выражения *R*, а не графоид автомата, представляющий это регулярное выражение. Для того чтобы базис соответствовал графоиду автомата, необходимо в уравнении R_2 член R_1 умножить на *S*. Тогда система примет вид

$$\begin{cases} R_2 = R_1 \cdot S \vee R_2 \cdot S, \\ R_1 = R_1 \cdot P \vee e. \end{cases} \qquad (1.11)$$

Аналогичные трудности возникают в других случаях, за исключением умножения слева на дизъюнкцию, один из членов которой заканчивается итерацией.

Для устранения указанных затруднений можно воспользоваться правилом развертывания итерации

$$\{S\}^* = S \cdot \{S\}^* \vee e$$

или тождеством

$$\{S\}^* = \{S \vee e\}^*,$$

которые позволяют находить базис, определяющий графоид автомата. Однако число состояний такого автомата слишком велико, и в дальнейшем обязательно требуется провести этап минимизации автомата.

Пример 6. Пусть заданы регулярные выражения

$P = x_1 \cdot \{x_3x_2\}^* \cdot (x_4 \vee x_3x_5)$

и

$S = \{ x_1 \cdot x_2 \vee x_1 \cdot x_3 \cdot x_1\}^* \cdot \{x_1 \{x_2x_3\}^*\}^*$.

Построить автоматы A_1 и A_2, представляющие выражения *P* и *S* некоторым множеством своих состояний. Запишем выражение *P* в форме многочлена

$P=x_1 \{x_3x_2\}^* x_4 \vee x_1 \{x_3x_2\}^* \cdot x_3 \cdot x_5$.

Используя построение базиса для простейших регулярных выражений и последовательно разворачивая выражение *P*, получим систему уравнений

$$\begin{cases} R_4 = R_2 \cdot x_4 \vee R_3 \cdot x_5, \\ R_2 = R_1 \cdot x_1 \vee R_2 x_3 x_2 = R_1 \cdot x_1 \vee R_3 \cdot x_2, \\ R_1 = e, \\ R_3 = R_2 \cdot x_3, \end{cases}$$

определяющую автомат A_1, графоид которого показан на рис. 1.29.

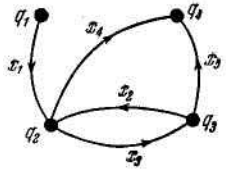


Рис. 1.29.

Автомат A_1 представляет выражение P состоянием q_4, в которое ведут все пути, оканчивающиеся элементами x_4 или x_5. На основании (1.11) для выражения S запишем

$$\begin{cases} R_2 = R_1 x_1 \{x_2 x_3\}^* \vee R_2 x_1 \{x_2 x_3\}^*, \\ R_1 = R_1 \cdot (x_1 x_2 \vee x_1 x_3 x_1) \vee e. \end{cases}$$

Последовательно находим одночлены, входящие в выражения R_1 и R_2:

$$R_1 = R_1 x_1 x_2 \vee R_1 x_1 x_3 x_1 \vee e = R_3 x_2 \vee R_4 x_1 \vee e,$$
$$R_3 = R_1 x_1,$$
$$R_4 = R_3 x_3,$$
$$R_2 = R_1 x_1 \vee R_2 x_1 \vee R_2 x_2 x_3 = R_1 x_1 \vee R_2 x_1 \vee R_5 x_3,$$
$$R_5 = R_2 x_3.$$

Перепишем систему уравнений в виде

$$\begin{cases} R_1 = R_3 x_2 \vee R_4 x_1 \vee e, \\ R_2 = R_1 x_1 \vee R_2 x_1 \vee R_5 x_2, \\ R_3 = R_1 x_1, \\ R_4 = R_3 x_3, \\ R_5 = R_2 x_3, \end{cases}$$

которая определяет автомат A_2, представляющий событие S состояниями q_1 и q_2. Графоид автомата A_2 показан на рис. 1.30.

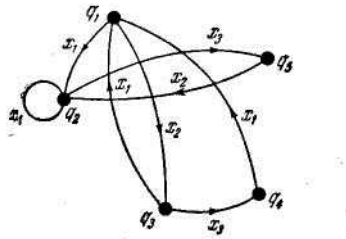


Рис. 1.30.

В заключение отметим, что если регулярное выражение минимально, то при синтезе получаем автомат, число состояний которого меньше или, по крайней мере, равно числу состояний автомата, построенного по любому другому регулярному выражению.

1.6. Этапы синтеза автоматов

В теории автоматов процесс синтеза автомата принято подразделять на отдельные этапы.

На первом, или предварительном, этапе синтеза формулируются (часто словесно) условия работы автомата, т. е. определяются условия его взаимодействия с другими устройствами или какими-либо объектами; выявляются необходимые входы и выходы автомата и намечается общий закон появления выходных сигналов в зависимости от воздействия на входы автомата. При синтезе достаточно сложного автомата часто его разбивают на отдельные блоки. В этом случае **первый этап синтеза** иногда называют *этапом блочного синтеза автомата*

В Институте кибернетики АН УССР под руководством акад. В. М. Глушкова разработана формализованная методика блочного синтеза ЭВМ , позволяющая получать более экономичные решения, чем при интуитивном методе.

М. А. Гаврилов разработал блочный метод синтеза, позволяющий синтезировать один сложный автомат из простых блоков, выполняющих отдельные его функции. Такой метод дает возможность упростить формулировку условий работы (на языке таблиц переходов) и процесс построения автомата.

На этом этапе условия работы автомата принято задавать в виде процессов управления на естественном или одном из формализованных языков, в частности на широко распространенном языке сетей Петри.

На **втором этапе синтеза** выявляются закон функционирования автомата (его функции переходов и функции выходов) и формальное описание автомата одним из принятых для этого способов. Этот этап синтеза принято называть *этапом абстрактного синтеза или синтезом абстрактного автомата.*

На этапе абстрактного синтеза автомата не интересуются теми физическими элементами, из которых он должен состоять. Нет нужды рассматривать, какие значения могут принимать входы, выходы и элементы памяти автомата. Важно знать лишь число возможных различных состояний входов и выходов автомата, число различных его внутренних состояний, а также законы изменения внутренних состояний автомата и выработки им тех или иных состояний выходов при поступлении на его входы той или иной последовательности состояний входа.

Исследование абстрактного синтеза автоматов было начато С. К-Клини, которым был предложен так называемый *язык регулярных событий.* В дальнейшем абстрактный синтез автомата, основанный на использовании этого языка, был усовершенствован В. М. Глушковым, который разработал алгоритм абстрактного синтеза, годный для программирования на ЭВМ .

Другим языком, на котором базируется абстрактный синтез автомата, является язык исчисления предикатов, исследованный Б. А. Трахтенбротом.

Результатом второго этапа синтеза является задание автомата одним из стандартных способов, к которым относятся описанные выше таблицы и матрицы переходов, а также таблицы включений и диаграммы переходов.

В некоторых случаях удобно задавать автомат временными функциями, а МА —логическими схемами алгоритмов.

На втором этапе синтеза выявляется емкость памяти автомата. При этом, несмотря на стремление получить уже при абстрактном синтезе автомат с минимальным числом внутренних состояний, в ряде случаев получаем автоматы, у которых число внутренних состояний заметно отличается от минимального. В связи с этим на следующем**, третьем этапе** производится *минимизация числа внутренних состояний автомата.*

На **четвертом этапе синтеза** производится *кодирование* или, как говорят, *размещение внутренних состояний.* Этот этап находится на грани абстрактного и структурного рассмотрения автомата. После кодирования внутренних состояний автомата, состояний входа и выхода выписываются канонические уравнения.

На **пятом этапе синтеза** завершается выбор структуры автомата построением так называемой *функциональной схемы* комбинационной части автомата (однотактной схемы) или, как говорят, логического преобразователя. При этом синтез автомата с памятью (многотактной схемы) сводится к синтезу автомата без памяти (однотактной схемы) с помощью понятия *однотактного эквивалента*, когда s обратных связей *n, k*-по-юсника обрывается и производится синтез преобразователя с $n+s$ входами и $k+s$ выходами.
В настоящее время этот этап синтеза в теории автоматов наиболее развит.
Шестой этап синтеза включает электрический или другой расчет элементов схемы и моделирование автомата с целью проверки его работоспособности.
На **седьмом этапе** осуществляется размещение деталей на платах, составление монтажных схем и технической документации. При использовании интегральных схем и особенно схем с большой и сверхбольшой степенью интеграции (БИС и СБИС) на этом этапе осуществляется размещение элементов функциональной схемы по корпусам, а затем размещение корпуса на стандартных платах, часто называемых *типовыми элементами замены* (ТЭЗ) —типовые элементы замены по блокам и блоки по шкафам.
Разделение процесса синтеза на этапы, с одной стороны, облегчает процесс построения автомата, а с другой стороны, может привести к усложнению его структуры. Так, рассматривая на отдельных этапах процессы минимизации числа внутренних состояний, кодирование внутренних состояний и синтез структуры логического преобразователя, можно прийти к менее оптимальному решению по сравнению с тем случаем, когда эти три этапа были бы совмещены. Этот недостаток пытаются уменьшить тем, что на каждом этапе синтеза стараются учесть его влияние на следующий этап.
Разделение процесса синтеза автомата на такие семь этапов является достаточно условным, хотя это и позволяет облегчить решение всей задачи синтеза автомата. Часто некоторые этапы синтеза не разделяются между собой, а иногда просто опускаются.
В ряде работ процесс синтеза излагается несколько иначе. При этом в ряде работ изложена теория проектирования автоматов, основанная на семантике выполняемых преобразований.
Следует заметить, что наряду с изложенным выше делением процесса синтеза весь процесс проектирования сложного управляющего автомата подразделяют на более **крупные этапы: системное, программно-логическое, техническое и технологическое проектирование.**

На этапе *системного проектирования,* включающего в себя блочный синтез автомата, определяют общую структуру автомата, функции и назначение его отдельных блоков. В результате выполнения этапа системного проектирования должны быть получены состав автомата, структура соединений и общее *алгоритмическое описание* каждого из блоков.

На этапе *программно-логического проектирования* разрабатывают функциональные схемы каждого из блоков. При этом, если используется аппаратурная реализация автомата, то на этапе *логического проектирования* производят абстрактный синтез автомата, минимизацию числа его внутренних состояний и их кодирование, синтез функциональной схемы автомата, разрабатывают функциональные схемы каждого из блоков автомата. При этом по общему алгоритмическому описанию выбирают принцип построения каждого из блоков и строят функциональные схемы в заданном базисе логических элементов. В результате выполнения этого логического проектирования должна быть получена для каждого блока автомата функциональная схема, т. е. должно быть получено описание автомата на языке *структурного описания.*

При программной реализации автомата процесс логического синтеза, включающего частично второй и с третьего по пятый этапы, заменяют процессом программирования в базисе команд применяемого программного автомата (микропроцессора, микроЭВМ и т. п.).

На этапе *технического проектирования* решаются следующие три главные задачи:

1) на основе функциональной схемы каждого из блоков производится разбиение логических элементов по корпусам с выбором типов корпусов (БИС или СБИС), корпусов по ТЭЗ, а ТЭЗ — по модулям с учетом минимальной межкорпусной, межплатной и межмодульной проводности монтажа;

2) для полученных разбиений производится размещение элементов в корпусе (БИС или СБИС), корпусов—на ТЭЗ, а ТЭЗ — в модулях с учетом минимальной длины соединений и числа их пересечений;

3) для получения размещений производится трассировка печатных соединений на ТЭЗ и межплатных соединений в модуле.

В результате выполнения этапа технического проектирования должна быть получена полная документация в установленной форме на изготовление автомата.

При изготовлении автомата на БИС и особенно СБИС этап технологического проектирования должен быть автоматизирован полностью практически на всех основных технологических операциях. В

последнем случае после этапа технического проектирования должна быть выдана документация в таком виде, чтобы она могла быть использована непосредственно (без переработки вручную) в качестве исходной документации для выполнения автоматизированного технологического проектирования.

На основе опыта создания систем автоматизации проектирования ЭВМ и устройств автоматики можно выделить две основные разновидности таких систем: *автоматическая система сквозного проектирования и автоматизированная система проектирования.*

В первой системе не предполагается участие человека в процессе проектирования. Такая система требует введения только исходной информации, а само проектирование производится ЭВМ без вмешательства человека. Однако практика разработки таких систем показала, чго не только при существующем парке ЭВМ, но и при парке ЭВМ в обозримый период времени эффективная система такого типа не может быть создана не только для всех этапов проектирования, но даже для каждого этапа в отдельности.

В связи с этим практический интерес представляют системы автоматизации проектирования. В автоматизированной системе проектирования разумно сочетаются опыт и интуиция проектировщика с быстродействием ЭВМ, выполняющих громоздкие вычисления, благодаря чему проектировщик может рассмотреть большое число возможных вариантов.

Таким образом, в автоматизированной системе проектирования процесс проектирования основан на диалоге между человеком и машиной.

При проектировании сложных автоматов и ЭВМ широко применяются *экспертные вычислительные системы.*

1.7. Распознавание множеств автоматами

Автоматы Мура. Напомним, что конечный автомат называется *автоматом Мура,* если его функция выходов зависит только от состояний, т. е. для любых $q, x_i, x_j, \psi(q, a_i) = \psi(q, a_j)$. Функцию выходов автомата Мура естественно считать одноаргументной функцией; обычно ее обозначают буквой μ и называют функцией отметок (так как она каждому состоянию однозначно ставит в соответствие отметку — выход). В графе автомата Мура выход пишется не на ребрах, а при вершине. Общая модель конечного автомата, которая рассматривалась ранее, называется *автоматом Мили.*

Несмотря на то что автомат Мура — частный случай автомата Мили, возможности этих двух видов автоматов совпадают.

Теорема 1. Для любого автомата Мили существует эквивалентный ему автомат Мура.

Пусть дан автомат Мили $A = (X, Q, Y, \varphi, \psi)$, $X = \{x_1, ..., x_m\}$, $Q = \{ q_1,\ldots, q_n\}$. Определим автомат Мура A_M следующим образом: $X_M = X$, $Y_M = Y$, Q_M содержит $mn + n$ состояний: mn состояний q_{ij} $(i = 1, ..., n;\ j = 1, ..., m)$, соответствующих парам (q_i, x_j) автомата A, и n состояний q_{i0} $(i = 1, ..., n)$. Функции $\varphi_м$ и μ определяются так: $\varphi_м(q_{l0}, x_k) = q_{ik}$, Для $i = 1, ..., n$ $\varphi_м(q_{ij}, x_k) = q_{lk}$, где l таково, что $\varphi(q_j, x_k) = q_l$; $\mu(q_{i0})$ не определено; для остальных состояний $\mu(q_{ij}) = \psi(q_i, x_j)$.

Для доказательства теоремы достаточно показать, что для любого q_i и любого α $A(q_i, \alpha) = A_M(q_{i0}, \alpha)$. Это делается индукцией по длине α.

Из этой теоремы следует, что при исследовании возможностей автоматов достаточно пользоваться автоматами Мура. Это удобно потому, что автомат Мура можно рассматривать как автомат без выходов, состояния которого различным образом отмечены. Без потери общности можно считать, что этих отметок всего две (например, 0 и 1), и они делят состояния на два класса. Зафиксируем один из этих классов и будем называть его состояния заключительными. Это приводит еще к одному определению автомата— автомата без выходов $A = (X, Q, q_1, F)$, где $F \subseteq Q$ — множество заключительных состояний. В дальнейшем до конца настоящего пункта будут без специальных оговорок рассматриваться инициальные автоматы без выходов A и с начальным состоянием q_1.

Представление событий в автоматах. Множество групп во входном множестве называется *событием*. (Этот термин стал традиционным в теории автоматов, хотя и необязателен: можно было обойтись просто «множеством групп»). *Событие* $E \subseteq X^*$ *представимо в автомате* $A = (X, Q, \varphi, q_1, F)$, если $\varphi(q_1, \alpha) \in F$ тогда и только тогда, когда $\alpha \in E$. Всякому автомату (при фиксированных q_1 и F) однозначно соответствует представимое в нем событие; на графе автомата это событие изображается множеством всех путей, ведущих из q_1 в вершины из F. Событие называется представимым (в конечном автомате), если существует конечный автомат, в котором оно представимо. Другие названия этого понятия— множество, определимое, или допускаемое, или распознаваемое конечным автоматом. Все эти термины также не обязательны, поскольку *представимое в автомате событие — это конечно-автоматный аналог разрешимого множества*; событие E, представимое в автомате A, можно было бы назвать множеством, разрешимым автоматом A. Может оказаться, что $q_1 \in F$. В этом случае автомат, еще ничего не получив на входе, уже «что-то представляет».

Удобно считать, что это «что-то» — пустая группа (группа нулевой длины); она содержится в событии, представимом таким автоматом. Пустую группу будем обозначать *е.* Для любой группы α *е*α =α*е* =α; таким образом, в свободной полугруппе групп входного множества, где умножением является приписывание групп друг к другу, *е* играет роль единицы.

Пустую группу не следует путать с пустым событием (т. е. с пустым множеством). Автомат представляет пустое событие, если ни одно из его заключительных состояний не достижимо из начального состояния.

Пример 1. 1) Любое конечное множество групп $E = \{\alpha_1, ..., \alpha_k\}$ представимо в автомате. Идея построения автомата по конечному множеству групп иллюстрируется графом на рис. 1.31, *а,* где заключительные состояния $q_{n-k}, ..., q_{n-1}$ изображены двойным кружком.

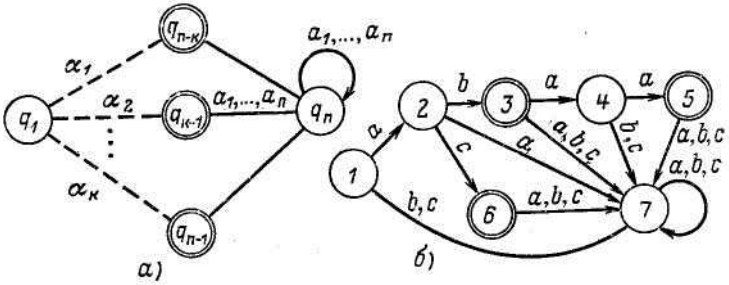


Рис. 1.31.

Для конкретных множеств эта идея модифицируется в связи с тем, что группы могут иметь общие начала (тогда начала соответстующих путей нужно объединить, чтобы не нарушить условие автоматности) либо просто содержаться друг в друге (тогда из одного заключительного состояния имеется путь в другое заключительное состояние). Пример автомата для $E = \{ab,\ ac,\ abaa\}$ с заключительными состояниями и $F = \{3, 5, 6\}$ приведен на рис. 1.84, *б.*

В автомате, представляющем конечное множество групп путь из начального состояния в любое заключительное состояние не может содержать циклов или содержаться в цикле, так как тогда имелось бы бесконечное множество путей из начального состояния из F и соответствующее событие было бы бесконечным. Поэтому такой автомат не может быть сильно связным; он является устройством, так сказать, одноразового действия.

2) Автономные автоматы представляют события в одноэлементном множестве; группы в таких событиях отличаются только длиной. Граф автомата, заданного приведенной ниже таблицей

q	a
1	3,0
2	4,0
3	4,0
4	7,0
5	4,2
6	5,0
7	6,1
8	9,0
9	9,1

изображен на рис. 1.32 (входные элементы на ребрах опущены; выходное множество $Y = \{0, 1, 2\}$).

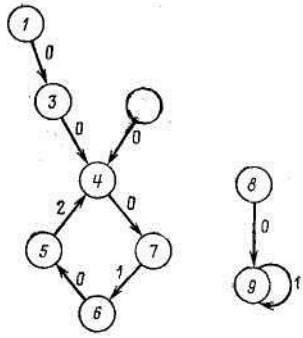


Рис. 1.32.

автомат имеет начальное состояние 1 и $F = \{7\}$ (выходы опускаем) представляет бесконечное событие, состоящее из всех групп, длины которых при делении на 4 дают в остатке 3. Если же положить $F=\{2\}$, то этот автомат представляет пустое событие.

3) Автомат, граф которого приведен на рис. 1.33 ($F = = \{1\}$), представляет бесконечное множество $\{e, aba, abaaba, ..., (aba)^n ...\}$.

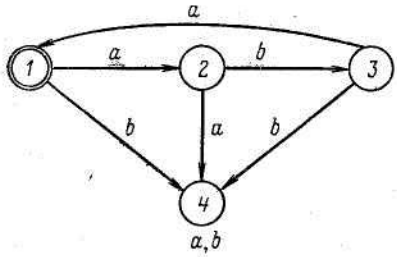


Рис. 1.33.

События — это множества конечных групп. Однако можно говорить, что автомат распознает бесконечную последовательность элементов $\alpha = x_{i_1}, x_{i_1} x_{i_2} \ldots$, если он представляет множество $E = \{ x_{i_1}, x_{i_1} x_{i_2} \ldots\}$, состоящее из всех начальных отрезков последовательности α.

Теорема 2. Существуют события, непредставимые в конечных автоматах; более конкретно: никакая непериодическая бесконечная последовательность не распознаваема конечным автоматом.
Первая половина теоремы должна быть очевидной: во-первых, из мощностных соображений (множество событий несчетно, множество автоматов счетно), во-вторых, просто потому, что существуют неразрешимые множества. Поэтому интересно было бы получить пример множества, которое разрешимо (например, машиной Тьюринга), но непредставимо конечным автоматом. Существование такого примера утверждается второй половиной теоремы (пример эффективно заданной непериодической последовательности в множестве {0, 1} : 010110111011110...).

Пусть непериодическая последовательность $\alpha = x_{i_1} x_{i_2} \ldots$ распознаваема автоматом A с n состояниями. Тогда для любого ее начального отрезка $x_{i_1} \ldots x_{i_j}$ $\varphi(q_1, x_{i_1} \ldots x_{i_j}) = q_{i_j}$, где q_{i_j} — заключительное состояние; поэтому при переработке этой последовательности автомат проходит последовательность заключительных состояний $q_{i_1} \ldots q_{i_j}$, ... Так как Q_A конечно, то некоторое состояние встретится в этой последовательности дважды: $q_{i_j} = q_{i_{j+k}}$ и, следовательно, $\varphi(q_{i_j}, x_{i_{j+1}} \ldots x_{i_{j+k}}) = q_{i_j}$, причем все состояния, проходимые автоматом, заключительные. Поэтому, если на вход автомата в состоянии q_1 подавать бесконечную периодическую

последовательность $\alpha_1 = x_{i_1} \ldots x_{i_j}$ $(x_{i_{j+1}} \ldots x_{i_{j+k}})$, где в скобки взят ее период, то автомат будет проходить последовательность заключительных состояний. Следовательно, все начальные отрезки α_1 входят в событие, представимое автоматом, т. е. автомат не отличает α_1 от α и, значит, не распознает α вопреки предположению.

Из теоремы 2 следует, что класс множеств, представимых конечными автоматами, является лишь частью (собственным подклассом) класса разрешимых множеств. В свою очередь, из этого обстоятельства и теоремы Раиса вытекает, что *свойство множества «быть представимым в конечном автомате» алгоритмически неразрешимо.* Более точно это утверждение формулируется так. Рассмотрим класс всех вычислимых функций с областью значений в X^*. (Фиксация мнжества X не нарушает общности, так как значения любых других вычислимых функций можно эффективно кодировать группами из X^*.) Каждую функцию из этого класса можно считать функцией, перечисляющей некоторое подмножество X^*, т. е. событие в множестве X. Тогда в силу теоремы Раиса не существует алгоритма, который по данной функции определяет, представимо ли в конечном автомате множество, перечислимое этой функцией, или нет. Поэтому не имеет смысла описывать множества, представимые автоматами, в терминах произвольных разрешимых множеств; следует искать более слабые средства их описания. К изучению таких средств мы и переходим.

Алгебра регулярных событий. Пусть даны события E_1 и E_2 в множестве X. Введем три операции над событиями

1. *Объединение* $E_1 \vee E_2$ (обычное объединение множеств).
2. *Умножение (конкатенация)* E_1E_2: $E = E_1E_2$—эти множество всех групп вида $\alpha_1\alpha_2$, где $\alpha_1 \in E_1$, $\alpha_2 \in E_2$.
3. *Итерация* $E_1^* = e \vee E_1 \vee E_1E_1 \vee \ldots \vee (E_1)^n \vee \ldots = \bigvee_{i=0}^{\infty} E_1^i$.

События $\{x_i\}$, где $x_i \in X$, будем называть элементарными и обозначать просто буквами x_i. К элементарным отнесем также событие e.

Событие называется *регулярным,* если оно может быть построено из элементарных событий с помощью конечного числа применений объединения, умножения и итерации; эти операции также назовем регулярными. Иначе говоря, класс R регулярных событий — это наименьший класс подмножеств X^*, содержащий все множества $\{ x_i \}$, а также e и замкнутый относительно регулярных операций. Тем самым определена алгебра $(R; \vee, \bullet, *)$, основным множеством которой является некоторая система подмножеств X^*, а именно — класс R регулярных событий; элементарные события — образующие этой

алгебры. Из определения следует, что каждый элемент этой алгебры (т. е. регулярное событие) может быть описан формулой, содержащей символы образующих (*e* и элемента *X)* и знаки регулярных операций под ними. Такие формулы называются *регулярными выражениями.* Регулярные выражения эквивалентны, если они описывают одно и то же регулярное событие.

Приведем некоторые эквивалентные соотношения в алгебре регулярных событий.

Если E, E_1, E_2, E_3 — регулярные события, то

$E_1(E_2 \vee E_3) = E_1E_2 \vee E_1E_3;$ (1.12)

$(E_1 \vee E_2) E_3 = E_1E_3 \vee E_2E_3;$ (1.13)

$(E_1E_2)E_3 = E_1(E_2E_3);$ (1.14)

$(E^*)^* = E^*;$ (1.15)

$E^* = e \vee E(E)^*.$ (1.16)

Кроме того, напомним, что объединение ассоциативно и коммутативно.

Пример 2. 1) Давно употребляемое нами обозначении X^* для множества всех групп в мнжестве X может теперь показаться двусмысленным, поскольку * обозначает итерацию. Однако оба смысла этого обозначения совпадают: регулярное выражение $(x_1 \vee \ldots \vee x_n)^*$ действительно задает множество всех групп (включая пустую) в множестве X.

2) Множество, состоящее из одной группы $x_{i_1} \ldots x_{i_k}$ является регулярным событием, поскольку любое выражение вида $x_{i_1} \ldots x_{i_k}$ регулярно: оно построено из элементов с помощью конкатенации.

Любое конечное множество групп $E = \{\alpha_1, \ldots, \alpha_k\}$ регулярно и описывается выражением $E = \alpha_1 \vee \ldots \vee \alpha_k$ не содержащим итерации.

Обратно, если регулярное выражение E не содержит итерации, то раскрытие скобок преобразует его к виду $\alpha_1 \vee \ldots \vee \alpha_k$; следовательно, E описывает конечное событие. Если же E содержит итерацию, то оно бесконечно, за исключением случаев, когда итерация применяется только к e ($e^* = e$, т. е. конечно).

3) Регулярное выражение вида X^*E, где E — конечное обытие, не содержащее пустой группы, описывает бесконечное событие, содержащее все группы из X^*, кончающиеся группами из E. Такие события называются *определенными,* или дефинитными. Например, событие $(a \vee b \vee c)^*(a \vee cb)$ содержит все группы в множестве $\{a, b, c\}$, кончающиеся на a или cb.

4) Событие $E_1 = (a \vee b)^* c (a \vee b)^* c (a \vee b)^*$ состоит из всех групп в множестве *{a, b, c},* содержащих *c* ровно два раза. Событие E_1^* состоит из всех групп, содержащих *c* четное число раз.

5) Регулярное событие *E* называется *асинхронным событием,* если для любых групп α_1, α_2 и элементы *x* из того, что $\alpha_1 x^k \alpha_2 \in E$ для некоторого *k,* следует, что $\alpha_1 x^k \alpha_2 \in E$ для любого $k = 1,2...$; иначе говоря, если $\alpha \in E,$ то в *E* содержатся все группы, полученные из *a* повторениями некоторых элементов а либо вычеркиванием из α некоторых повторений элементов. Например, если *E* — асинхронное событие и $abbcccd \in E,$ то $abcd \in E,$ $aabccdd \in E$ и т. д. Регулярные выражения для асинхронных событий могут быть построены из событий aa^* с помощью тех же трех операций; при этом они не должны содержать подформул вида aa^*aa^*. Например, событие $\{aa^*bb^* \vee cc^*)^*$ асинхронно, а событие $(aacc^* \vee cc^*)^*$ не является асинхронным, так как оно содержит группу *aac,* но не содержит группы *ac,* получающейся из *aac* вычеркиванием повторения *a*.

Регулярные события тесно связаны с автоматами. Изучение этой связи удобно вести, используя описание событий с помощью графов, к которому мы сейчас и переходим.

Источники. Уже говорилось, что на графе автомата событие, представляемое автоматом, изображается множеством путей из начальной вершины в заключительные вершины. Аналогичным образом для описания множеств групп можно использовать произвольные графы (не обязательно автоматные), на ребрах которых написаны буквы элементов. Такие графы называются *источниками.* Более точно, источником над множеством *X* называется ориентированный граф, в котором

1) выделены начальные и заключительные вершимы (вершина может быть одновременно и начальной, и заключительной),

2) на каждом ребре написана либо буква из *X,* либо пустая группа *e* (такие ребра назовем пустыми).

Каждым источник *G* однозначно определяет событие *E* в множестве *X,* порождаемое множеством всех путей из начальных вершим в заключительные (если путь содержит пустое ребро, то ему соответствует группа $\alpha e \beta = \alpha\beta$). В этом случае говорят, что источник G представляет событие *E.* Два источники называются эквивалентными, если они представляют одно и то же событие. Граф автомата без выходов — это частный случай источника.

Для любого источника *G* существует эквивалентный ему двухполюсный источник G_0 (с одной начальной и одной заключительной вершиной, которые могут совпадать), строящийся

так. Если в G имеется несколько начальных вершин, то в G_0 вводится новая вершина q_0, которая объявляется единственной начальной вершиной G_0 и соединяется с прежними начальными вершинами G пустыми ребрами (ребер, заходящих в q_0, в G_0 нет). Если в G имеется несколько заключительных вершин, то в G_0 вводится новая вершина q_z, которая объявляется единственной заключительной вершиной G_0; из прежних заключительных вершин в q_z проводятся пустые ребра; ребер, выходящих из q_z в G_0, нет. В остальном G_0 совпадает с G.

Теорема 3. Для любого регулярного события E существует двухполюсный источник, представляющий E.

Теорема доказывается индукцией по глубине регулярного события E. Элементарное событие представляется источником, состоящим из двух вершин—начальной и заключительной и ребра, идущего из начальной вершины в заключительную, на котором написано данное событие *(a_i или e)*.

Пусть теперь построены двухполюсные источники: G_1, представляющий регулярное событие E_1, и G_2, представляющий регулярное событие E_2; их начальные вершины q_{10} и q_{20}, заключительные вершины — q_{1z} и q_{2z} соответственно. Тогда источник G с начальной вершиной q_0 и заключительной вершиной q_z, который представляет событие E — результат регулярной операции над E_1 и E_2, строится так.

1. $E = E_1 \vee E_2$. G строится «параллельным соединением» G_1 и G_2 (рис. 1.34, *а)*.

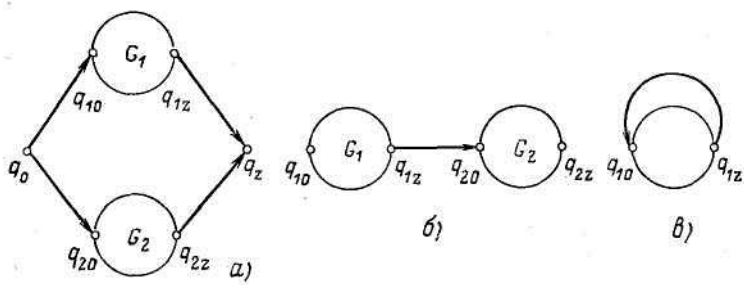


Рис. 1.34.

Он состоит из источников G_1 и G_2 и новых вершин q_0 и q_z; из q_0 проводятся пустые ребра в q_{10} и q_{20}. из q_{1z} и q_{2z} проводятся пустые ребра в q_z.

2. $E = E_1E_2$. G строится «последовательным соединением» G_1 и G_2 (рис. 1.34, б): из q_{1z} проводится пустое ребро в q_{20}, начальной вершиной G объявляется q_{10}, заключительной вершиной G объявляется q_{2z}.

3. $E = E_1^*$. G строится зацикливанием G_1 (рис. 1.34, *в):* из q_{1z} проводится пустое ребро в q_{10}, q_{10} объявляется начальной и заключительной вершиной G.

Доказательство того, что построенные таким образом источники действительно представляют соответствующие события, несложно. Пусть, например, $E = E_1E_2$ и $\alpha \in E$. Тогда $\alpha = \alpha_1\alpha_2$, где $\alpha_1 \in E_1$, $\alpha_2 \in E_2$. По предположению, G_1 представляет E_1, G_2 представляет E_2; поэтому существуют путь α_1 из q_{10} в q_{1z} и путь α_2 из q_{20} в q_{2z}. Но тогда по построению существует путь $\alpha_1\alpha_2$ из q_{10} (начальной вершины G) в q_{2z} (заключительную вершину G). И наоборот, всякий путь α из q_{10} в q_{2z} обязательно проходит через q_{1z} и q_{20}, поэтому он имеет вид $\alpha = \alpha_1\alpha_2$, где α_1 — путь из q_{10} в q_{1z} и α_2 — путь из q_{20} в q_{2z}, откуда следует, что $\alpha_1 \in E_1$ и $\alpha_2 \in E_2$. Доказательства для объединения и итерации аналогичны.

Введение пустых ребер (вместо простого склеивания вершин) объясняется необходимостью избежать «ложных путей». Некоторые из введенных пустых ребер в дальнейшем можно удалять, не меняя представляемого события.

Детерминизация источников. Если источник имеет одну начальную вершину, не содержит пустых ребер и удовлетворяет условиям автоматности, то он является графом автомата без выходов. Такой источник часто называют *детерминированным источником.* Этот термин связан с интерпретацией произвольного источника как *недетерминированного автомата*, т. е. автомата, для которого $\varphi(q, x)$ определена неоднозначно; точнее, значением $\varphi(q, x)$ является подмножество Q (быть может, пустое). Иначе говоря, при фиксированной вершине q символу x соответствует множество ребер, а группе α — множество путей, ведущих из q; на каждом шаге недетерминированный автомат как бы совершает выбор из возможных ребер (q, x). Понятие представимости в недетерминированном автомате совпадает с понятием представимости в источнике.

Теорема 4 (теорема детерминизации). Для любого источника G с n вершинами существует эквивалентный ему детерминированный источник G', имеющий не более чем 2^n вершин.

Назовем множество $\tilde{q}$ вершин источника замкнутым, если из того, что $q_i \in \tilde{q}$, следует, что $\tilde{q}$ принадлежит любая вершина, в которую из q_i ведет пустое ребро. Для источника без пустых ребер все множества вершин замкнуты. Обозначим через $\tilde{q}_1$ наименьшее замкнутое множество вершин G, содержащее все начальные вершины G.

Источник *G'* строится так. Образуем все замкнутые подмножества вершин *G* (их не более чем 2^n) и каждому такому подмножеству поставим в соответствие вершину *G'* (в дальнейшем эти подмножества и соответствующие им вершины будем отождествлять и обозначать $\tilde{q}_i$). Начальной вершиной *G'* объявим $\tilde{q}_0$, заключительными вершинами — все подмножества $\tilde{q}_i$, содержащие хотя бы одну заключительную вершину *G*. Если в источнике *G* из множества вершин $\tilde{q}_i$ пути *x* (они могут содержать пустые ребра) ведут в множество $\tilde{q}_j$ (т. е. $\tilde{q}_j$ — это множество концов всех ребер *x*, начала которых принадлежат множеству $\tilde{q}_i$), то в источнике *G'* из вершины $\tilde{q}_i$ проводится ребро *x* в вершину $\tilde{q}_j$. Если же в *G* никакая из вершин множества $\tilde{q}_i$ не имеет выходящего из нее пути *x*, то в *G'* из вершины $\tilde{q}_i$ проводится ребро *x* в вершину *Ø*, соответствующую пустому подмножеству вершин *G*. Таким образом, каждой вершине $\tilde{q}_i$ источника *G'* и каждому входному символу *x* соответствует ровно одно ребро *x*, выходящее из вершины $\tilde{q}_i$, и, следовательно, источник *G'*— детерминированный. Другими еловами, *G'* — это граф автомата с начальным состоянием $\tilde{q}_1$ *;* описанное ранее построение ребер *G'* определяет функцию переходов автомата: $\varphi_G(\tilde{q}_i, x) = \tilde{q}_j$.

Источник *G'* обладает следующим свойством: в *G'* непустой путь α из $\tilde{q}_1$ в $\tilde{q}_j$ существует тогда и только тогда, когда в *G* для любой вершины $q \in \tilde{q}_j$ существует путь α из некоторой начальной вершины $q_1 \in \tilde{q}_1$ в *q*. (Если α = *e,* то $\tilde{q}_j = \tilde{q}_1$ по условию замкнутости; пустых ребер в *G'* по построению нет.) Доказательство проведем индукцией по длине группы α. Если α = *x,* то это свойство выполняется по построению ребер *x* в *G'*. Предположим, что оно выполняется для всех групп α длины $\le k$, и докажем, что оно выполняется для α*x,* где *x* — произвольный входной элемент.

Пусть в *G'* имеется непустой путь α*x* из $\tilde{q}_1$ в $\tilde{q}_j$: $\varphi_G(\tilde{q}_1, \alpha x) = \tilde{q}_j$. Если $\varphi_G(\tilde{q}_1, \alpha) = \tilde{q}_l$, то из $\tilde{q}_l$ в $\tilde{q}_j$ ведет ребро *x*. По предположению, в *G* для любой вершины $q^* \in \tilde{q}_l$ существует путь α из некоторой

начальной вершины q_1 в q^*. По построению G' из того, что в G' есть ребро x из $\tilde{q}_l$ в $\tilde{q}_j$, следует, что в G для любой вершины $q \in \tilde{q}_j$ - найдется вершина $\tilde{q}_l$, из которой ведет путь x в q; поэтому в G имеется путь x из q^* в q и, следовательно, путь αx из q_1 в q.

Аналогично доказывается обратное утверждение: в предположении, что в G для любой вершины $q \in \tilde{q}_j$ есть путь αx из некоторой начальной вершины $q_1 \in \tilde{q}_1$ в q, рассматривается множество всех путей αx из начальных вершин в вершины из $\tilde{q}_j$ и множество $\tilde{q}_l$ всех вершин, в которые ведут отрезки α этих путей. С использованием индуктивного предположения и построения ребер G' показывается, что в G' $\varphi(\tilde{q}_1, \alpha x) = \tilde{q}_j$.

Из доказанного свойства G' и определения заключительных вершин G' следует, что в G' путь α из $\tilde{q}_1$ в заключительную вершину существует тогда и только тогда, когда в G имеется путь α из начальной вершины в заключительную. Поэтому $\alpha \in E'$, если и только если $\alpha \in E$.

Из теорем 3 и 4 непосредственно следует одна из важнейших теорем теории автоматов, доказанная впервые Клини.

Теорема 5. (теорема синтеза). Для любого регулярного события E существует конечный автомат, представляющий это событие.

Метод построения (синтеза) автомата, представляющего E, заключается в том, что сначала строится источник, представляющий E, а затем этот источник детерминизируется путем процедуры, описанной при доказательстве теоремы 4. На практике процедура детерминизации несколько модифицируется с целью ее упрощения. Дело в том, что некоторые подмножества вершин G (т. е. состояния G') не достижимы из начальной вершины; их удаление не изменит события, представляемого источником. Поэтому в таблицу переходов G' включаются только те подмножества, которые порождаются процедурой детерминизации, начатой с подмножества q_1 (см. пример 3).

При такой модификации построенный автомат может иметь меньше чем 2^n состояний (n— число вершин исходного источника). Однако в общем случае эту оценку понизить нельзя. Например, в трехэлементном множестве для любого n существует источник с n состояниями, такой, что любой эквивалентный ему автомат имеет не менее чем 2^n состояний. Пример такого источника для $n = 5$ приведен на рис. 1.35.

Пример 3. Построим автомат, представляющий событие $(ab \vee c^*)\ (a \vee bc)^*\ a$. В соответствии с ранее изложенным синтез автомата проводится в два этапа. Сначала по событию строится представляющий его источник G. Он изображен на рис. 1.36.

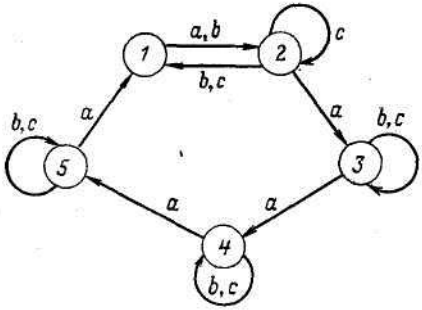


Рис. 1.35.

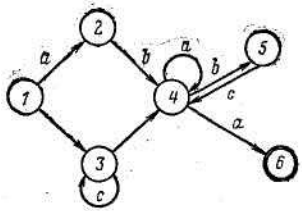


Рис. 1.36.

В этом источнике ребра, которым не приписано букв, — пустые; некоторые «лишние» пустые ребра, возникающие, если строго следовать методу теоремы 3, удалены. Вершина *1* является начальной, вершина *6* — заключительной.

Затем методом теоремы 4 построенный источник детерминизуется, при этом строятся только подмножества, достижимые из начального подмножества {1}. Функция переходов (переводов) детерминированного источника G', вершинами которого служат подмножества вершин G, приведена в табл. 1.5,*а* (знаком Ø, как обычно, обозначено пустое множество); после перенумерации этих подмножеств она приобретает обычный вид таблицы переходов (табл. 1.5,*б*).

Таблица 1.5,*а*

	a	*b*	*c*
1	2, 4, 6	5	3,4
2, 4, 6	4,6	4,5	Ø
5	Ø	Ø	4
3,4	4,6	5	3,4
4,6	4,6	5	Ø
4,5	4,6	5	4
Ø	Ø	Ø	Ø
4	4,6	5	Ø

Таблица 1.5,*б*

	a	*b*	*c*
1	2	3	4
2	5	6	7
3	7	7	8
4	5	3	4
5	5	3	7
6	5	3	8
7	7	7	7
8	5	3	7

В табл. 1. 1,*б* заключительные состояния 2 и 5 выделены; они соответствуют подмножествам из табл. 1.50,*а*, содержащим заключительное состояние 6 источника *G*.

Анализ автоматов. Если процесс построения автомата по заданному событию (и вообще по некоторому описанию множества групп во входном множестве) называется синтезом, то обратный процесс — получения описания множества входных групп, представимого заданным автоматом, называется анализом автомата. Из теоремы 6 видно, что результат анализа также может быть описан с помощью регулярных событий (этот факт также доказан Клини).

Теорема 6 (теорема анализа). Всякое событие, представимое конечным автоматом, регулярно.

Определим индуктивно событие E^k_{ij}:

1) E^0_{ij} — множество всех входных элементов x, таких, что $\varphi(q_i, x) = q_j$;

2) $E^k_{ij} = E^{k-1}_{ij} \vee E^{k-1}_{ik}(E^{k-1}_{kk})^* E^{k-1}_{kj}$. Обозначим через M^k_{ij} множество всех входных групп, ведущих из q_i в q_j и не проходящих при этом через состояния с номерами большими, чем k.

Лемма. $E^k_{ij} = M^k_{ij}$.

Докажем эту лемму по индукции. Для $k = 0$ она очевидна. Пусть она верна для $k = r-1$. Тогда $M^r_{ij} = E^{r-1}_{ij} \cup \tilde{M}^r_{ij}$, где $\tilde{M}^r_{ij}$ — множество всех групп из M^r_{ij}, проходящих через q_r. Если $\alpha \in \tilde{M}^r_{ij}$ и проходит через q_r s раз, то α представимо в виде $\alpha = \alpha_0 \alpha_1 \ldots \alpha_{s-1} \alpha_s$, где $\alpha_0 \in E^r_{ir}, \alpha_s \in E^r_{rj}, \alpha_1 \ldots \alpha_{s-1} \in E^{r-1}_{rr}$ (т. е. содержит $s-1$ цикл, проходящий через q_r). Поэтому

$$\alpha \in E^{r-1}_{ir}(E^{r-1}_{rr})^* E^{r-1}_{rj}.$$

И наоборот, если это включение выполнено, то из регулярной формулы его правой части и индуктивного предположения следует, что $\alpha \in \tilde{M}^r_{ij}$. Таким образом, $\tilde{M}^r_{ij} = E^{r-1}_{ir}(E^{r-1}_{rr})^* E^{r-1}_{rj}$ и, следовательно, $M^r_{ij} = E^{r-1}_{ij} \cup E^{r-1}_{ir}(E^{r-1}_{rr})^* E^{r-1}_{rj} = E^r_{ij}$, что и доказывает лемму.

Очевидно, что для автомата с n состояниями, начальным состоянием q_1 и заключительными состояниями $q_{z_1}, \ldots, q_{z_p}$ представляемое им событие имеет вид $M^n_{1z_1} \vee \ldots \vee M^n_{1z_p}$. Из леммы следует, что это событие регулярно.

Доказательство теоремы по существу представляет собой алгоритм анализа. Изложенный здесь алгоритм (алгоритм Макнотона— Ямады) очень компактен и удобен для доказательств, однако приводит, как, впрочем, и другие алгоритмы анализа, к довольно громоздким регулярным выражениям. Заметим при этом, что в алгебре регулярных

событий нет нетривиальных методов отыскания минимального регулярного выражения, эквивалентного данному. В то же время из теоремы синтеза следует, что проблема распознавания эквивалентности регулярных выражений разрешима, так как по выражениям E_1, E_2, можно построить представляющие их автоматы A_1, A_2 и проверить их на эквивалентность (сравнив автоматы, минимальные для A_1, A_2); следовательно, метод минимизации регулярных выражений существует, хотя он очень громоздкий.
Итак, теоремы анализа и синтеза устанавливают взаимно однозначное соответствие между автоматами (с точностью до эквивалентности) и регулярными событиями. Поэтому регулярные события —это один из основных языков для описания поведения автоматов, используемых в теоретичсских исследованиях. Важным свойством регулярных событий является их замкнутость относительно булевых операций: пересечение регулярных событий и дополнение к регулярному событию также регулярны (объединение — регулярная операция по определению). Действительно, если событие F описывается источником G, то E описывается источником G, который отличается от G тем, что заключительными вершинами G служат те и только те вершины, которые не являются заключительными в G. Так как для любых множеств E_1 и E_2 $E_1 \cap E_2 = \overline{\overline{E}_1 \cup \overline{E}_2}$, то из источников для E_1 и E_2 можно построить источник для $E_1 \cap E_2$; в силу теорем синтеза и анализа всякое событие, заданное источником, регулярно и искомая замкнутость доказана. Этот факт позволяет расширить язык регулярных событий, дополнив его операциями дополнения и пересечения.

Пример 4. 1) Автономные автоматы представляют регулярные события в одноэлементном множестве. Если для автомата, указанного в приме 1, п. 2 (с начальным состоянием 1), множество заключительных состояний $F = \{7\}$, то он представляет событие $E = xxx(xxx)^*$, а если $F = \{3, 4, 6\}$, то
$E = x \vee xx\,(xxxx)^* \vee xxxx\,(xxxx)^* = x \vee xx\,(e \vee xx)\,(xxxx)^*$.
2) Автомат на рис. 1.33 представляет событие $(aba)^*$.
3) Автоматы, представляющие определенные события (см. пример 2, п. 3), называются определенными автоматами или автоматами с конечной глубиной памяти. Смысл последнего термина — в том, что такие автоматы одинаково реагируют на группы, у которых их «хвосты» (заранее фиксированной для данного автомата длины) совпадают.

4) Автомат называется *асинхронным,* если в нем для любых qt и a б $(q_h\, aa) =$ б $(q_h\, a)$. Состояние $q_j =$ б (q, a) с таким свойством называется

устойчивым по *a;* поэтому можно сказать, что в асинхронном автомате любое состояние устойчиво по любому входу, ведущему в это состояние. Событие является асинхронным, если и только если оно представимо в асинхронном автомате (докажите!).

5) Пусть $E = (a \vee b \vee c)^* (ab \vee c)$. Тогда

$\overline{E} = e \vee b \vee (a \vee b \vee c)^* (bb \vee cb \vee a)$.

1.8. Сети автоматов

Комбинационные и логические автоматы. Автомат называется *комбинационным,* если для любого входного символа a и любых состояний q_i и q_j $\lambda(q_i, a) = \lambda(q_j, a)$, иначе говоря, если выходной символ не зависит от состояния и определяется текущим входным символом. В таком автомате все состояния эквивалентны и, следовательно, минимальный комбинационный автомат имеет одно состояние. Функция переходов в нем вырождена; его поведение однозначно задается функцией выходов с одним аргументом:

$\lambda(a_i) = v_j$.

Автомат называется *логическим,* если его входной алфавит состоит из 2^m двоичных наборов длины m, а выходной — из 2^n двоичных наборов длины n. Функция выходов логического комбинационного автомата — это просто система n логических функций от m переменных (i-я функция определяет значения i-й компоненты в выходном векторе автомата).

Последовательные автоматные вычисления. В этом параграфе речь будет идти о различных способах комбинирования автоматов друг с другом. Если рассматривать автоматы просто как частный случай алгоритмов, то естественно прежде всего изучить автоматные блок-схемы, т. е. блок-схемы, все блоки которых являются конечно-автоматными алгоритмами. Поскольку общие свойства блок-схем оказываются очень простыми, ограничимся рассмотрением этих свойств на примере.

Пусть дана блок-схема на рис.1.37, на которой S_1 и S_2 — автоматные операторы, а S_3 — автоматный предикат.

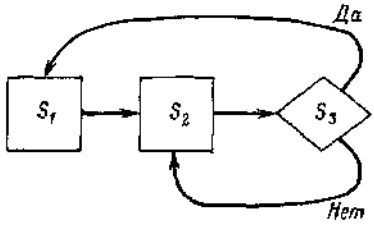


Рис. 1.37

Как и обычно, сама блок-схема не содержит указаний о том, с какой информацией работают автоматы; она лишь указывает передачи управления: сначала работает S_1, затем S_2, затем автомат S_3 проверяет условие; при положительном ответе работает S_1, при отрицательном — S_2. Понятно, что при таком взаимодействии автоматы должны «уметь останавливаться», как алгоритмы. Формально будем считать, что каждый из автоматов в заключительном состоянии не определен, т. е. не воспринимает последующих входных сигналов. Автомат, вычисляющий предикат, будет иметь два заключительных состояния — для «да» и для «нет». Зафиксируем (пока без входного алфавита) конкретные таблицы переходов для S_1, S_2, S_3 (см. табл. 1.6, а—в) и рассмотрим два крайних случая, которые возможны для входной информации автоматов. (Выходы для данных рассмотрений несущественны: для определенности будем считать, что они во всех клетках различны.)

Таблица 1.6

q_{11}	q_{11}	q_{12}	q_{12}
q_{12}	q_{12}	q_{11}	q_{12}
q_{13}	—	—	—

а)

q_{21}	q_{23}	q_{22}
q_{22}	q_{22}	q_{22}
q_{23}	q_{21}	q_{23}
q_{24}	—	—

б)

q_{31}	q_{31}	q_{320}	q_{321}
q_{32}	q_{321}	q_{321}	q_{321}
q_{320}	—	—	—
q_{321}	—	—	—

в)

Случай 1: входные алфавиты исходных автоматов не пересекаются. Пусть $A_1 = \{a, b, c\}$, $A_2 = \{d, f\}$, $A_3 = \{g, h, i\}$. Тогда блок-схеме на рис. соответствует табл. 1.37, *а.* В ней заключительные состояния отождествлены с начальными состояниями последующих автоматов.

Таблица 1.7

	a	*b*	*c*	*d*	*f*	*g*	*h*	*i*
q_{11}	q_{11}	q_{21}	q_{12}					
q_{12}	q_{21}	q_{11}	q_{12}					
q_{21}				q_{23}	q_{22}			
q_{22}				q_{31}	q_{22}			
q_{23}				q_{21}	q_{23}			
q_{31}						q_{31}	q_{21}	q_{32}
q_{32}						q_{11}	q_{11}	q_{31}

а)

	a	*b*	*c*	*d*	*f*	*g*	*h*	*i*
1	1	1	2	3	2	1	1	2
2	1	1	2	1	2	1	1	1
3	—	—	—	1	3	—	—	—

б)

Полученную таблицу переходов можно минимизировать по правилам минимизации частичных автоматов; результат приведен в табл.1.7, б. Разумеется, автомат, описываемый в табл.1.7, б, будет работать в соответствии с блок-схемой рис. 37 только в том случае, если в нужные моменты времени будут происходить «переключения внешней среды», т. е. переходы от алфавита A_1 к алфавиту A_2 и т. д. Поскольку эти моменты соответствуют переходам от одного подавтомата к другому и, следовательно, автоматом с табл.1.7, б распознаются, то переключения можно осуществлять с помощью выходных сигналов. В нашем примере переключающими сигналами будут

$\lambda(1, b)$, $\lambda(2, a)$, $\lambda(2, d)$, $\lambda(1, h)$, $\lambda(2, g)$, $\lambda(2, h)$.

Итак, в этом случае блок-схема из автоматов — это также автомат, алфавит которого является объединением алфавитов исходных автоматов, а число состояний не превосходит $\max |Q_i|$, где максимум берется по мощностям множеств состояний исходных автоматов. Можно сказать, что такой автомат (табл. 1.7, б) реализует блок-схему рис. 1.37 на общей памяти с переключением входов.

Случай 2: входные алфавиты исходных автоматов совпадают или включаются друг в друга. Если в рассматриваемом примере положить $A_1 = A_3 = \{a, b, c\}$, $A_2 = \{a, b\}$, то для блок-схемы получим табл.1.8.

Таблица 1.8

	a	b	c
q_{11}	q_{11}	q_{21}	q_{12}
q_{12}	q_{21}	q_{11}	q_{13}
q_{21}	q_{23}	q_{22}	—
q_{22}	q_{31}	q_{22}	—
q_{23}	q_{21}	q_{23}	—
q_{31}	q_{31}	q_{31}	q_{32}
q_{32}	q_{11}	q_{11}	q_{31}

В этом случае объединения состояний, вообще говоря, не происходит, если только состояния из разных подавтоматов не оказываются эквивалентными. Общий входной алфавит является объединением исходных алфавитов; его мощность равна мощности максимального из исходных алфавитов. Число состояний полученного автомата не превосходит суммы чисел состояний исходных автоматов.

Очевидно, что остальные возможные случаи соотношения алфавитов — это их частичное пересечение; получающиеся таблицы имеют вид, «промежуточный» между табл. 1.7 и 1.8; возможности их минимизации будут зависеть от конкретного вида исходных автоматов. Наконец, заметим, что для графов автоматов указанные преобразования крайне просты и сводятся к отождествлению заключительных вершин с соответствующими начальными вершинами. Правда, возможности минимизации видны на графе не столь наглядно.

Подведем итог. Блок-схема из конечных автоматов $S_1, \ldots, S_k$, работающих последовательно (т. е. неодновременно), — это конечный автомат S, следовательно, множество автоматов замкнуто относительно условного и безусловного перехода. Мощности входного алфавита и множества состояний S не превосходят суммы мощностей соответственно входных алфавитов и множеств состояний $S_1, \ldots, S_k$. Автомат S называют суммой $S_1, \ldots, S_k$; часто, впрочем, термин «сумма» относят лишь к операции безусловного перехода (т. е. последовательного соединения с неодновременной работой).

Синхронные сети из автоматов. Если автоматы рассматривать как устройства с входами и выходами, то присоединение выходов одних автоматов ко входам других дает *схему, или сеть из автоматов,* все автоматы которой работают одновременно. Под состоянием сети из m одновременно работающих автоматов $S_1, \ldots, S_m$ (компонент сети)

понимается вектор $(q_{i_1}, \ldots, q_{i_m})$, где q_{i_j} — состояние автомата S_r Поэтому в общем случае число возможных состояний сети равно произведению чисел состояний составляющих ее компонент. Возникает вопрос, является ли сеть из автоматов автоматом; если да, то как получить ее автоматное описание из описаний ее подавтоматов. Прежде всего заметим, что вектор-состояние сети указывает, в каких состояниях находятся компоненты сети в один и тот же момент времени. Таким образом, **при описании автоматных сетей — в отличие от абстрактных автоматов — необходимо явно вводить понятие времени. Существуют два основных способа введения времени — синхронный и асинхронный**. Синхронный способ заключается в следующем. Вводится шкала времени, которая делится на отрезки одинаковой длины (такты); границы тактов называются моментами автоматного времени и нумеруются натуральными числами, начиная с нуля. Длина такта принимается за единицу времени. Входное слово (последовательность букв) рассматривается как временная последовательность сигналов или импульсов (**каждый сигнал соответствует букве**); интервал между соседними импульсами равен в точности длине такта. (При такой интерпретации импульс является «точечным». Можно считать, что сигнал длится на протяжении всего такта, но тогда в случае, если сигналы в соседних тактах одинаковы, нужны внешние часы, указывающие границы такта). Следовательно, слово длины k занимает во времени ровно k тактов; его буквы можно считать функциями от времени: $a(t)$ — буква, появившаяся на входе в момент t. Автоматные функции δ и λ реализуются с задержкой. Время задержки функции δ равно единице: $\delta(q(t), a(t)) = q(t+1)$; состояние $q(0)$ определено заранее. Время задержки функции λ обычно считается равным нулю: $\lambda(q(t), a(t)) = v(t)$, но иногда равным единице: $\lambda(q(t), a(t)) = v(t+1)$; во втором случае должно быть определено $v(0)$. Таким образом, под действием последовательности входных сигналов одинаковой длины каждый из автоматов порождает последовательность промежуточных или внутренних сигналов (реализующих состояния) и последовательность выходных сигналов, причем длины тактов этих последовательностей совпадают с длиной входного такта. Каким образом на практике достигается такая всеобщая синхронизация — за счет внешних синхронизирующих часов либо за счет идеальных временных характеристик автоматов и среды, в которой они функционируют — на данном уровне рассмотрения несущественно (но, разумеется, становится существенным при физической реализации таких сетей).

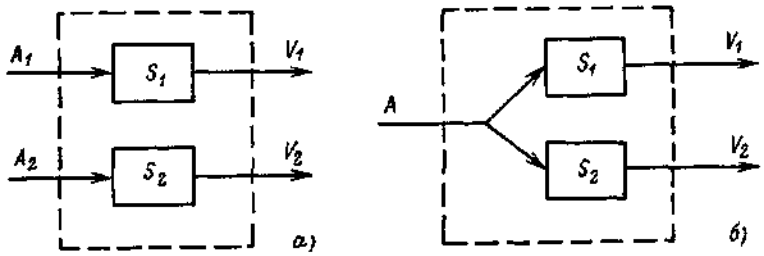


Рис.1.38.

Прежде чем говорить о сетях из автоматов общего вида, рассмотрим некоторые основные виды соединения автоматов.

1) **Параллельное соединение** (рис. 1.38): *а*— с раздельными входами и алфавитами A_1 и A_2; *б*— с общим входом и алфавитом A.

Могут сказать, что соединение *а* — это вообще не соединение; тем не менее пару автоматов $S_1 = (A_1, Q_1, V_1, \delta_1, \lambda_1)$; $S_2 = (A_2, Q_2, V_2, \delta_2, \lambda_2)$ можно рассматривать как один автомат $S = (A, Q, V, \delta, \lambda)$, который определяется так: $A = A_1 \times A_2$, $Q = Q_1 \times Q_2$, $V = V_1 \times V_2$; следовательно, входной символ автомата S — это пара символов: $a = (a^1, a^2)$, состояние автомата S — это пара состояний: $q = (q^1, q^2)$, где $q^1 \in Q_1$, $q^2 \in Q_2$. Далее $\delta(q, a) = \delta((q^1, q^2), (a^1, a^2)) = (\delta_1(q^1, a^1), \delta_2(q^2, a^2))$, т. е. смена состояний происходит независимо и одновременно; аналогично $\lambda(q, a) = (\lambda_1(q^1, a^1), \lambda_2(q^2, a^2)) = (v^1, v^2)$, где $v^1 \in V_1$; $v^2 \in V_2$. Автомат S называется *прямым произведением* автоматов S_1 и S_2.

До сих пор речь шла об автоматах с одним входом и одним выходом. Сеть на рис.1.38, *а* дает пример автомата, о котором удобно говорить, что он имеет несколько входов (в данном случае два) и несколько выходов. В общем случае про автомат, входные символы которого — векторы длины k, говорят, что он имеет k входов; это же касается и выходов. В частности, логический автомат с входными наборами длины m и выходными наборами длины n имеет m двоичных входов и n двоичных выходов.

Автомат на рис. 1.38, *б* определяется аналогично, с той разницей, что входные алфавиты S_1, S_2 и S совпадают; поэтому, например, $\delta(q, a) = (\delta_1(q^1, a), \delta_2(q^2, a))$.

2. **Последовательное соединение** (рис.1.39).

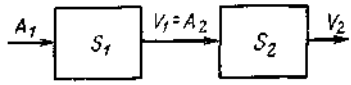


Рис.1.39

Эту сеть также можно описать как автомат $S = (A, Q, V, \delta, \lambda)$, причем $A = A_1$, $V = V_2$, $V_1 = A_2$; как и прежде, $Q = Q_1 \times Q_2$. Для определения δ и λ существенна задержка функции λ_1. Если задержка λ_1 равна нулю: $\lambda_1 (q^1 (t), a (t)) = v^1 (t)$, то

$$q (t + 1) = (q^1 (t + 1), q^2 (t + 1)) =$$
$$= (\delta_1 (q^1 (t), a (t)), \delta_2 (q^2 (t), \lambda_1 (q^1 (t), a (t)))). \quad (1.12)$$

Выражение в правой части зависит только от $q (t) = (q^1 (t), q^2 (t))$ и $a (t)$ и, следовательно, определяет $q (t + 1)$ как функцию от этих переменных. Эта функция и есть функция δ для S: $q (t + 1) = \delta (q (t), a (t))$; ее конкретная таблица зависит от таблиц δ_1, δ_2 и λ.

Если же время задержки λ_1 равно единице: $\lambda_1 (q^1 (t), a (t)) = v^1 (t + 1)$, то $q (t + 1) = (\delta_1 (q^1 (t), a (t)), \delta_2 (q^2 (t), \lambda (q^1 (t - 1), a (t - 1))))$ и зависимости только от предыдущего такта не получается. Общий метод получения автоматного описания для этого случая будет изложен позже, а пока рассмотрим очень простой пример. Пусть автоматы S_1 и S_2 имеют по одному состоянию, двоичные вход и выход, а на выходе переписывают то, что подано на вход, но с задержкой на такт: $\lambda (q (t), a (t)) = a (t + 1)$. В этом случае S_1 и S_2 на рис. 1.39 — это просто устройства задержки на такт или, как говорят, **элементы задержки**. Для этих элементов необходимо определить начальное значение выхода; положим его равным нулю. Казалось бы, такая сеть тривиальна и по свойствам прямого произведения $Q = Q_1 \times Q_2$ должна иметь одно состояние. Однако она осуществляет (при введенной ранее синхронной интерпретации тактов) отображение $S (\alpha) = 00\alpha_{-2}$ (α_{-2} обозначает слово a, у которого отброшены две последние буквы), которое нельзя реализовать автоматом с одним состоянием. Выход заключается в том, чтобы интерпретировать элемент задержки с двоичным входом и выходом как автомат с двумя состояниями, значения которых совпадают со значениями входа. Его таблица переходов приведена в табл.1.9.

Таблица 1.9

	0	1
q_0	q_0, 0	q_1 ,0
q_1	q_0, 1	q_1, 1

В этой таблице предполагается, что λ не имеет задержки: $\lambda(q(t), a(t)) = v(t) = q(t)$. Таким образом, автомат с одним состоянием и единичной задержкой на выходе оказывается эквивалентным автомату с двумя состояниями без задержки на выходе. Но тогда сеть на рис. 1.39 оказывается автоматом с четырьмя состояниями; ее таблица переходов строится из табл. 1.9 по формуле (1.12) и приведена в табл.1.10, где пары (q_i, q_j) обозначены ij, а значение выхода совпадает с состоянием второго элемента задержки.

Таблица 1.10

ij	0	1	выход
00	00	10	0
01	00	10	1
10	01	11	0
11	01	11	1

Кажется парадоксальным, что цепь из двух элементов задержек удается описать автоматом, в котором функция λ не имеет задержки. Это происходит потому, что задержки «загоняются» в состояния. Аналогично можно описать цепь из любого числа элементов задержки.

3. Обратная связь (рис.1.40): это соединение заключается в том, что выход автомата S_1 присоединяется к одному из его входов.

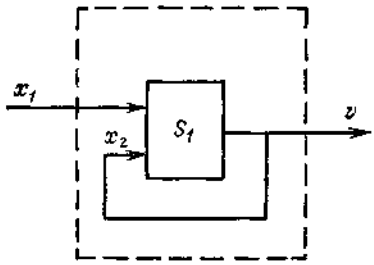


Рис.1.40

Рассмотрим случай, когда S_1 — комбинационный автомат без задержки, и предположим, что S_1 реализует логическую функцию $\overline{x_1 x_2}$; вход x_1 оставлен внешним, на вход x_2 подана замкнутая обратная связь. Пусть в момент f_v=0; тогда

$x_2(t) = 0$ и $v(t) = \overline{x_1(t)\, x_2(t)} = 1$, т. е. равен и нулю, и единице в один и тот же момент. Если же $v(t) = 1$, то $x_2(t) = 1$ и при $x_1(t) = 1$ $z(t) = \overline{x_1(t)\, x_2(t)} = 0$, т. е. опять получаем противоречие. Предоставляем читателю убедиться, что введение в обратную связь задержки устраняет противоречия.

Этот пример показывает, что сеть из автоматов, содержащая контур обратной связи без задержек, может не иметь конкретной автоматной интерпретации. Правда, это бывает не всегда, но тем не менее сети с контурами обратной связи без задержек в теории автоматов, как правило, исключаются из рассмотрения.

С другой стороны, обратная связь с задержкой оказывается мощным средством построения автоматов. Всякий автомат при синхронной интерпретации может быть реализован как сеть, состоящая из комбинационных автоматов и элементов задержки. Действительно, такая сеть для автомата с функциями $a(t), v(t) = \lambda(q(t), a(t))$ приведена на рис.1.41.

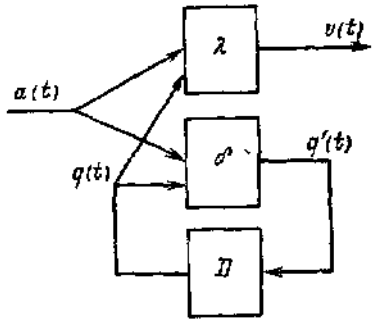


Рис. 1.41.

На этом рисунке блок λ — комбинационный автомат с входным алфавитом $A \times Q$ и выходным алфавитом V, блок δ — комбинационный автомат с тем же входным алфавитом и выходным алфавитом Q. Блок D — это блок, задерживающий поступающие в него сигналы на один такт (блок задержки). Он представляет собой автомат Мура, входной и выходной алфавиты которого совпадают и равны $Q = \{q_1, \ldots, q_n\}$, алфавит состояний $R = \{r_1, \ldots, r_n\}$ имеет ту же мощность, что и Q, $\lambda_D(r_i) = q_i$, $\delta_D(r_i, q_j) = r_j$ для всех $i, j = 1, \ldots, n$.

Частный случай *D* — двоичный элемент задержки. Сигнал *q′(t)* на выходе блока δ — это вычисленное значение δ*(q (t) a (t)),* которое, будучи задержано блоком *D* на такт, появится в момент $t + 1$ на входах блоков δ и λ, т. е. $q'(t) = \delta(q(t), a(t)) = q(t+1)$.

В важном частном случае, когда алфавиты *A, V, Q* состоят из двоичных наборов, блоки δ и λ — это логические комбинационные автоматы, двоичные выходы которых в момент *t* являются логическими функциями от двоичных переменных, образующих наборы *a (t)* и *q (t);* блок *D* — это параллельное соединение двоичных элементов задержки Число этих элементов равно *n* — длине вектора Q, а число состояний блока *D,* как и в общем случае, равно мощности его входного алфавита, т. е. $|Q| = 2^n$.

Поскольку любые конечные алфавиты *A, Q, V* можно закодировать двоичными кодами подходящей длины (т. е. установить взаимнооднозначное соответствие между элементами любого из этих алфавитов и двоичными наборами), получаем один из первых и фундаментальных результатов теории автоматов.

Теорема 7. Любой конечный автомат при любом двоичном кодировании его алфавитов *A, Q, V* может быть реализован синхронной сетью из логических комбинационных автоматов и двоичных задержек, причем число задержек не может быть меньше $\log_2 |Q|$.

В дальнейшем логические комбинационные автоматы для краткости будем называть логическими блоками (их входы и выходы двоичные), а блок задержки на один такт с одним двоичным входом и выходом — элементом задержки. Вход элемента задержки будем обозначать Y, а выход — *y*.

Сеть из логических блоков и элементов задержки (и те и другие будем называть блоками) называется правильно построенной логической сетью (ППЛС), если: 1) к каждому входу блока сети присоединен не более чем один выход блока сети (однако допускается присоединение выхода более чем к одному входу, т. е. разветвление выходов); 2) в каждом контуре обратной связи, т. е. в каждом цикле, образованном блоками и соединениями между ними, имеется по крайней мере один элемент задержки. Входами такой сети называются те входы блоков, к которым не присоединены никакие выходы; выходами сети называются те входы блоков, которые не присоединены ни к каким входам.

Сеть на рис. 1.41 превращается в ППЛС, если алфавиты *A, Q, V* закодировать двоичными наборами, число входов и выходов блоков δ, λ, *D* согласовать с длинами соответствующих наборов, а блок D

реализовать, как указано ранее, параллельным соединением двоичных задержек. Поэтому в формулировке теоремы 7 фактически подразумевается представление автомата синхронной ППЛС (СЛС) и следующую теорему можно считать обратной теореме 7.

Теорема 8. Всякая СЛС со входами $x_1, ..., x_m$, выходами $z_1, ..., z_n$ и k элементами задержки является конечным автоматом, входной алфавит которого состоит из 2^m двоичных наборов длины m, выходной алфавит — из 2^n наборов длины n, множество состояний — из 2^R наборов длины k.

Начнем с рассмотрения СЛС без задержек. Такая сеть по определению не может иметь циклов. Рассмотрим любой ее выход z.

Блок, которому он принадлежит, реализует на этом выходе логическую функцию $z = f^1(p_1^1, \ldots, p_r^1)$, где $p_1^1, \ldots, p_r^1$ — входы блока. Каждый из них либо является входом сети, т. е. переменной x, либо присоединен к выходу другого блока: $p_i = f_i^1(p_{i1}^2, \ldots, p_{im_i}^2)$ и, следовательно, $z = f^1(f_1^1(p_{11}^2, \ldots, p_{1m_1}^2), \ldots, f_r^2(p_{r1}^2, \ldots, p_{rm_r}^2))$, где вместо некоторых f_j^2, возможно, стоят переменные x. Еще одно повторение этой процедуры даст суперпозицию глубины три с функциями f_j^3 и переменными p_i^3 и т. д., причем верхний индекс переменной p_i^l равен числу блоков между узлом p_i^l и выходом z. Поскольку сеть ациклическая, этот процесс закончится и все переменные p_i^l будут заменены переменными x. Отсюда (с учетом отсутствия задержек в сети) видно, что z (t) является логической функцией от некоторых из $x_1(t), \ldots, x_m(t)$ и, следовательно, ППЛС без задержек всегда является логическим комбинационным автоматом.

Пусть теперь дана произвольная СЛС G. Если из нее удалить элементы задержки, то получим ациклическую сеть G_o без задержек, которая, как только что было показано, является логическим комбинационным автоматом. Входами G_o служат, во-первых, входы G, а во-вторых, выходы $y_1, ..., y_k$ элементов задержки G; выходы G_o — это выходы G и входы $Y_1, ..., Y_k$ элементов задержки G (рис.1.42).

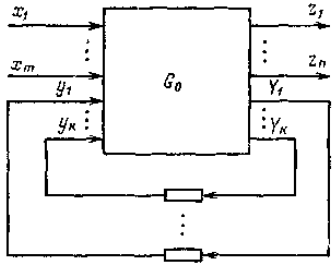


Рис. 1.42

Поэтому входной набор G_o имеет вид $(x_1, \ldots, x_m, y_1, \ldots, y_k)$, выходной набор — $(z_1, \ldots, z_n, Y_1, \ldots, Y_k)$. Если теперь набор $(x_1(t), \ldots, x_m(t))$ считать входным сигналом $a(t)$ сети G, набор $(z_1(t), \ldots, z_n(t))$ — выходным сигналом $v(t)$ сети G, а набор $(y_1(t), \ldots, y_k(t))$ — состоянием $q(t)$ сети G и учесть, что $(Y_1(t), \ldots, Y_k(t)) = (y_1(t+1), \ldots, y_k(t+1)) = q(t+1)$, то получим, что сеть G_0 вычисляет две системы логических функций от набора $(x_1, \ldots, x_m, y_1, \ldots, y_k) = a(t) \times q(t)$ — систему $(Y_1(t), \ldots, Y_k(t)) = q(t+1)$, т. е. функцию δ, и систему $z_1(t), \ldots, z_n(t)$, т. е функцию λ (эти две системы называются *каноническими уравнениями* сети G).

Сеть G в целом принимает вид рис.1.42, где блоки δ и λ образуют логический блок G_o (но при этом δ и λ — не обязательно отдельные блоки и могут иметь общие части), а блок задержки *D* состоит из *k* двоичных задержек. Это и дает автоматное описание СЛС G.

Для ациклических СЛС с задержками справедливы следующие два утверждения: 1) любая ациклическая ППЛС реализует определенный автомат (см. пример 4); 2) любой определенный автомат можно реализовать ацикли ческой СЛС с задержками. Доказательство этих утверждений не очень сложно и предоставляется читателю.

Пример 5. Для иллюстрации метода теоремы 8 приведем сеть на рис.1.43, которая содержит три логических блока (один из них — с тремя входами и двумя выходами, реализующими функции p_1p_2 и $p_1 \vee \bar{p}_3$ от входов p_1, p_2, p_3 блока) и две задержки.

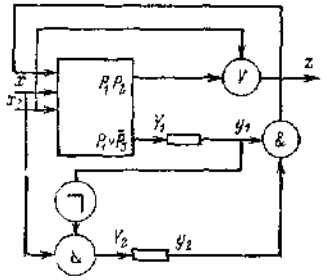


Рис.1.43.

Канонические уравнения сети:

$Y_1 = \bar{x}_2 \vee y_1y_2$, $Y_2 = x_1\bar{y}_1$, $z = x_1y_1y_2 \vee x_2$.

Таблица переходов автомата, реализуемого этой сетью, приведена в табл.1.11.

Таблица 1.11

y_1y_2	x_1x_2			
	00	01	10	11
00	10,0	00,1	11,0	01,1
01	10,0	00,1	11,0	01,1
10	10,0	00,1	10,0	00,1
11	10,0	10,1	10,1	10,1

Заметим, что первые два состояния этого автомата эквивалентны. СЛС с единичными задержками и конечные автоматы — адекватные друг другу описания, в том смысле, что параметры *m, n, k* СЛС определяют соответствующие параметры $|A|$, $|V|$, $|Q|$ автомата, и наоборот. Если же попытаться обобщить понятие задержки и считать, что задержки могут быть не только единичными, то связь между *k* и $|Q|$ становится менее очевидной. Рассмотрим, например, сеть (см. рис. 1.38, а), где S_1 и S_2—двоичные элементы задержки с величинами τ_1 и τ_2 (τ_1 и τ_2— произвольные натуральные числа). Со структурной точки зрения сложность этой сети всегда одна и та же: она определяется числом элементов и соединений между ними и не зависит от значений τ_1 и τ_2. Сложность же соответствующего автоматного описания существенно зависит от τ_1 и τ_2. Дело в том, что значения выходов y_1, y_2 сети в момент *t* определяются не только значениями входов Y_1, Y_2 в момент t и выходов в момент t — 1; если $y(t) = 0$, а $Y(t) = 1$, то для вычисления $y(t + 1)$ нужно знать, сколько тактов назад *V* изменился на единицу. Автоматное описание можно получить, разбив каждую задержку на единичные. Полученный автомат будет иметь $\tau_1 + \tau_2$ единичных задержек и $2^{\tau_1 + \tau_2}$ состояний, откуда видно, что сложность автоматного описания простых сетей с различными целочисленными задержками может быть сколь угодно большой. Поэтому для описания синхронных сетей с нетривиальными временными зависимостями применяются методы, не использующие понятие абстрактного автомата.

Асинхронные логические сети. Естественное стремление допустить в качестве значений задержек произвольные действительные числа для синхронных сетей бессмысленно, так как в них время по определению целочисленно; все, что происходит между соседними моментами *t* и $t + 1$, синхронной сетью не воспринимается. Однако синхронный способ описания взаимодействия автоматов во времени — не единственный, а часто — и не самый практичный. Это связано с тем, что

единая синхронизация входных, промежуточных и выходных сигналов требует введения внешних по отношению к исходному автомату синхронизирующих часов (поскольку абсолютная точность собственных временных характеристик элементов физически не достижима). Кроме того, длительность входного сигнала не всегда можно контролировать, особенно в задачах управления. Поэюму часто используется другой способ введения времени в автоматные сети, называемый асинхронным.

Он свободен от указанных недостатков, но имеет взамен них свои собственные, о которых будет сказано далее

Основная особенность асинхронного времени в том, что оно непрерывно. Когда речь идет о моменте времени *t,* то *t* может принимать произвольные неотрицательные действительные значения. Будем рассматривать функционирующие в этом времени двоичные сети, где по каждому каналу идут только двоичные сигналы 0 и 1 (из предыдущего видно, что это не нарушает общности), построенные из двух типов блоков: логических блоков с произвольным числом входов и выходов и элементов задержки с одним двоичным входом Y и одним двоичным выходом *y.* Сигналы в этих блоках поступают и выдаются непрерывно, т. е. в любой точке *t* шкалы непрерывного времени любой вход и выход любого блока сети имеет одно из двух значений. Логические блоки работают без задержки: значение выхода блока в момент *t* есть логическая функция от значений его входов в момент *t.* Каждому элементу задержки приписано положительное действительное число τ (время задержки). Имеются два основных типа задержки:

1) линейная задержка

$$y(t)=\begin{cases} 0, & \text{если } t<\tau, \\ Y(t-\tau), & \text{если } t\geqslant\tau; \end{cases}$$

2) задержка-фильтр

а) $y(t) = 0$, если $t < 0$;

б) y изменяется в момент $t + \tau$ с 0 на 1 (с 1 на 0) в том и только в том случае, если в момент t Y изменяется с 0 на 1 (с 1 на 0 соответственно) и на отрезке от t до $t + \tau$ не меняется.

Линейные задержки пропускают любые сигналы, поступающие на их вход; фильтры — только достаточно длинные (не короче τ) сигналы. В дальнейшем будут рассматриваться только задержки-фильтры.

Сеть, построенная из логических блоков и элементов задержки с соблюдением тех же двух правил, что и для СЛС, а именно: 1) к каждому входу блока присоединяется не более чем один выход, 2) каждый цикл должен содержать элемент задержки, — называется правильно построенной *асинхронной логической сетью* (в дальнейшем — АЛС).

Входной сигнал АЛС в момент t — это набор $(x_1(t), \ldots, x_m(t))$ значений ее входов в момент t. Будем считать, что на моменты изменения входных сигналов наложено условие дискретности: на каждом конечном отрезке времени этих моментов изменения — конечное число. Это условие позволяет рассматривать непрерывный по времени поток входных сигналов как дискретную последовательность следующим образом. Отрезок времени между двумя соседними моментами изменения входного набора (для того чтобы соседние моменты всегда существовали, и пришлось ввести условия дискретности) называется *входным тактом*. Последовательность входных наборов АЛС называется *входной последовательностью* для данной АЛС, если двум соседним в этой последовательности наборам соответствуют два соседних по времени такта. Например, совокупности трех двоичных сигналов на рис. 1.44 (такие рисунки называют временными диаграммами) соответствует последовательность 000,010, 110, 111, 110, 101, 001.

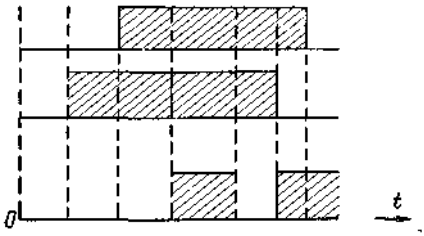


Рис.1.44.

Пунктиром отмечены границы тактов. Как видно, при таком определении длины тактов не играют роли, и разным временным диаграммам могут соответствовать одинаковые последовательности. Характерной особенностью асинхронных последовательностей является то, что соседние символы в них различны.

Точно так же вводится понятие выходного такта и выходной последовательности

Состоянием АЛС в момент t называема набор $(y_1(t), \ldots, y_k(t))$ значений выходов элементов задержки. Аналогично прежним определениям вводится понятие *внутреннею такта* — отрезка между соседними изменениями состояний — и понятие *последовательности состояний*. Таким образом, при асинхронном способе введения времени возникают два типа переменных: *непрерывная переменная t* (абсолютное время) и *целочисленные переменные* — номера входных, выходных и внутренних тактов. Длительности этих тактов не совпадают; задача их согласования и автоматной интерпретации поведения АЛС оказы-

вается более сложной, чем в синхронном случае. Этой интерпретацией мы и займемся.

Начнем с автономных АЛС, т.е. АЛС без входов. Автономная СЛС с k единичными элементами задержки реализует автономный автомат с 2^k состояниями. Поэтому последовательность ее состояний является периодической последовательностью с периодом, по длине не превосходящим 2^k (см. теорему 1.1). Для автономных АЛС это утверждение неверно, что видно уже из предварительных соображений об СЛС с различными целочисленными задержками. Сама процедура вычисления последовательности состояний автономной АЛС оказывается более сложной, чем для случая СЛС: она учитывает, сколько времени осталось каждой задержке до срабатывания. Поэтому следующее состояние определяется не только предыдущим состоянием $q(t)$ и каноническими уравнениями сети, но и вектором $r(t)$, образованным этими остатками времени. Число различных значений $r_i(t)$ — i- й компоненты $r(t)$ — оценивается сверху числом различных линейных форм от $\tau_1, \ldots, \tau_k$, т.е. выражений вида $\Sigma c_j \tau_j$; при этом c_j— произвольные целые числа, а $\Sigma c_j \tau_j \leqslant \tau_i$. Если времена задержек несоизмеримы (т. е. какая-нибудь из дробей τ_i/τ_j иррациональна), то это число линейных форм бесконечно, и последовательность состояний может оказаться непериодической. Например, автономная АЛС, приведенная на рис.1.45, имеет непериодическую последовательность состояний, если дробь τ_1/τ_2 иррациональна (можно показать, что эта последовательность определенным образом связана с разложением τ_1/τ_2 в цепную дробь).

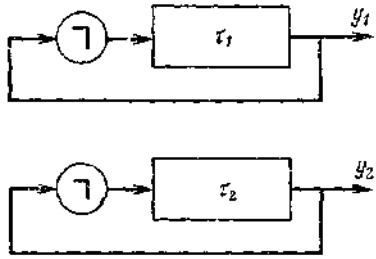


Рис.1.45.

Такая АЛС не может быть описана никаким конечным автоматом С другой стороны, ясно, что если все задержки автономной АЛС одинаковы, то описание ее поведения почти ничем не будет отличаться от описания поведения автономных СЛС. Единственное — правда, существенное — отличие таких АЛС от СЛС заключается в том, что

синхронному случаю бесконечной периодической последовательности состояний с периодом 1 в АЛС соответствует конечная последовательность, так как бесконечно повторяющемуся состоянию в АЛС соответствует один бесконечный такт. Состояние АЛС в этом такге называется устойчивым. Нетрудно убедиться, что состоящие автономной АЛС в момент *t* устойчиво, если и только если $y_i(t) = Y_i(t)$ для всех $i = 1, ..., k$.

Наконец, возможен случай, когда последовательность состояний автономной АЛС бесконечна, а выходная последовательность конечна. Это происходит тогда, когда всем состояниям последовательности, начиная с некоторого, соответствует один и тот же выход.

Рассмотрим теперь АЛС со входами. На протяжении одного входного такта АЛС ведет себя как автономная АЛС, канонические уравнения которой получаются из канонических уравнений исходной АЛС подстановкой констант вместо входных переменных. Например, если сеть (см рис.1.43) рассматривать как АЛС, то при входном сигнале 11 она становится автономной АЛС с уравнениями $Y_1 = y_1 y_2$; $Y_2 = \bar{y}_1$, $z = 1$, а при входе 10 ее уравнения имеют вид $Y_1 = 1$,; $Y_2 = \bar{y}_1$, $z = y_1 y_2$. Так как длина входного такта может быть сколь угодно большой, то на его протяжении АЛС может генерировать некоторую последовательность состояний (называемую *переходным процессом)* и выходную последовательность; в них длины тактов определяются только соотношениями задержек АЛС. Входной набор и значения $y_1, ..., y_k$ в начале входного такта называются *начальными условиями переходного процесса.*

В зависимости от поведения АЛС на протяжении одного входного такта возможны три основных случая.

1. Для некоторых входных наборов возникающая автономная АЛС порождает бесконечную (точнее, она была бы бесконечной при бесконечном входном такте) выходную последовательность, которая оборвется после наступления следующего входного такта, причем место ее обрыва будет зависеть от момента наступления этого такта. Поэтому общая выходная последовательность такой АЛС зависит не только от входной последовательности, но и от длин ее тактов; следовательно, такая АЛС не осуществляет однозначного отображения входных слов в выходные слова, и ее поведение нельзя интерпретировать в автоматных терминах.

2. Для любого входного набора a_j и любого состояния q_i, устойчивого к моменту подачи сигнала a_j , переходный процесс с начальными условиями (q_i, a_j) является конечным, т. е заканчивается в устойчивом состоянии q_l .Тогда и выходная последовательность в данном входном такте заканчивается устойчивым выходом *v*. Длительности (точнее,

времена перехода в устойчивое состояние) $T_q(q_i, a_j)$ и $T_m(q_i, a_j)$ этих последовательностей определяются логикой и задержками АЛС, при чем $T_m \leq T_q$. В этом случае можно сказать, чго для любой входной последовательности, такты которой не меньше $T = \max_{i,j} T_q(q_i, a_j)$, однозначно определены функция $\delta(q_i, a_j)$, равная устойчивому состоянию q_l, которым заканчивается переходный процесс с начальными условиями (q_i, a_j), и функция $\lambda(q_i, a_j)$,равная устойчивому выходу v. Таким образом, поведение АЛС в этом случае интерпретируется как поведение конечного автомага (с точностью до переходного процесса, который либо игнорируется, либо с помощью дополнительных фильтров пропускается на выход). Заметим, что конечный автомат, получающийся при такой интерпретации, всегда является асинхронным в смысле примера 1.9. Верно и обратное — для любого асинхронного автомата можно построить реализующую его АЛС

3. Для любого входного такта достигается устойчивый выход, однако переходный процесс бесконечен. АЛС с таким свойивом можно охарактеризоватъ так, что они будут реализовать однозначные отображения входных последовательностей в выходные. Эти отображения удовлетворяюг условиям автоматности , но могут не быть конечно-автоматными. Показано, что если в АЛС добавить еще два типа элементов, которые могут реагировать на сколь угодно короткие входные сигналы, то такие расширенные АЛС могут представлять нерегулярные события, более конкретно, распознавать непериодические входные последовательности. В констукции такой АЛС существенным образом используется несоизмеримость ее задержек. Ясно, что физически несоизмеримость задержек нереализуема, поэтому данный рсчультат можно интерпретировать как предельную формулировку следующею утверждения: «АЛС с фиксированным числом состояний может распознавать периодическую последовательность со сколь угодно большим периодом».

На практике наибольший интерес представляет случай 2, причем АЛС считается тем более надежной. чем больше область ее устойчивости, т. е. область допустимых отклонений времен задержек от заданных величин, не изменяющих переходный процесс. (Изменения переходных процессов в результате отклонения задержек от заданных величин называются состязаниями, или гонками.)

Синтез автоматных сетей. Рассмотренные ранее методы получения единого автоматного описания для сетей из автоматов называются методами *композиции* автоматов. Методы композиции решают задачу анализа сетей, поскольку исследование поведения сетей и, в частности,

реализуемого сетью отображения, удобно проводить при наличии автоматного описания: таблицы переходов, графа переходов или системы канонических уравнений. Обратная задача — построение сети по заданному автомату — называется задачей *декомпозиции* или разложения автоматов. Возможны различные задачи декомпозиции в зависимости от того, какие условия накладываются на искомую автоматную сеть. Например, существуют методы разложения автомата на сеть из автоматов, соединенных заданным образом — последовательно, параллельно и т. д.

Наиболее практически важной и хорошо изученной задачей декомпозиции является задача реализации автомата в заданном автоматном базисе, т. е. задача построения сети, реализующей данный автомат, из заданного набора автоматов (автоматного базиса), называемых в этом случае *элементарными автоматами*, или просто элементами. Эта задача обычно называется задачей *структурного синтеза* или структурной реализации автоматов, а получаемые сети — *семами из функциональных элементов.* Из проблем, которые здесь возникают, кратко коснемся двух основных: 1) любой ли автомат S можно реализовать схемой из множества элементов Σ (если да, то Σ называется автоматно-полным набором элементов); 2) среди всех схем в базисе Σ, реализующих S, найти минимальную схему. Начнем со структурного синтеза логических комбинационных автоматов, т. е. со схемной реализации систем логических функций. Если функция f задана формулой F над множеством функций Σ, то формуле F можно поставить в соответствие
схему G из комбинационных автоматов, реализующих функции Σ. Построение схемы G определяется индукцией по глубине формулы: 1) если $F = \varphi(x_{i_1}, \ldots, x_{i_k})$, где $\varphi \in \Sigma$, $x_{i_1}, \ldots, x_{i_k}$ — исходные переменные, то схема G состоит из одногоэлемента φ, входы которого отождествлены с переменными $x_{i_1}, \ldots, x_{i_k}$, а выход — с переменной z; 2) если $F = \varphi(F_1, \ldots, F_k)$, где F_i — переменная x_{ji} или функция, уже реализованная схемой G_i, то схема G для F строится так: к i-му входу элемента φ присоединяется выход схемы G_i (если F_i— функция) или переменная (если $F_i = x_{ji}$); выходом z схемы G объявляется выход элемента φ. Например, формуле $x_1\overline{x_2x_3} \vee x_2x_3x_4$ соответствует схема на рис1.46, *а.*

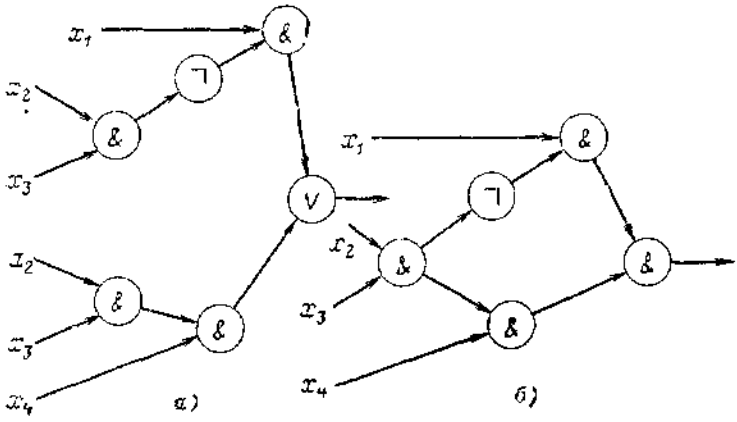


Рис.1.46.

Схема, полученная описанным методом, всегда имеет вид ориентированного дерева; ее входы соответствуют концевым вершинам, а выход—корню дерева. Между множеством формул над Σ и множеством древовидных схем из элементов, реализующих функции Σ, существует взаимно-однозначное соответствие: по любой формуле *F* над *G* описанный метод однозначно строит схему G, с другой стороны, анализ схемы *G*, приведенный методом теоремы 1.8, даст исходную формулу *F* (второй факт и дает основание утверждать, что *G* действительно реализует *F).* Число знаков операций в *F* равно числу элементов в *G*. Это обстоятельство сводит задачу преобразования (и, в частности, минимизации) древовидных схем к задаче преобразования логических формул.

Итак, по формуле над Σ всегда можно построить древовидную схему из элементов Σ, обратное утверждение в силу теоремы 1.8 верно для произвольных (а не только древовидных) схем над Σ. Отсюда получаем следующий важный факт: *для того чтобы произвольная логическая функция могла быть реализована схемой над* Σ*, необходимо и достаточно, чтобы множество функций* Σ *было функционально полным* . Ясно, что для реализации системы функций ответ в точности тот же.

Вторая задача — задача минимизации — оказывается гораздо более сложной. Проблема минимизации формул сама по себе довольно трудна, однако она не исчерпывает всех возможностей минимизации схем. Например, схема на рис.1.46, *б* реализует ту же формулу, что и

рис.1.46, *а,* однако схема *б* не древовидная и имеет меньше элементов, чем любая формула в булевом базисе.

До настоящего времени известно очень небольшое количество классов функций, для которых найдены минимальные схемные реализации. Исследования в этой области дают — пока косвенные — свидетельства того, что в общем случае поиск минимальных схемных решений невозможен без большого перебора и практически не реализуем; достаточно точно оценить по данной функции хотя бы число элементов в минимальной схеме — не проводя синтеза — также не удается. Поэтому теоретические исследования задачи синтеза пошли по пути глобальных характеристик сложности схем. Основной результат здесь формулируется следующим, образом.

Пусть $L_\Sigma(f)$— число элементов минимальной схемы в базисе Σ, реализующей функцию *f* Обозначим $L_\Sigma(n) = \max_{f \in P_2(n)} L_\Sigma(f)$, где максимум берется по всем функциям от *n* переменных. Функция $L_\Sigma(n)$ называется *функцией Шеннона* для базиса Σ, она равна минимальному числу элементов из Σ, достаточному для реализации любой функции от *n* переменных

Теорема 1.9 (теорема Шеннона — Лупанова). Для любого базиса

$$L_\Sigma(n) \sim c_\Sigma \frac{2^n}{n},$$

где c_Σ— константа, зависящая от базиса, причем для любого $\varepsilon > 0$ доля функций *f,* для которых $L_\Sigma(f) \leqslant (1-\varepsilon)\, c_\Sigma \frac{2^n}{n}$, стремится к нулю с ростом *n*. Здесь знак ~ обозначает асимптотическое равенство $\left(a(n) \sim b(n), \text{ если } \lim_{n\to\infty} \frac{a(n)}{b(n)} = 1\right)$. Смысл второго утверждения теоремы в том, что с ростом *n* почти все функции реализуются со сложностью, близкой к верхней границе, т. е. *L (n)*

Для автоматов общего вида уже задача существования схемы в заданном наборе Σ оказывается довольно трудной. Правда, из обсуждения схемной реализации логических функций и теоремы 1.7 следует, что набор *К* элементов, состоящий из элемента единичной задержки и любого функционального полного набора логических элементов, является автоматно полным. Однако в общем случае проблема автоматной полноты для произвольного набора автоматов оказывается алгоритмически неразрешимой (в отаичие от проблемы функциональной полноты для логических функций).

Практические методы синтеза автоуатов разработаны в основном для набора К описанного ранее, либо для наборов, полученных из набора *К* заменой элемента задержки на какой-либо дрегой элементарный

автомат. Долгое время основной считалась каноническая схема синтеза Хаффмена— Глушкова, которая разбивает процесс структурной реализации автоматов на следующие этапы: 1) построение автомата по какому-либо описанию автоматного отображения (например, по регулярному событию); 2) минимизация числа состояний автомата; 3) двоичное кодирование состояний (а также входных и выходных алфавитов, если исходный автомат абстрактный, а не логический) и получение канонических уравнений будущей сети, описывающих блоки δ и λ как системы логических функций; 4) реализация системы канонических уравнений схемой в функционально полном базисе. Число элементов задержки определяется на этапе 3, число логических элементов — на этапе 4.

Эта схема сыграла большую роль в развитии методов синтеза автоматов. Однако, надежды на ее широкое практическое использование оказались сильно преувеличенными. Это объясняется в основном следующими причинами. Во-первых, как было установлено ранее, схемы с k элементами задержками имеют 2^k состояний, поэтому описание сколько-нибудь больших схем в терминах состояний н таблиц переходов оказывается довольно громоздким. Во-вторых, на разных этапах решаются разные задачи минимизации, которые плохо связаны между собой. Известны примеры, когда уменьшение числа состояний приводит к усложнению логики схемы. Кроме того, различные варианты кодирования (для k задержек существует $(2^k)!$ вариантов) приводят к разным системам канонических уравнений; предвидеть сложность их реализации заранее, на этапе кодирования, невозможно. Поэтому в практике получают все большее распространение методы, обходящие каноническую схему синтеза.

1.9. Программная реализация логических функций и автоматов

Представление автомата схемой из элементов — это исторически первый и наиболее исследованный вид структурной реализации автомата. Другой ее вид — **реализация автомата программой**. Здесь мы ограничимся вопросами программной реализации логических функций.

Под программой будем понимать пронумерованную последовательность команд $k_1, \ldots, k_s$, взятых из некоторого фиксированного набора (системы команд). Программа работает над конечным множеством пронумерованных (или проименованных) двоичных ячеек. Номер (или имя) ячейки называется ее адресом;

именем ячейки часто будет служить имя логической переменной, значения которой хранятся в этой ячейке.

Система команд содержит команды-операторы вида $b := f(a_1, \ldots, a_p)$ (выполнить операцию *f* над содержимыми ячеек $a_1, \ldots, a_p$ и результат положить в ячейку *b)* и двухадресные условные переходы двух видов: 1) «если *a,* то *i,* иначе *j*» (если *a* = 1, то перейти к выполнению команды k_i, иначе перейти к k_j) и 2) «если $\bar{a}$ $(a = 0)$, то *i,* иначе *j*».

Операция $f(a_1, \ldots, a_p)$ — это логическая функция *p* переменных; в частности, она может быть константой 0 или 1. Если $j=i+1$, то переход можно считать одноадресным: «если *a,* то *i* (иначе перейти к k_{i+1})», а если $i = j$, то безусловным: «перейти к *i*». Любая из команд указанных типов может быть заключительной, что указывается словом «конец».

Процессом вычисления программы $k_1, \ldots, k_s$ называется последовательность шагов $k(1), k(2), \ldots, k(t)$, на каждом из которых выполняется одна команда программы. Эта последовательность определяется так: 1) $k(1) = k_1$; 2) если $k(i) = k_r$ — оператор, то $k(i+1) = k_{r+1}$; 3) если $k(i)$ — условный переход, то номер команды $k(i+1)$ указывается этим переходом; 4) если $k(i)$ — заключительная команда, то процесс вычисления останавливается после ее выполнения. Число *t* называется временем вычисления.

Программа П вычисляет (или реализует) логическую функцию $f(x_1, \ldots, x_n) = y$, если для любого двоичного набора $\sigma = (\sigma_1, \ldots, \sigma_n)$ при начальном состоянии памяти $x_1 = \sigma_1, x_2 = \sigma_2, \ldots, x_n = \sigma_n$ (состояние остальных ячеек несущественно) программа через конечное число шагов останавливается и при этом в ячейке *y* лежит величина $f(\sigma_1, \ldots, \sigma_n)$.

Если под сложностью схемы, реализующей автомат, обычно понимается число элементов в схеме, то сложность программы можно понимать в различных смыслах: 1) число команд в тексте программы; 2) объем промежуточной памяти, т. е. число ячеек для хранения промежуточных результатов вычислений; 3) время вычисления, которое будет характеризоваться двумя величинами: средним временем $t_{ср}(\Pi) = \frac{1}{2^n}\sum_{\sigma} \tau_P(\sigma)$ и максимальным временем $t_{макс}(\Pi) = \max_{\sigma} \tau_P(\sigma)$, где сумма и максимум берутся по всем 2^n двоичным наборам σ, а τ_P — время работы программы П на наборе σ.

Любой схеме, реализующей функцию *f* и содержащей N элементов, нетрудно поставить в соответствие программу, реализующую *f* и

состоящую из N команд, следующим образом. Занумеруем элементы схемы числами 1, 2, ..., n так, чтобы на любом пути от входа к выходу номера элементов возрастали; при этом номер 1 получит один из входных элементов, а номер N — выходной элемент. Пусть элемент e_i схемы реализует функцию φ_i и к его входам присоединены выходы элементов $e_{i_1}, \ldots, e_{i_p}$ (некоторые из них, возможно, являются входами схемы). Поставим элементу e_i в соответствие либо ячейку a_i и команду $a_i := \varphi_i(a_{i_1}, \ldots, a_{i_p})$, если $i \neq N$; либо ячейку y и команду «$y := \varphi_N(a_{i_1}, \ldots, a_{i_p})$ конец», если $i = N$. Получим программу, не содержащую условных переходов (такие программы будем называть *операторными),* в которой порядок команд в точности соответствует нумерации элементов в схеме, а система команд—базису схемы. Проблемы синтеза операторных программ в основном сводятся к проблемам синтеза схем. в частности, вопросы функциональной полноты системы команд и минимизации собственного текста операторной программы совпадают соответственно с задачами о функциональной полноте системы функций и о минимизации схем. Поскольку операторная программа не содержит условных переходов, время ее вычисления на любом наборе одно и то же; отсюда $t_{макс}=t_{ср}=N$, т. е. оба вида временной сложности совпадают со сложностью текста программы и, следовательно, в силу теоремы 1.9 при достаточно больших n почти для всех функций близки к $2^n/n$. Наоборот, проблема минимизации памяти (за счет многократного использования одной ячейки для нескольких промежуточных результатов) для операторных программ является нетривиальной комбинаторной задачей.

Другой «крайний» вид программ, вычисляющих логические функции — это программы, состоящие из команд вида $y := \sigma$ ($\sigma = 0$ или $\sigma = 1$) и условных переходов. Такие программы называются *бинарными программами.* Всякую булеву формулу F, содержащую N букв, можно реализовать бинарной программой, вычисляющей F за время $t_{макс} = N$ и содержащей N команд условного перехода (а также две команды $y := 0$ и $y := 1$, которые в оценках не учитываются). Для описания метода реализации используем представление програлмы в виде графа (в котором вершины соответствуют командам, а ребра — переходам). Пусть G_1 — граф программы для функции f_1 с начальной вершиной v_{10} и двумя заключительными вершинами v''_{1z} (с командой «$y = 0$») и v^1_{1z} (с командой «$y == 1$»); G_2 — граф программы для f_2 с началом v_{20} и концами v^0_{2z} и v^1_{2z}. Тогда 1) программа, граф G которой получен присоединением G_2 к «нулю» G_1 (т. е. отождествлением вершин v^0_{1z} и v_{20}; команда «$y := 0$» при этом отбрасывается), вычисляет функцию

$f = f_1 \vee f_2$; 2) программа, граф G' которой получен присоединением G_2 к «единице» G_1 (отождествлением v^1_{1z} и v_{20}), вычисляет $f = f_1 \,\&\, f_2$; 3) программа, граф которой получен из G_1, заменой команд в v^2_{1z} и v^1_{1z} на инверсные («y: =0» на «y : = 1» и наоборот) вычисляет $\bar{f}_1$. Заметим, что в графе G получаются две единичные, а в графе G' — две нулевые заключительные вершины. В обоих случаях их надо отождествить.

Пример1.6. Формула $(x_1 \vee x_2)(\bar{x}_3 x_4 \vee x_5)$ реализуется бинарной программой, граф которой приведен на рис. 1.47.

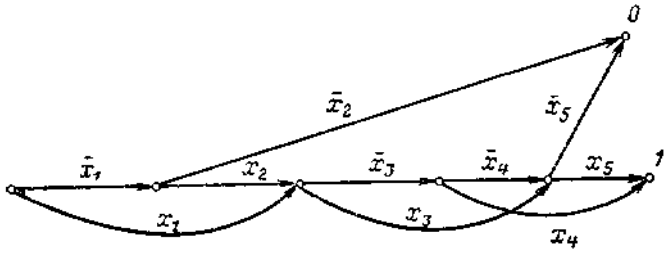


Рис.1.47.

Описанный метод не гарантирует минимальность получаемых программ по времени и числу команд. Существуют другие методы, которые, в частности, для любой функции n переменных дают бинарную программу с $t_{макс} \le n$ независимо от длины исходной формулы. В общем же случае сложность бинарных программ характеризуется следующими асимптотическими оценками. Пусть $L_Б(n)$ — функция Шеннона для числа команд бинарных программ.

$$L_Б(n) \sim \frac{2^n}{n},$$

Теорема1.11. 1) причем существует метод синтеза бинарных программ, удовлетворяющих этой оценке, для которых $t_{макс} \sim n$; 2) для любого $\varepsilon > 0$ доля функций, для которых а) $L_Б(f) \leqslant (1-\varepsilon)\frac{2^n}{n}$, б) $t_{ср}(f) \leqslant (1-\varepsilon)n$, стремится к 0 при $n \to \infty$.

Таким образом, сложность бинарных программ по числу команд асимптотически равна сложности операторных программ. Важным достоинством бинарных программ является отсутствие в них промежуточной памяти и малое время вычисления

Приведем еще несколько утверждений, относящихся к времени вычисления логических функций бинарными программами.

1. Для любой функции f, существенно зависящей от всех n переменных $[\log_2 n + 1] \leqslant t_{макс}(f) \leqslant n$, причем обе границы достигаются.

2. Существуют последовательности функций $f_1(x_1)$,

$f_2(x_1, x_2), \ldots, f_n(x_1, \ldots, x_n), \ldots$, для которых
$\lim_{n \to \infty} t_{cp}(f_n) = 2.$

иначе говоря, с ростом n $t_{cp}(f_n)$ практически не увеличивается. Примером такой последовательности является любая последовательность $f_1, \ldots, f_n, \ldots$, где $f_1 = x_1, \ldots$ $\ldots, f_{n+1} = x_{n+1} \bigcirc f_n$ ($\bigcirc$ — дизъюнкция или конъюнкция).

3. Для любой системы m логических функций n переменных существует бинарная программа, вычисляющая ее за время, не превосходящее $n + m$. (В отличие от оценок времени для вычисления одной функции здесь число команд вида $y := \sigma$ не меньше m и входит в оценку.)

Как уже отмечалось, число команд в бинарных и операторных программах асимптотически равно $2^n/n$. Если же в программе использовать и операторы, и условные переходы, то эта оценка понижается вдвое.

Мы отметим, что логический автомат всегда может быть реализован бинарной программой, не требующей промежуточной памяти.

2. Декомпозиция абстрактных автоматов

2.1. Постановка задачи декомпозиции автоматов

В этой главе рассматривается задача декомпозиции произвольных абстрактных автоматов. *Под декомпозицией в общем случае понимают представление сложного автомата работой нескольких более простых* (в *частности, элементарных) автоматов, которые на структурном уровне с помощью отождествления входных и выходных каналов образуют функциональную или структурную схему сложного автомата.* Обычно ставится задача *оптимальной* декомпозиции, которая заключается в отыскании *минимального числа элементарных автоматов,* образующих схему сложного автомата.

Как известно, при синтезе автоматов используют два класса элементарных автоматов: элементарные автоматы с памятью и элементарные автоматы без памяти — логические элементы. Поэтому **схема любого конечного автомата содержит две части: запоминающую часть, состоящую из элементарных автоматов с памятью, и комбинационную часть, образованную логическими элементами.** В результате оптимальной декомпозиции конечного автомата осуществляется *минимизация числа логических элементов,* входящих в комбинационную часть автомата.

На абстрактном уровне декомпозиция сложного абстрактного автомата соответствует параллельной, последовательной или смешанной работе более простых абстрактных автоматов, т. е. сводится к разложению абстрактного автомата по операции умножения, суммирования, суперпозиции или по нескольким операциям. Поэтому можно рассматривать несколько видов декомпозиции абстрактных автоматов: *параллельную, последовательную* и *смешанную* декомпозицию.

Параллельная декомпозиция соответствует разложению абстрактного автомата в произведение или сумму двух или большего числа абстрактных автоматов, каждый из которых проще, чем исходный автомат. В зависимости от разложения автомата по операции умножения или суммирования можно говорить о параллельной *одновременной* и о параллельной *неодновременной (поочередной)* декомпозиции абстрактных автоматов. *Последовательная* декомпозиция соответствует разложению автомата по операции суперпозиции, а *смешанная* — одновременно по двум операциям, например, умножения и суперпозиции и т. п.

Все перечисленные случаи декомпозиции абстрактных автоматов представляют собой так называемые «чистые» случаи декомпозиции автоматов. Таких автоматов, которые раскладываются только в параллельную или в последовательную или даже в смешанную работу автоматов, незначительное число по сравнению с множеством автоматов, которые не представимы параллельной, последовательной или смешанной декомпозицией. В связи с этим вводится понятие *общей* декомпозиции абстрактного автомата, которая понимается как представление абстрактного автомата совместной работой элементарных абстрактных автоматов со связями между ними. Общая декомпозиция соответствует разложению абстрактного автомата в композицию двух или большего числа абстрактных автоматов.

Заметим, что в качестве элементарных абстрактных автоматов могут быть выбраны абстрактные автоматы с любым числом состояний, а параллельную, последовательную и смешанную декомпозиции абстрактных автоматов можно рассматривать как частные случаи общей декомпозиции автоматов.

Впервые поставил и решил задачу параллельной и последовательной декомпозиции Хартманис, который ввел специальный язык парной алгебры на множестве состояний абстрактного автомата. Впоследствии появляется большое число работ, продолжающих и развивающих методы декомпозиции абстрактных автоматов, к которым относятся работы Хартманиса и Стёрнза, Иоли, Кохави и др.

Задачи параллельной и последовательной декомпозиции абстрактных автоматов можно сформулировать как задачи разложения абстрактных автоматов по операциям умножения, суммирования и суперпозиции автоматов по аналогии с графами. Поэтому изучение свойств формальных операций над абстрактными автоматами, задаваемых на языке теории графов, имеет большое значение в теории автоматов.

В связи с бурным развитием микроэлектроники и однородных вычислительных структур, возникла задача *декомпозиции* сложного автомата на *заданные блоки (автоматы)*. В общем случае требуется, если это возможно, выделить из сложного автомата заданный автомат так, чтобы он с оставшейся частью сложного автомата работал последовательно (справа или слева) либо параллельно одновременно, либо параллельно поочередно и т. п. Использование алгебраических операций позволяет на абстрактном уровне решать задачу декомпозиции сложного автомата на заданные стандартные автоматы путем сведения ее к формальному решению суперпозиционных уравнений над автоматами.

Ниже будет показано, что любой абстрактный автомат может быть построен из элементарных абстрактных автоматов, в качестве которых можно выбрать автоматы с любым числом состояний, используя две формальные операции — **умножение автоматов и запреты переходов**. С содержательной точки зрения это означает, что произвольный автомат может быть представлен совместной работой более проешх автоматов с некоторым числом связей между ними, при помощи которых реализуются запреты переходов. Приводится алгоритм декомпозиции произвольного абстрактного автомата на элементарные абстрактные автоматы, который соответствует представлению произвольного автомата композицией элементарных абстрактных автоматов. Решается задача оптимальной декомпозиции, при которой число связей между элементарными автоматами, представляющими работу произвольного автомата, минимально. Это приводит к минимальным функциям возбуждения элементарных автоматов и, следовательно, к упрощению комбинационной части автомата.

Предлагаемая методика абстрактной декомпозиции автоматов позволяет решить задачу размещения состояний автомата на абстрактном уровне и свести этап структурного синтеза к непосредственной записи функций возбуждения и функций выходов по матрицам соединений элементарных абстрактных автоматов. Поэтому оптимальная декомпозиция автомата на элементарные абстрактные автоматы, приводящая к декомпозиционному методу синтеза, решает

задачу структурного синтеза автоматов, которая рассматривается в следующей главе, на абстрактном этапе.

2.2. Введение в теорию декомпозиции автоматов

Напомним определение абстрактного автомата. **Абстрактный автомат** — это математическая идеализация реального объекта или явления, реагирующего на различные входные возмущения. При этом мы считаем, что входной сигнал строго определен как функция времени, а внутренние обратные связи, имеющиеся в электронных схемах, не учитываем. Различаются два типа элементарных схем, к первому типу относятся комбинаторные логические схемы без памяти (они декодируют входные сигналы в выходные), ко второму— схемы с памятью, они-то и связаны с полугруппами.

Автомат имеет конечное множество состояний $Q = \{q_1, q_2, ..., q_h\}$, конечный входной алфавит $X = \{a_1, a_2, ..., a_m\}$ и конечный выходной алфавит $Y = \{\alpha_1, \alpha_2, ..., \alpha_n\}$. Действие автомата определяется функцией переходов $\delta : Q \times X \to Q$ и функцией выходов $\lambda : Q \times Y \to Y$

Способы задания автомата. Функция переходов и функция выходов могут быть заданы следующими способами:

1) с помощью графа. Каждая вершина этого графа соответствует некоторому состоянию. Вершины соединены ориентированными ребрами, каждому такому ребру приписана пара элементов входного и выходного алфавита, показывающая, что из состояния в начале стрелки автомат переходит в состояние в конце стрелки при данном входе, а из состояния в начале стрелки при данном входе попадает в данный выход. Если две вершины соединены несколькими ориентированными ребрами, то это означает, что для каждой из них имеется свой вход, переводящий одно состояние в другое;

2) с помощью таблицы переходов и таблицы выходов;

3) с помощью булевых матриц и выходных векторов. Этот метод может применяться только для описанных далее автоматов типа состояние—выход. Каждому элементу входного алфавита соответствует матрица. Строки и столбцы каждой матрицы занумерованы элементами множества состояний. Если данный входной элемент вызывает переход автомата из состояния i в состояние j, то в соответствующей этому элементу матрице в пересечении i-й строки и j -го столбца стоит 1, а в остальных местах i-й строки стоят нули. Задан вектор-строка, а именно вектор начального состояния, его элементы занумерованы состояниями автомата. Если автомат начинает работать из состояния i, то i-й элемент вектора начального состояния равен 1, а

остальные—нулю. Задан выходной вектор-столбец, его элементы занумерованы состояниями автомата, если состоянию i соответствует выходной элемент α_i, то i-я координата выходного вектора равна α_i. Для того чтобы определить состояние, в которое попадает автомат при поступлении заданной иходной последовательности, надо умножить вектор начального состояния последовательно на матрицы соответствующих входных элементов, в результате получится вектор-строка, одна координата которого—единица, а остальные—нули, номер единичной координаты есть искомое состояние. Для того чтобы найти результирующий выход автомата, надо умножить полученный вектор-строку на выходной вектор-столбец. В результате получится искомый выходной элемент;

4) с помощью функций, отображающих всевозможные входные последовательности в элементы выходного алфавита при каждом заданном начальном состоянии. Для описания внешнего поведения автомата нет необходимости задавать функцию, переводящую всевозможные входные последовательности в соответствующие последовательности выходных элементов. Вполне достаточно функции, переводящей различные входные последовательности в результирующие выходные элементы. Для начального состояния i имеем функцию f_i: $\Sigma X \to Y$, где ΣX—множество всевозможных конечных последовательностей над алфавитом X. При таком описании функция переходов не задана, однако далее будет показано, что существует функция переходов, которая может быть, естественно, связана с введенными функциями, отображающими входные последовательности в выходные элементы. На практике это наиболее удобный способ представления автомата. По этой причине Крон и Роудз определяют автомат как функцию $\Sigma X \to Y$. На рис. 1 изображены всевозможные описания действия автомата.

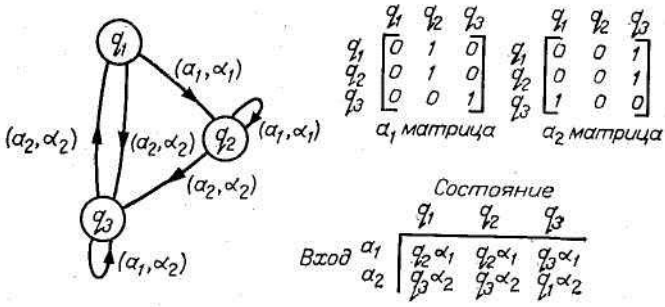


Рис. 1. Пример представления функции переходов графом, таблицей и матрицей.

1 Автоматы типа состояние—выход и автоматы с функцией выходов общего вида. Мы рассмотрели автоматы с функцией выхода общего вида, или автоматы Мили. Их выход зависит как от текущего состояния, так и от поступающего входного сигнала. Частный случай автоматов Мили представлен автоматами, выход которых зависит только от того состояния, в которое они попадают. Следовательно, каждому состоянию такого автомата соответствует вполне определенный выходной элемент, при этом, конечно, разным состояниям может соответствовать один выходной элемент. При заданном отображении множества состояний в выходной алфавит выходной элемент автомата может быть найден с помощью функции переходов. Такие специальные автоматы называются автоматами Мура, или автоматами типа состояние—выход.

Для заданного автомата с функцией выходов общего вида всегда можно построить имитирующий его внешнее поведение автомат типа состояние—выход. Пусть имеется автомат общего вида с множеством состояний Q и выходным алфавитом Y. Построим новый автомат типа состояние—выход с множеством состояний $Q^1 = \{(q_i, \alpha_j) : q_i \in Q, \alpha_j \in Y\}$. Новая функция переходов δ' опоелеляется с помощью соотношения $\delta'[(q_i, \alpha_k), a_j] = [\delta(q_i, a_j), \lambda(q_i, a_j)]$, новая функция выходов λ' — с помощью соотношения $\lambda'[(q_i, \alpha_k), a_j] = \alpha_k$.

Состояния нового автомата, недостижимые из других состояний, можно отбросить. Состояния первоначального автомата, недостижимые из других состояний, приводят в новом автомате к недостижимым состояниям. Построенный автомат определен корректно.

В автомате типа состояние—выход отличие функции переходов от функции выходов проявляется более ярко. Достаточно просто интерпретируется функционирование автомата типа состояние—выход во времени: входной сигнал, поступающий в некоторый момент времени вызывает в последующий момент выходной сигнал.

2. Следствия из специального описания конечного автомата. Из приведенных в разделе 2.1 описаний работы конечного автомата вытекают следствия:

1. Множество функций типа вход—выход замкнуто относительно левых сдвигов, порожденных входными последовательностями. Рассмотрим функцию типа вход—выход для некоторого автомата, находящегося в начальном состоянии q_i. Выберем входную последовательность x, имеющую вид $x = x_1x_2$, где x_1 и x_2 — также входные последовательности. Пусть на вход автомата с начальным состоянием q_i по-

дается сперва только входная последовательность x_1. Автомат должен тогда перейти в некоторое состояние q_j. Представим это состояние как начальное и подадим входную последовательность x_2. Получающееся в результате выходное значение совпадает с выходным значением, определяемым входной последовательностью x при начальном состоянии q_i, следовательно, $f_j(x_2) = f_i(x_1x_2)$. Последнее равенство справедливо при любых значениях x_2 и поэтому можно построить функцию типа вход— выход для состояния q_j с помощью соотношения $f_j(x) = f_i(x_1x)$. Запишем функцию f_j как $f_i L_{x_1}$, где L_{x_1}— оператор левого сдвига $L_x(x') = xx'$. Выбирая всевозможные последовательности для x, построим конечное множество функций $f_i L_x$. Каждая новая функция, найденная с помощью левого сдвига, соответствует некоторому новому состоянию автомата. В том случае, когда множество функций замкнуто относительно левых сдвигов, получается в точности автомат с минимальным числом состояний, и его функция переходов в новое состояние получается посредством левых сдвигов на входные последовательности, состоящие из одного входного элемента. В принципе эта конструкция решает задачу нахождения автомата с минимальным множеством состояний, реализующего заданное внешнее поведение. Однако на практике обычно задается только некоторая часть этой функции, определяемая небольшим подмножеством пар «входная последовательность— выходной сигнал», остальной частью множества этих пар пренебрегают. Задача восстановления всей функции для того, чтобы можно было минимизировать множество состояний, пока еще не решена.

2. Конечный автомат определяет правоинвариантное конечное отношение эквивалентности на множестве входных последовательностей. Пусть для заданного автомата фиксировано начальное состояние q_i. Для каждой входной последовательности отметим состояние, в которое эта последовательность переводит q_i. Входные последовательности, для которых соответствующие состояния одинаковы, будем считать эквивалентными. Если последовательности x и y эквивалентны, то для любой входной последовательности z $\delta(q_i, xz) = \delta(q_i, yz)$. Следовательно, $xz \equiv yz$, т. е. введенное отношение эквивалентности правоинвариантно. И наоборот, для заданного правоинвариантного конечного отношения эквивалентности на множестве входных последовательностей мы можем построить автомат с минимальным множеством состояний, который определяет отношение эквивалентности на множестве входных последовательностей, совпадающее с первоначальным. Каждому классу эквивалентности соответствует вполне определенное состояние

автомата. Следует как-то выделить начальное состояние, для этого дополним множество входных последовательностей пустой последовательностью. Пустая входная последовательность обозначается Λ, а все множество входных последовательностей плюс пустая последовательность обозначается X^*. Класс эквивалентности, содержащий элемент Λ, соответствует начальному состоянию. Может так случиться, что этот класс содержит только пустую последовательность. Функция переходов определяется с помощью соотношения $\delta([x], a_i) = [xa_i]$, где *[x]* обозначает состояние, соответствующее классу эквивалентности, содержащему входную последовательность *x*. Эти правоинвариантные классы эквивалентности тесно связаны с множеством функций, получающихся с помощью левых сдвигов, и могут быть использованы для нахождения автомата с минимальным множеством состояний и с заданным внешним поведением. Приведем примеры отождествления различных входных последовательностей, когда состояния, не принадлежащие минимальному множеству, не учитываются. Предположим, что надо построить автомат с множеством входных сигналов $x = \{0, 1\}$, распознающий входную последовательность 111. Очевидно, что для этого автомата получится четыре класса эквивалентности: {1}, {11}, {111} и еще класс, содержащий все остальные последовательности. Если же мы строим автомат, в котором последующее состояние определяется после того, как в точности три символа попадут на вход, то для него классы будут иметь вид: {0, 00, ..., 111, ...}, {1, 01, ...}, {11, 101, ...}. Здесь можно не рассматривать последовательности, длины которых больше трех.

3. Конечный автомат определяет конечное отношение конгруэнтности на множестве входных последовательностей. Множество классов эквивалентности в этом случае получается путем измельчения классов эквивалентности, введенных в пункте 2. Для каждой входной последовательности автомат начинает поочередно работать из каждого начального состояния, переходя в соответствующее конечное состояние. Последовательности будут эквивалентными, если они вызывают для каждого начального состояния одинаковые переходы. Если последовательности *x* и *y* эквивалентны в этом смысле, то для любых входных последовательностей *u, v* и для любого состояния $q_i \in Q$ справедливо соотношение $\delta(q_i, uxv) = \delta(q_i, uyv)$. Следовательно, $uxv \equiv uyv$, т. е. введенное отношение эквивалентности левоинвариантно и правоинвариантно. Таким образом, получено отношение конгруэнтности. В силу свойства лево- и правоинвариантности классы эквивалентности можно умножать друг на друга по формуле *[x] [y] = [xy]*. Множество классов эквивалентности с введенным законом композиции образует конечную полугруппу. Более удобно

множество классов конгруэнтности ввести при помощи внешнего поведения автомата. Мы воспользуемся функциями типа вход—выход для того, чтобы ввести понятие эквивалентности последовательностей, а именно назовем последовательности x и y эквивалентными, если $f_i(uxv) = f_i(uyv)$ для всех f_i из данного множества функций. Вновь с помощью введенного отношения конгруэнтности на X^* можно построить автомат. Так же как и раньше, состояния и функция перехода определяются из классов эквивалентности. Введенная классификация состояний играет большую роль в описании поведения конечного автомата. Всевозможные действия, выполняемые конечным автоматом, можно рассматривать как некоторые конструкции на этих классах.

2.2.1. Функции переходов

Для автомата типа состояние—выход можно изучать переходы в множестве состояний независимо от тех выходов, которые они определяют. Мы исследуем структуру переходов состояний, рассмотрев все переходы, вызываемые каждым входным элементом, т. е. отображения множества состояний в себя, определяемые входными последовательностями. Эти отображения — основа алгебраической теории автоматов.

1. Полугруппа отображений переходов. Отображения переходов, соответствующие входным элементам, можно перемножать. В результате получится отображение переходов, соответствующее входной последовательности, которая есть результат сочленения входных элементов. Так как множество состояний конечно, то множество отображений перехода замкнуто относительно умножения и конечно. Следовательно, множество отображений переходов представляет собой **абстрактную конечную полугруппу**. Этот результат позволяет применять в теории автоматов алгебраические методы. Все основные свойства конечных автоматов находятся при помощи введенной полугруппы. Кроме того, полугруппа отображений переходов совпадаете полугруппой, построенной с помощью отношения конгруэнтности на множестве входных последовательностей. Последовательность каждого класса определяет отображение переходов, соответствующее этому классу.

Алгебраическая теория автоматов решает следующие задачи:

1. Построение стандартного представления для произвольного автомата.

2. Построение различных реализаций с разными множествами состояний для данного внешнего поведения.
3. Декомпозиция автомата в последовательно-параллельное соединение связанных между собой компонент с применением методов декомпозиции полугрупп.
4. Определение при помощи декомпозиции полугрупп неприводимых компонент, из которых строятся все автоматы.

2. Стандартный полугрупповой автомат. С произвольным автоматом можно связать автомат некоторого стандартного вида с внешним поведением как у первоначального автомата. Прежде всего определено множество классов конгруэнтности на множество X^*, а следовательно, определена полугруппа для произвольного автомата. Эта полугруппа, обозначаемая S_M, содержит единицу. Построим автомат с множеством состояний S_M, ВХОДНЫМ алфавитом S_M И ВЫХОДНЫМ алфавитом Y, таким же, как у первоначального автомата. Функция переходов определяется из соотношения $\delta(s_1, s_2) = s_1 \cdot s_2$. Новый автомат есть автомат типа состояние — выход с функцией выходов $i_M : S_M \to Y$. В обоих автоматах действия входов в некотором смысле одинаковы, но входы полугруппового автомата получаются с помощью кодирующей функции $h_M : X \to S_M$, которая переводит входной элемент a_i в элемент полугруппы S_M, соответствующий классу конгруэнтности, содержащему a_i. Действие автоматов окажется одинаковым только тогда, когда они начинают работать из соответствующих начальных состояний. Полугрупповой автомат должен начинать работать из единичного состояния, а первоначальный автомат —из некоторого выбранного состояния. Функция выходов i_M строится следующим образом: из каждого класса конгруэнтности выбирается произвольная входная последовательность, соответствующее ей состояние находится в первоначальном автомате при выбранном заранее начальном состоянии, для построенного состояния находится в первоначальном автомате выходной элемент. Многие автоматы имеют одинаковые полугруппы и это их родство очень важно для понимания того, как различные автоматы могут производить одинаковую работу. Соответствующее исследование будет проведено после выделения различных типов автоматов.

3. Типы отображений переходов. Существуют два основных вида отображений переходов:
1) перестановки, когда отображение взаимно однозначно на множестве состояний; автомат только с такими переходными отображениями обладает полугруппой, которая в действительности есть группа;

2) стягивания, когда два различных состояния переходят в одно и то же состояние.

Основной результат теории декомпозиции Крона—Роудза заключается в том, что посредством собственной декомпозиции произвольного автомата получаются компоненты, являющиеся групповым автоматом или весьма элементарным стягивающим автоматом.

2.2.2. Групповые автоматы

Переходы в пространстве состояний группового автомата представляют собой перестановки этого множества, образующие группу. Часто множество перестановок образуют группу, но интерес представляют только те, которые оказываются транзитивными, т. е. когда любое состояние может быть получено из произвольного состояния.

Перестановки можно найти с помощью выделения произвольной подгруппы и образования правых классов смежности. Каждый класс смежности соответствует теперь элементу множества состояния; трудность, возникающая на этом этапе, состоит в том, что представление может не быть точным. Следующее условие гарантирует точность представления: выбранная подгруппа должна содержать нетривиальные нормальные подгруппы первоначальной группы.

Каждая подгруппа абелевой группы является нормальным делителем, следовательно, каждый автомат с абелевой группой имеет только одно транзитивное множество состояний, каждый элемент которого соответствует элементу группы.

Групповые автоматы с выходом. Может существовать несколько транзитивных представлений группы и, следовательно, возможно несколько множеств состояний для автомата с заданной группой. Тем не менее множество состояний не может быть выбрано произвольно, так как выбор состояний может быть ограничен поведением автомата на выходе. Для автомата типа состояние—выход это следует прежде всего из рассмотрения стандартного автомата с состояниями, соответствующими каждому элементу группы. Каждому состоянию соответствует вполне определенный выходной элемент, который определяется с помощью первоначально заданного автомата. Следовательно, состояния можно объединить в правые классы смежности, если только все состояния каждого класса имеют одинаковый выход.

В качестве простого примера рассмотрим нециклическую группу порядка 6. Она обладает двумя образующими *A, C,* которые можно

рассматривать как пару входных элементов стандартного автомата. Таблицы состояний и выходов имеют тогда вид:

Состояние	I	A	A^2	C	CA	CA^2
Вход A	A	A^2	I	CA	CA^2	C
C	C	CA^2	CA	I	A^2	A

Состояние	I	A	A^2	C	CA	CA^2
Вход	a	b	c	a	b	c

с выходным алфавитом $\{a, b, c\}$. Легко видеть, что классам смежности $\{I, C\}$, $\{A, CA\}$, $\{A^2, CA^2\}$ соответствуют вполне определенные выходы, это позволяет ввести только три состояния.

2.2.3. Последовательно-параллельная декомпозиция групповых автоматов

Групповой автомат можно разложить в последовательно-параллельное соединение двух автоматов, когда группа содержит подгруппу. Основная задача — разложение функции переходов, так как функция выходов всегда может быть восстановлена при помощи отображения переходов и отображения выходов из всей группы в выходной алфавит. Идея доказательства заключается в параметризации элементов группы, а следовательно, и в параметризации состояний стандартной машины. Переходы состояний находятся тогда отдельно для каждого параметра в соответствующем автомате.

Рассмотрим стандартный групповой автомат, каждому элементу полугруппы которого соответствует одно состояние. Группа G содержит подгруппу H. Мы можем, следовательно, представить группу G в виде $G = H + Hc_1 + Hc_2 + \ldots + Hc_n$, где элементы c_i образуют соответствующие классы смежности (Элементы $c_1, \ldots, c_n$ — представители соответствующих смежных классов и в дальнейшем фиксированы). Таким образом, каждый элемент группы представляется в виде $g = hc$. Если групповой автомат находится в состоянии g, а на вход поступает элемент $g_{1,t}$ то он переходит в новое состояние $g' = gg_1$. Пользуясь разложением состояний на классы смежности, получаем соотношение $h'c' = hcg_1$. Произведем это умножение следующим образом: во-первых, положим $c' = \overline{cg_1}$, где $\overline{x}$ обозначает представитель класса смежности, содержащего элемент x. Тогда получаем равенство $h' = hcg_1\,(\overline{cg_1})^{-1}$.

Элемент cg_1 принадлежит к группе и поэтому может быть переписан в виде $h''c''$, откуда вытекает соотношение $cg_1\,(\overline{cg_1})^{-1} = h''$.

Следоательно, элемент h' представляется как произведение элементов, принадлежащих подгруппе H. Процесс нахождения представителя нового смежного класса зависит только от представителя старого смеж-

ного класса и входного элемента, благодаря чему его можно изобразить как действие автомата, каждое состояние которого соответствует представителю смежного класса. Этот **автомат** будем называть **начальным. Отображение переходов начального автомата при поступлении входного элемента, принадлежащего группе, соответствует переходу старого смежного класса в новый**. Для выполнения приведенного ранее разложения элемента h' необходимы выходной элемент из начального автомата, а также элемент h. Автомат, состояния которого соответствуют элементам полугруппы H, — стандартный автомат для Н.

Для того чтобы можно было построить соединение двух введенных автоматов, необходимо разрешить только один вопрос: **выходной элемент начального автомата не воспринимается вторым автоматом, в него должен поступать элемент $cg_1\,(\overline{cg_1})^{-1}$, который может быть найден с помощью кодирующей функции.** Эта кодирующая функция зависит только от текущих элементов c и g_1, ее **отличает от автомата отсутствие памяти**. Следовательно, **кодирующая функция представляет собой комбинаторную логическую схему.**

Выходной элемент последовательно-параллельного соединения двух автоматов находится следующим образом: с помощью кодирующей функции из выходных элементов начального и подгруппового автомата формируется их произведение, являющееся элементом группы, затем этот групповой элемент кодируется в выход автомата. Последовательно-параллельная декомпозиция группового автомата представлена на рис. 2.

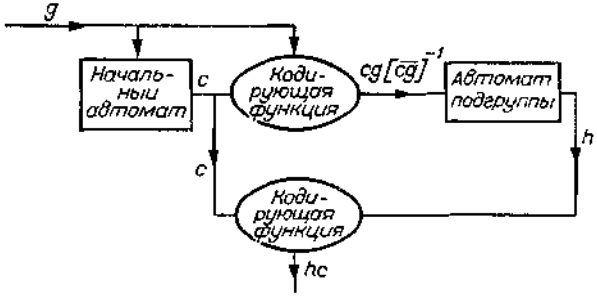


Рис . 2. Последовательно-параллельная декомпозиция группового автомата

Эквивалентность, имеющаяся между последовательно-параллельным соединением автоматов и первоначальным стандартным групповым автоматом, заключается в том, что каждой паре элементов (c, h), являющейся состоянием последовательно-параллельного соединения,

соответствует некоторое состояние группового автомата. Следовательно, эта эквивалентность не зависит от начального состояния, в то время как соотношение между произвольным автоматом и его стандартным групповым автоматом зависит от начального состояния. Заметим также, что в двух построенных нами автоматах, а также в их последовательно-параллельном соединении между входом и выходом имеется задержка на единичный элемент времени.

Приведенная декомпозиция может быть продолжена. Второй автомат — это стандартный групповой автомат, т. е для него можно применять указанный метод разложения. Первый автомат не является, как правило, групповым, прежде чем производить разложение, его следует заменить стандартным групповым автоматом. Далее будет показано, что группа первого автомата—гомоморфный образ группы *G*. Если *K* — наибольшая подгруппа в H, представляющая собой нормальный делитель в *G,* то группа первого автомата есть *G/K*—факторгруппа группы *G* по *K*. Если группа *G* не содержит нетривиальных нормальных делителей, т. е. если она *простая,* то группа первого автомата есть в точности *G*. Следовательно, производить разложение простых групп не имеет смысла. При продолжении процесса разложения автомата с группой *G* будут появляться простые группы, являющиеся гомоморфными образами всех подгрупп первоначальной группы *G*. Предположим, что на некотором этапе разложения все простые группы, являющиеся гомоморфными образами подгрупп группы G, будут выделены, тогда никаких новых групп возникнуть не может. Эти простые группы представляют собой **фундаментальные компоненты**, из которых строится первоначальный автомат.

Любой групповой автомат разлагается на автоматы простых групп, являющихся гомоморфными образами подгрупп первоначальной группы. Автоматы простых групп неразложимы.

Если использованная для разложения подгруппа H есть нормальный делитель в группе G, то первый автомат — **стандартный групповой автомат с группой *G/H***. Кодирование входных элементов первоначального группового автомата во входные элементы первого автомата последовательно-параллельного соединения осуществляется с помощью функции, которая есть гомоморфизм из группы *G* в факторгруппу *G/H*.

Примеры декомпозиции группового автомата. 1. Циклическая группа порядка 6: $G = \{I, A, A^2, A^3, A^4, A^5\}$. Выберем подгруппу $H = \{I, A^2, A^4\}$. Смежные классы имеют вид $\{I, A^2, A^4\}$, $\{A, A^3, A^5\}$.

В результате начальный автомат последовательно-параллельного соединения будет автоматом группы второго порядка, а второй автомат — автоматом группы третьего порядка.

Выбирая в качестве представителей смежных классов элементы I и A, найдем для соответствующих автоматов кодирующую функцию, определенную на множестве пар (c, g):

(I, I), (I, A), (A, I), (A, A^5) переходит в I;
(I, A^2), (I, A^3), (A, A^2), (A, A) переходит в A^2;
(I, A^4), (I, A^5), (A, A^3), (A, A^4) переходит в A^4.

2. Нециклическая группа порядка 6: $G = \{I, A, A^2, C, CA, CA^2\}$. Выберем подгруппу $H = \{I, A, A^2\}$. Смежные классы имеют вид $\{I, A, A^2\}$, $\{C, CA, CA^2\}$.

Вновь начальный автомат последовательно-параллельного соединения будет автоматом группы второго порядка, а второй автомат — автоматом группы третьего порядка.

Если элементы I и C выбраны в качестве представителей смежных классов, то кодирующая функция для автоматов имеет вид:

(I, I), (I, C), (C, I), (C, C) переходит в I;
(I, A), (I, CA^2), (C, CA), (C, CA^2) переходит в A;
(I, A^2), (I, CA), (C, A), (C, CA^2) переходит в A^2.

Два рассмотренных примера показывают, что кодирующая функция между автоматами последовательно-параллельного соединения имеет существенное значение для построения группы первоначального автомата.

2.2.4. Декомпозиция полугрупповых автоматов

1. Возвраты. В том случае, когда полугруппа автомата не является группой, некоторые из его отображений переходов должны быть **стягивающими**. Мы рассмотрим сначала специальный случай **стягивающих отображений.**

Отображение переходов, переводящее все состояния в некоторое одно состояние, будет называться возвратным. Если элемент r полугруппы S отображений переходов возвратный, то $sr = r$ для всех элементов $s \in S$, т. е. элемент r— правый нуль в абстрактной полугруппе. При произвольном фиксированном $s \in S$ элемент $rs = r'$ возвратный в полугруппе S. Если отображения переходов автомата перестановочные или возвратные, то такой автомат называется **перестановочно-возвратным.**

2. Декомпозиция перестановочно-возвратных автоматов. Очевидно, что полугруппа S перестановочно-возвратного автомата представляет собой объединение группы G и множества возвратных элементов *R.* Стандартный автомат полугруппы *S* может быть представлен последовательно-параллельным соединением группового автомата с группой G и стандартного полугруппового автомата с полугруппой R^1, полученной из *R* в результате добавления двусторонней единицы.

Каждое состояние стандартного полугруптюового автомата с полугруппой S параметризуется как s = *rg,* где $r \in R^1$, $g \in G$, т. е. каждый из параметров — состояние соответствующей компоненты. Если автомат полугруппы S находится в состоянии *rg* и на вход поступает элемент *s,* то он переходит в новое состояние *r'g',* имеющее вид *r'g' = rgs.* Рассмотрим два случая:

1. $s \in R$, $r'g' = rgs = s$. Положим в этом случае $g' = g$ и тогда $r' = rgsg^{-1} = sg^{-1}$.

Теперь все определяется работой автомата полугруппы R^1 и поэтому входная кодирующая функция кодирует элемент *s* как единицу на входе группового автомата и как элемент sg^{-1} на входе автомата полугруппы R^1. Следовательно, для действия автомата полугруппы R^1 нужны состояния автомата группы *G,* следовательно, автомат полугруппы R^1 — второй элемент последовательно-параллельного соединения.

2. $s \in G$. Положим в этом случае $g' = gs$ и оставим $r' = r$, так что $r'g' = r(gs)$. Здесь входная кодирующая функция кодирует элемент s как единицу на входе автомата полугруппы R^1 и как элемент s на входе автомата группы *G.*

Схема изменений состояния включает два произвольных выбора, но всегда дает правильный результат. Выходная кодирующая функция формирует из двух выходных элементов автомата группы *G* и автомата полугруппы R^1 их произведение. На рис. 3 изображена описанная декомпозиция.

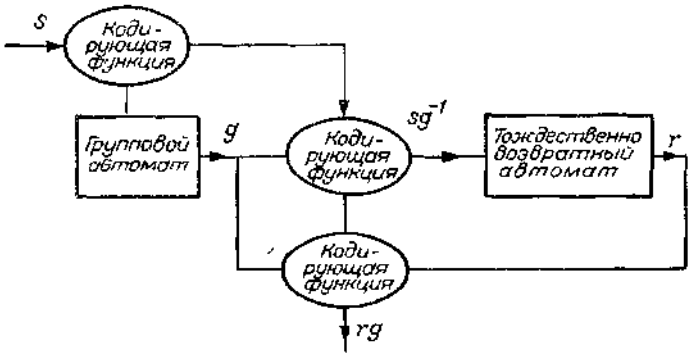


Рис. 3. Последовательно-параллельная декомпозиция перестановочно-возвратного автомата

3. Тождественно-возвратные автоматы. Простейший тожлественно-возвратный автомат имеет два состояния, а его полугруппа состоит из трех элементов. Эта полугруппа имеет вид $\{r_0, r_1, I\}$, где элемент r_0 возвращает в нулевое состояние, элемент r_1 — в состояние 1, а элемент I соответствует тождественному переходному отображению.

Действие любого тождественно-возвратного автомата можно воспроизвести с помощью параллельного соединения некоторого числа тождественно-возвратных автоматов с двумя состояниями. Каждый входной элемент произвольного тождественно-возвратного автомата кодируется двоичной последовательностью соответствующей длины и ее разряды воспринимаются автоматами с двумя СОСТОЯНИЯМИ. ЕСЛИ в разряде цифра 0, то автомат возвращается в состояние 0, в противном случае— в состояние 1. Тождественный входной элемент кодируется в тождественные входные элементы каждого тождественно-возвратного автомата с двумя состояниями.

Следовательно, **перестановочно-возвратный автомат может быть разложен в соединение группового автомата и набора параллельно соединенных тождественно-возвратных автоматов с двумя состояниями.**

4. Общие полугрупповые автоматы. Зейгер показал, что любой общий полугрупповой автомат можно разложить в соединение перестановочно-возвратных компонент. Напомним, что в случае групповых автоматов состояния объединялись в блоки (смежные классы) так, что под действием произвольного входного сигнала осуществлялся переход всего блока в какой-либо вполне определенный блок. Такие

движения блоков называются переходами состояний в начальном автомате последовательно-параллельного соединения. В случае группового автомата введенные блоки состояний не пересекались, однако для общего полугруппового автомата они будут пересекаться.

Для разложения автомата в последовательно-параллельное соединение перестановочно-возвратных автоматов обычно требуется более двух компонент, процесс разложения итеративный и на каждом шаге меняется последовательность компонент. Сначала вырабатывается множество блоков, затем оно измельчается, что позволяет получить множество состояний последующих автоматов.

Для того чтобы начать декомпозицию, рассмотрим стандартный полугрупповой автомат с множеством состояний S. Выберем в полугруппе все элементы c_i, определяющие стягивающие отображения переходов. Сформируем подмножества вида Sc_i, отбросив те из них, которые целиком содержатся в других. Все элементы полугруппы, не принадлежащие введенным множествам, объединим в отдельное подмножество. Построенные подмножества образуют начальное множество блоков. При поступлении на вход автомата стягивающего элемента c_j множество Sc_i переходит в множество Sc_ic_j, целиком содержащееся в множестве Sc_j так, что каждый блок переходит в некоторый новый блок. Следовательно, стягивающие входные элементы основного автомата представляют собой возвратные входные элементы начального автомата. Входной элемент $p \in S$, определяющий перестановку в множестве состояний, переводит блок Sc_i в множество Sc_ip, для разных блоков эти множества будут разными и снова будут представлять собой блоки. Следовательно, входные элементы, определяющие перестановки в множестве состояний основного автомата, определяют перестановки и для начального автомата. Если переставляющие входные элементы p_1, p_2, p_3 удовлетворяют соотношению $p_1 \cdot p_2 = p_3$, то $Sc_ip_1p_2 = Sc_ip_3$, поэтому множество перестановок начального автомата — гомоморфный образ множества перестановок полугруппы S, т. е. подгруппа. В результате подразбиения каждого блока начального множества в совокупность пересекающихся подблоков получим измельчение начального множества блоков. Можно ввести второй автомат и связать его в последовательно-параллельное соединение с начальным автоматом. Второй автомат имеет достаточно состояний для того, чтобы можно было указать, как происходит подразбиение блоков в начальном автомате. Входными элементами второго автомата являются пары, состоящие из входных элементов основного автомата и состояний начального автомата. Более детальное ознакомление с конструкцией, показывающей, как произвольный полугрупповой автомат можно

представить в виде последовательно-параллельного соединения перестановочно-возвратных автоматов, группы которых — гомоморфные образы подгрупп полугруппы данного автомата, будет приведено в последующих разделах.

5. Результат Крона — Роудза. Примеры декомпозиции групповых и перестановочно-возвратных автоматов — частные случаи общей теоремы Крона—Роудза, утверждающей, что автоматы, полугруппы которых есть простые группы, а также тождественно-возвратный автомат с двумя состояниями неразложимы на более элементарные компоненты. Кроме того, любой автомат может быть разложен в последовательно-параллельное соединение тождественно-возвратных автоматов и автоматов простых групп, причем эти простые группы— гомоморфные образы всевозможных подгрупп полугруппы данного автомата. Для одного и того же автомата может существовать несколько разложений в последовательно-параллельное соединение автоматов простых групп и тождественно-возвратных автоматов с двумя состояниями. При этом может использоваться различное число автоматов простых групп и тождественно-возвратных компонент с двумя состояниями.

6. Некоторые методы декомпозиции Крона—Роудза. Для того чтобы показать некоторые методы декомпозиции, отличающиеся от приведенных ранее, кратко изложим некоторые результаты, принадлежащие Крону и Роудзу и составляющие основную часть их фундаментальной теоремы. Крон и Роудз разбивают все **конечные полугруппы на три класса: циклические полугруппы, простые слева полугруппы и полугруппы, содержащие собственный левый идеал и собственную подполугруппу, объединение которых есть вся полугруппа.** Соответствующие автоматы разлагаются следующими способами:

1) **полугрупповой автомат, полугруппа которого циклическая**. Циклическая полугруппа содержит элементы вида $a, a^2, \ldots, a^n, a^{n+1}, \ldots, a^{n+m} = a^n, a^{n+m+1} = a^{n+1}$ и т. д. Этот автомат может быть реализован с помощью преобразования состояния (переходы показаны на рис. 4).

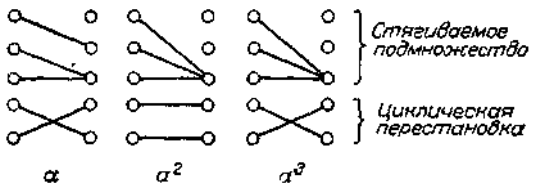


Рис. 4. Переходное отображение для циклической полугруппы

Множество состояний есть объединение двух подмножеств - одно из них соответствует стягиваниям, а другое — циклическим перестановкам; эти подмножества состояний не пересекаются, с помощью их строим параллельное соединение автоматов. Один из них будет возвратным, а другой — автоматом циклической группы;

2) **полугрупповой автомат, полугруппа которого простая слева.** Для любого подмножества *I* простой слева полугруппы S выполняется соотношение *SI* = S. Из этого с очевидностью вытекает, что простая слева полугруппа есть объединение группы и множества ее левых нулей. Множество отображений переходов, соответствующих множеству левых нулей, имеет следующую структуру. Множество состояний содержит два подмножества. Одно подмножество соответствует неподвижной части каждого отображения, второе— стягиваниям в элементы первого подмножества. Общая простая слева полугруппа имеет вид: группа × (левые нули полугруппы); соответствующий ей автомат разлагается в параллельное соединение группового автомата и автомата полугруппы левых нулей. Как и в первом случае, получаем отдельно возвратный и групповой автоматы;

3) **автоматы подполугруппы и левого идеала**. Так как случаи 1 и 2 уже разобраны, остается рассмотреть класс полугрупп *S,* таких, что S содержит собственный левый идеал *T* и собственную подполугруппу *V,* причем выполнено соотношение $S = T \cup V$.

Элементы s полугруппы S параметризуются с помощью соотношения s = *tv,* где $t \in T', v \in V'$, а множества *T′* и *V′* представляют собой несколько модифицированные множества *T* и *V* соответственно. В рассматриваемом последовательно-параллельном соединении автомат, соответствующий множеству *V′,* будет первым, а автомат, соответствующий множеству *T′,* — вторым. Входная кодирующая функция изменяет входные элементы одним из следующих способов. Если вход s удовлетворяет соотношению $s \in S - T$, то он поступает на вход автомата, соответствующего множеству *V,* а на вход автомата множества *T′* поступает единица. Новое состояние определяется из старого с помощью соотношения *t′v′ = tvs,* умножение *vs* производится в автомате множества *V′.* Если же справедливо соотношение $s \in T$, то кодирующий элемент, имеющийся между автоматами, переводит элемент *vs* в автомат множества *T′.* Автомат множества *V′* переходит в некоторое специальное состояние.

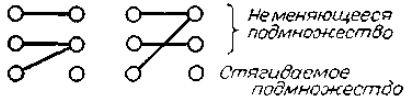


Рис. 5. Переходные отображения для левых нулей

Покажем, как связаны автоматы множеств T' и V с автоматами идеала T и полугруппы V. Автомат множества T' получается в результате добавления к идеалу T единичного элемента. Автомат множества V' получается в результате добавления к полугруппе V некоторого специального состояния c. Если элемент c поступает на вход автомата, то он определяет возврат в состояние c, если автомат находится в состоянии c, а на вход поступает элемент $v_i \in V$, то он определяет переход в состояние v_i.

Построенное разбиение на автоматы, соответствующие множествам T' и V', полностью определяется рассмотренными случаями, в результате получаются групповая и возвратная компоненты. На каждом этапе имеется произвол в выборе идеала T, поэтому могут быть получены совершенно разные последовательности автоматов.

7. Значение последовательно-параллельного соединения.

Последовательно-параллельное соединение автоматов — наиболее важный тип соединений, в которых отсутствует обратная связь между выходом и входом. Для того чтобы представить себе различные явления, возникающие при обратной связи, рассмотрим двойной счетчик, входные элементы которого пропускаются через вентиль, как это показано на рис. 6.

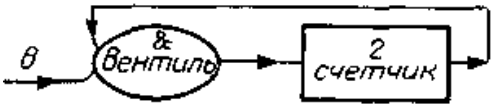


Рис. 6. Стягивающий автомат

Когда счетчик устанавливается в положение 0, подача входных сигналов прекращается и счетчик фиксируется в определенном положении. Следовательно, групповой автомат определяет стягивающее действие. В качестве другого примера рассмотрим задерживающий элемент, состоящий из двух параллельных двоичных разрядов, для которого вход и обратная связь проходят через комбинационную логическую схему. С помощью логической схемы выполняют умножение на элементы группы четвертого порядка. В настоящем примере задерживающий элемент представляет собой стягивающий автомат, который определяет такое действие, как групповой автомат.

В последовательно-параллельном соединении групповые компоненты всегда соответствуют действию групповых автоматов, а стягивающие компоненты добавляются для получения стягивающего действия. При обратной связи групповое и стягивающее свойства отображения переходов не сохраняются, в действительности же может быть получено

отображение переходов любого вида. Если суть последовательно-параллельной декомпозиции заключается в том, что она рассоединяет состояния различных автоматов, насколько это возможно, то обратная связь пересоединяет состояния различных автоматов.

2.2.5. Модель алгебраического автомата

Инженеры постоянно сталкиваются с конструированием конечных автоматов, так как последние являются важнейшими элементами вычислительных машин, средств связи и систем автоматического управления. Современная техника проектирования в основном эмпирическая, поэтому следовало бы приветствовать любые методы, которые вели бы к более надежному и хорошему конечному результату. Однако следует признать, что алгебраическая теория автоматов в настоящее время не нашла еще надлежащего практического применения. Это объясняется несколькими причинами, после их обсуждения мы попытаемся показать, как можно будет практически применять алгебраическую теорию автоматов, когда с самого начала общее действие автомата представлено в алгебраической форме. Следует указать еще одну алгебраическую область приложения — создание кодов с исправлением ошибок. Когда код описывается алгебраически, соответствующие кодирующий и декодирующий автоматы также описываются алгебраически. Однако методы исследования кодирующих и декодирующих автоматов и их применение выходят за рамки алгебраической теории автоматов. В теории кодирования, как правило, используются линейные автоматы, они будут рассмотрены в конце этого параграфа.

Модели алгебраической теории автоматов ограничиваются только такими явлениями, действие которых можно описать конечными автоматами. Рассматриваются только абстрактные состояния и их изменения под действием последовательности входных сигналов. Пренебрегают измененииелг состояний при установке электронных элементов — триггеров и линий задержки, а также в реакции на входные воздействия при установке этих элементов.

Работа, связанная с синтезом логических схем, может быть разбита на три этапа. Сначала осуществляется описание типа вход—выход (внешнее описание), первым производится логический синтез. Затем определяется множество состояний, находится таблица переходов и выходной элемент для каждого состояния. После этого переходят к аппаратуре. Каждому состоянию должна быть поставлена в соответствие определенная настройка запоминающего устройства (например, триггера), это — проблема распределения

состояний. Основная задача при этом — быстродействие, экономия, а также использование стандартных имеющихся в наличии приборов. После нахождения реальных приборов следует продолжить выбор, чтобы требуемые переходы состояний не искажались быстрыми несинхронными электрическими импульсами или незначительными задержками в характеристиках электронных элементов.

Алгебраическая теория автоматов непосредственно применяется только на первом этапе, однако не явно она может быть использована на втором и третьем этапах при построении других автоматов, реализующих заданное отображение вход—выход, т. е. обладающих одинаковым **внешним поведением**. На практике часто быстродействие необязательно и конструкторская работа в основном определяется необходимостью избежать ошибок в переходах состояний.

1. Реализация абстрактного автомата из отображения вход—выход. Алгебраическая теория автоматов позволяет установить связь между требуемым внешним описанием (функцией вход—выход) и реализующим его абстрактным автоматом. Следует уяснить большое значение этой связи и значительные практические трудности ее реализации.

Напомним, что заданное внешнее полное поведение определяется функцией $f : X^* \to Y$, где X — входной алфавит, X^* — множество всех цепочек над входным алфавитом плюс пустая цепочка, а Y — выходной алфавит. Элемент $f(x)$ есть конечный выходной символ, определяемый входной цепочкой x при условии, что автомат начинает работать из заданного начального состояния. Полугруппа автомата может быть тогда найдена при помощи образования классов эквивалентности над X^* с отношением эквивалентности $x \equiv y$, если $f(uxv) = f(uyv)$ для всех u и v из X^*. Множество классов эквивалентности, умножение которых задается сочленением соответствующих цепочек, определяет полугруппу автомата. Каждое представление этой полугруппы в множество отображений конечного множества состояний есть, по существу, таблица переходов и множество состояний для реализации автомата. Если еще добавить функцию выходов из множества состояний в выходной алфавит, то можно полностью восстановить функцию f.

На этапе построения полугруппы автомата могут возникнуть значительные практические трудности, так как часто имеется только часть внешнего описания, т. е. только некоторые пары, определяющие функцию f, заданы, а на остальном множестве входных цепочек функцию f можно доопределить произвольным образом. Задача

доопределения функции наиболее естественным способом в общем случае пока не решена.

Предположим, что полугруппа автомата найдена. Если она не является простой группой или тождественно-возвратной полугруппой, воспользуемся последовательно-параллельной декомпозицией. В результате получим полугруппы соответствующих автоматов последовательно-параллельного соединения. На этом этапе вновь возникают значительные трудности. Возможно полугруппу потребуется задать ее таблицей умножения. Эта таблица будет существенно больше, чем таблица переходов. Например, таблица автомата с 10 состояниями и 2 входами будет содержать 20 элементов, в то же время полугруппа этого автомата может иметь 10^{10} элементов, а, стало быть, ее таблица умножения состоит из 10^{20} элементов. Проблема размещения фактически запрещает использовать полугруппы, заданные в таком виде: они могут применяться благодаря своим общим качественным свойствам.

2. Некоторые достижения в решении задачи реализации автомата. В ряде случаев для некоторых специальных отображений вход—выход получены определенные результаты в решении задачи реализации. В частности, существует определенный подход, **когда задача сформулирована как вычислительный алгоритм**, при этом, быть может, действия должны производиться над элементами из некоторого конечного поля**. Например, контроль на четность—нечетность в декодировании.** Самый простой подход здесь заключается в переходе от вычислений к алгоритму, а затем к машинной программе. **Результирующим автоматом станет тогда специализированная вычислительная машина, действующая по данной программе.** Такой подход имеет большие практические преимущества при исправлениях и эксплуатации, так как работа автомата разложена в последовательность легко понятных шагов. Однако теряются некоторые свойства, связанные с конечной памятью.

Иногда переходы состояний автомата могут быть определены непосредственно из уравнений. К такой ситуации следует стремиться и она складывается при конструировании кодирующих и декодирующих элементов. Кодирующие автоматы часто оказываются линейными.

3. Линейные автоматы. В теории кодирования **элементы кода представляются полиномами над конечным полем, а декодирующий процесс включает сложение и умножение этих полиномов**. Такая арифметика приводит к линейным автоматам. Линейный автомат содержит запоминающие устройства, которые накапливают элементарные значения, принадлежащие конечному полю. Кроме того, он содержит логические элементы, определяющие

сложение и умножение на константы. Линейный автомат с *n* запоминающими устройствами имеет состояния, соответствующие конечному множеству *n* векторов над конечным полем *F*. Входной сигнал X_i в момент *i* также будет *n* вектором над *F*. Если V_i — это вектор состояния в момент *i*, то состояние в следующий момент времени определяется по формуле

$$V_{i+1} = AV_i + BX_i,$$

где *A* и *B* — матрицы над полем *F*. Состояние в момент времени (*i* + 2) находят по формуле

$$V_{i+2} = AV_{i+1} + BX_{i+1} = A^2V_i + ABX_i + BX_{i+1}.$$

Из уравнений перехода состояния вытекает, что элементы полугруппы линейного автомата имеют вид $(A^r,\ X)$, где *X* есть *n*-вектор над полем *F*. Умножение в этой полугруппе выполняется по следующей формуле:

$$(A^r,\ X) \times (A^p,\ X') = (A^{r+p},\ X'') = (A^{r+p},\ A^pX + X').$$

Из этого соотношения следует, что полугруппа S_M автомата есть полупрямое произведение конечной циклической полугруппы $A^r = S_A$ и аддитивной группы векторного пространства G_V. Автомат может быть получен при помощи последовательно-параллельного соединения двух автоматов, соответствующих S_A и G_V. Операцию в последовательно-параллельном соединении легко представить себе, когда матрица *A* невырождена, в этом случае полугруппа S_A будет группой. Автомат, соответствующий S_A, хранит элементы вида A^r, а автомат, соответствующий G_V, — элементы *X*. Новый входной сигнал *X'* кодируется как (A, X') и элемент *A* поступает на вход автомата группы S_A, который переходит в состояние A^{r+1}. Автомат группы G_V получает входной сигнал $A^{-(r+1)}BX'$ и переходит в состояние $A^{-(r+1)}(AX + BX')$. Для работы автомата группы G_V требуются состояния автомата группы S_A.

Можно пойти дальше и произвести декомпозицию автоматов полугруппы S_A и группы G_V. Так как полугруппа S_A циклическая, ее автомат можно представить как параллельное соединение автоматов. Если полугруппа S_A имеет индекс *m* и период *n*, то полугруппа одного параллельного автомата будет иметь индекс *m* и период 1, а для второго автомата получим циклическую группу порядка *n*. Автомат циклической группы снова можно разложить в последовательно-параллельное соединение счетчиков. Группа G_V аддитивная, поэтому она может быть разложена в прямое произведение циклических групп простого порядка. Следовательно, автомат группы G_V моделируется последовательным соединением счетчиков по простому модулю.

Рассмотренная декомпозиция линейного автомата приводит к реализациям, несколько отличающимся от обычных.

4. Возможные приложения алгебраической теории автоматов. В схеме стандартной реализации автомата, показанной на рис. 7, логический переключатель содержит в себе кодирование таблицы умножения полугруппы автомата.

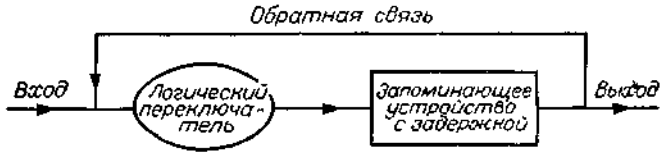


Рис. 7. Стандартная реализация автомата

На этом должно основываться конструирование схемы переключения. Ошибки в автоматах — это ошибки в переходах состояния. Они могут вызываться перемежающимися неисправностями в приборах или случайными причинами, связанными с конечностью времени действия приборов. При детальном изучении переходов состояния иногда удается так выбрать состояния, чтобы исключить лишние переходы и тем самым устранить случайности. Такое детальное изучение проводить очень трудно, поэтому следует рассматривать более общие схемы. Общие схемы устраняют ошибки вскоре после их возникновения в стандартных блоках памяти и переключающих схем. Они основываются на принципах кодов с исправлением ошибок; ошибки устраняются без учета причины их возникновения.

Связь между исправлением ошибок и полугруппой автомата пока еще не ясна, однако некоторые замечания можно сделать:

1) если полугруппа содержит правый нуль и он достаточно часто поступает на вход, то автомат будет возвратным и условий для ошибок не будет. В случае, когда отображение вход—выход задано не полностью, для получения предыдущей ситуации можно ввести правый нуль;

2) автоматы простых групп могут быть самостоятельными блоками исправления ошибок. Дальнейшее исправление их происходит внутри автоматов простых групп.

2.3. Декомпозиция автоматов и расширение полугрупп

Главная цель этого раздела состоит, во-первых, в том, чтобы установить явную связь между понятиями теории автоматов и понятиями

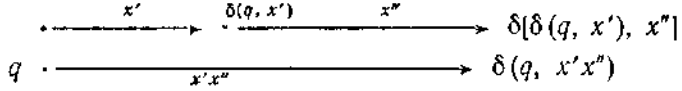
теории полугрупп (исследование автоматов с применением методов теории полугрупп получило широкое распространение), во-вторых, дать строгое доказательство интуитивно ясных и важных для дальнейшего изложения результатов. Изображение с помощью диаграмм каскадного соединения автоматов должно помочь читателю получить более простые доказательства известных результатов, а также открывать и доказывать новые факты.

В дальнейшем в алгебраической теории рассматриваются автоматы типа состояние—выход. Автоматом (или машиной) называется пятерка $M = (X, Y, Q, \delta, \beta)$, где X — конечное множество входных сигналов, Y — конечное множество выходных сигналов, Q— множество состояний (не обязательно конечное), $\delta : Q \times X \to Q$— одношаговая функция переходов, $\beta : Q \to Y$— функция выходов. Приведенное определение автомата позволяет интерпретировать его как систему, которая будучи в момент времени t в состоянии q генерирует на выходе сигнал $\beta(q)$, если же на вход в момент t поступил сигнал x, то в момент времени $t + 1$ автомат оказывается в состоянии $\delta(q, x)$.

Рассмотрим множество X* всех конечных последовательностей входных сигналов, включающее также пустую цепочку Λ. Множество X* представляет собой полугруппу относительно операции сочленения последовательностей, т. е. операция сочленения ассоциативна $(x_1x_2)\,x_3 = x_1\,(x_2x_3)$. Элемент Λ—единичный элемент полугруппы X*. Можно продолжить отображение δ до отображения множества $Q \times X^*$ в множество Q при помощи следующих соотношений:

$$\delta(q, \Lambda) = q$$

$$\delta(q, x'x'') = \delta[\delta(q, x'), x'']$$



Для каждого фиксированного состояния q автомата M при произвольной входной цепочке можно найти соответствующий выходной сигнал. Это определяет функцию $M_a : X^* \to Y$, где $M_q(x) = \lambda(q, x)$, а символ $\lambda(q, x)$ используется для более краткого обозначения композиции $\beta[\delta(q, x)]$. Автомат M ведет себя одинаково, если начальным состояниям q и q', из которых он начинает работать, соответствуют одинаковые функции типа вход—выход, т. е. если $M_q(x) = M_{q'}(x)$ для всех входных цепочек x. Поэтому если автомат интересует нас с точки зрения его внешнего поведения, то M следует привести к такому виду, чтобы одна и та же функция вход—выход не могла соответствовать более чем одному состоянию. Пусть автомат M,

находящийся в состоянии q, имеет функцию вход—выход f и пусть состояние q' *достижимо* из состояния q, т. е. можно найти входную последовательность $x \in X^*$, такую, что $\delta(q, x) = q'$. Тогда элементу q' соответствует функция

$$M_{q'}(x') = M_{\delta(q, x)}(x') = \lambda[\delta(q, x), x'] = \lambda(q, xx') = f(xx') = fL_x(x'),$$

где отображение $L_x : X^* \to X^*$ есть умножение слева на элемент x: $L_x(x') = xx'$. Отображение L_x *не является гомоморфизмом.*

Тогда для того чтобы построить приведенный автомат для M (здесь рассматриваются только конечные автоматы M, т. е. автоматы с конечным множеством состояний) , каждое состояние, достижимое из q, заменим на соответствующую ему функцию вход—выход fL_x, приведенный автомат (мы ограничиваемся только рассмотрением случая, когда все множество состояний достижимо из q) имеет конечное множество состояний, так как существует только конечное число различных функций вида fL_x (следовательно, к одной функции fL_x может приводить бесконечное число входных цепочек $x \in X^*$).

Далее мы будем рассматривать в основном автоматы, которые можно представить в виде

$$M(f) = (X, Y, Q_f, \delta_f, \beta_f),$$

где f — заданная функция, отображающая множество X^* в множество Y, причем

$$Q_f = \{g : X^* \to Y \mid g = fL_x \text{ для некоторого } x \in X^*\},$$

$$\delta_f(g, x) = gL_x,$$

$$\beta_f(g) = g(\Lambda)$$

и множество Q_f конечно. Следовательно, автомат $M(f)$ однозначно определяется функцией $f : X^* \to Y$, поэтому иногда мы будем рассматривать функцию $f : X^* \to Y$ как сам автомат, имея в виду на самом деле определяемую ею пятерку $M(f)$.

Автомату $M(f)$ можно сопоставить полугруппу f^S, а именно совокупность преобразований множества состояний Q_f, индуцированных входными последовательностями автомата, т. е.

$f^S = \{s : Q_f \to Q_f \mid \exists x \in X^*$, такой, что $s(q) = \delta(q, x)$ для всех $q \in Q\}$.

Через $[x]_f$ обозначим преобразование множества состояний, индуцированное цепочкой x. Введенная полугруппа конечна тогда и только тогда, когда конечно множество Q_f. Если автомат M не является приведенным, мы сопоставим ему (при некотором выбранном начальном состоянии q) полугруппу приведенного автомата $M(f)$, где f есть функция M_q.

Можно ввести функцию $i_f : f^S \to Y$, для которой выполняется ра-

венство $i_f(s) = f(x)$, если $s(q) = \delta(q, x)$. Это определение не зависит от выбора элемента x. Теперь построим новый полугрупповой автомат с отображением состояние—выход i_f, где

$$M(f^S, i_f) = (f^S, f^S, Y, \delta, i_f),$$

причем $\delta(s, s') = ss'$ (умножение элементов полугруппы). Обозначим как S^M автомат полугруппы S, это есть автомат $(S, S, S, \delta, 1)$, для которого $\delta(s, s') = ss'$ и $1(s) = s$. Мы говорим, что автомат M *моделирует* автомат M', если при условии соответствующего кодирования и декодирования входных и выходных символов автомат M может, находясь в подходящем состоянии, преобразовывать входные последовательности точно так же, как и автомат M'. Потребуем, чтобы кодирующие и декодирующие устройства не имели памяти (т. е. преобразовываем символы поодиночке) с тем, чтобы автомат M проделывал всю работу, связанную с запоминанием (см. рис. 1).

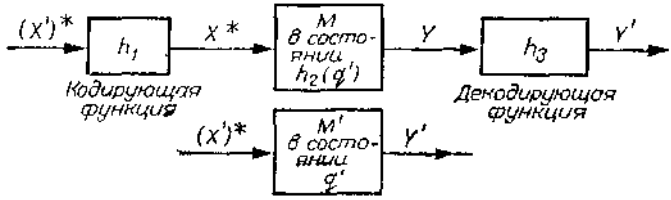


Рис. 1

Если автомат M моделирует M', станем писать M'|M и читать это так: *M' делит M.* Если $M' \mid M$ и $M \mid M'$, то автоматы M и M' называются *слабо эквивалентными.*

Полугруппы играют в теории автоматов важную роль, это связано с тем фактом, что автомат $M(f^S, i_f)$ слабо эквивалентен автомату $M(f)$. В самом деле, если мы введем отображение $j_f : X \to f^S$, для которого $j_f(x) = [x]_f$, то оба автомата идентичны с точки зрения их внешнего поведения при надлежащем выборе начальных состояний (см. рис. 2).

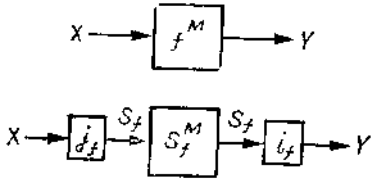


Рис. 2

Оказывается, что для подгрупп, так же как и для автоматов, имеется естественное понятие делимости. Мы говорим, что полугруппа S *делит* полугруппу *S'*, если существует полугруппа *S''* в *S'* и гомоморфизм *Z* подполугруппы *S''* на *S*, т. е.

$$S' \supseteq S'' \xrightarrow{Z} \twoheadrightarrow S.$$

Следовательно, в этом случае умножение в S' «моделирует» умножение в S и для этого требуется только часть полугруппы *S'*, а именно *S''*. Пусть S и S'— две полугруппы с определенными на них отображениями $i : S \to Y$ и $i' : S' \to Y'$, тогда скажем, что *(S, i) делит (S', i')*, если найдутся такие подполугруппа *S''* полугруппы *S'*, гомоморфизм *Z* подполугруппы *S''* на S и отображение $H: Y \to Y'$, что

$$i\,[Z\,(s)] = H\,[i'\,(s)] \text{ для всех } s \in S''.$$

Между понятиями делимости автоматов и полугрупп имеется важная связь. Автомат *M (g)*, начинающий работать из состояния *g*, моделируется автоматом *M* (*f*), начинающим работать из состояния *f*, тогда и только тогда, когда пара $(g^S,\ i_g)$ делит пару $(f^S,\ i_f)$. Нами рассматриваются только автоматы типа состояние— выход, множество состояний которых конечно.

2.3.1. Каскадные соединения

Рассматриваемый способ соединения автоматов обобщает рассмотренные последовательные и параллельные соединения автоматов.

1. Определение. Два заданных автомата типа состояние—выход *M'* и M, отображения $Z: \tilde{X} \times Y \to X'$ и $\eta: \tilde{X} \to X$ определяют автомат, называемый *каскадным соединением M'* и *M с отображением связи Z,* если

$$M' \times {}^{\eta}_{Z} M = (\tilde{X},\ Y' \times Y,\ Q' \times Q,\ \delta_Z,\ \beta_Z),$$

а отображения δ_Z и β_Z определяются согласно рис. 3 по формулам

$$\delta_Z\,[(q',\ q),\ \tilde{x}] = [\delta'\,\{q',\ Z\,(\tilde{x},\ \beta\,(q))\},\quad \delta\,(q,\ \eta\,(\tilde{x}))],$$

$$\beta_Z\,(q',\ q) = [\beta'\,(q'),\ \beta\,(q)].$$

Обычно отображение η в явном виде не упоминается.

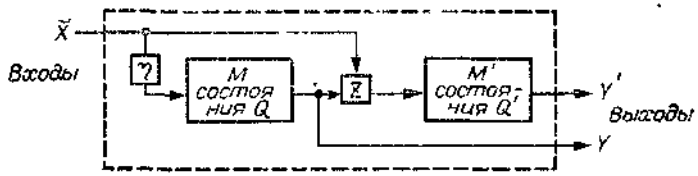


Рис. 3. $M' \times_Z^{\eta} M$ с состояниями

2. Определение. Автомат M называется *неприводимым,* если для каждого каскадного соединения $M' \times_Z M''$, которое он делит, выполнено одно из соотношений

$$M \mid M' \text{ или } M \mid M'',$$

т. е. автомат M неприводим, когда он не может быть разложен в каскадное соединение «меньших» автоматов.

Мы можем строить каскадное соединение автоматов, которые сами представляют собой каскадные соединения. В результате такой композиции получится некоторый линейно упорядоченный набор из n автоматов, в котором входной сигнал j автомата в момент времени t зависит от общего входного сигнала всей системы в момент t и от выходных сигналов первых (j— 1) автоматов в момент времени t.

Для того чтобы привыкнуть к введенным определениям, мы рекомендуем читателю самостоятельно проделать все вычисления при $n = 3$ и заодно убедиться, что каскадное соединение $M_3 \times_{Z'} (M_2 \times \times_{\hat{Z}}^{\hat{\eta}} M_1)$ эквивалентно каскадному соединению $(M_3 \times_{\hat{Z}}^{\hat{\eta}} M_2) \times_{\hat{Z}'}^{\hat{\eta}} M_1$ при специально выбранных отображениях $\hat{Z}'$, $\hat{Z}''$ и $\hat{\eta}$ (см. рис. 4).

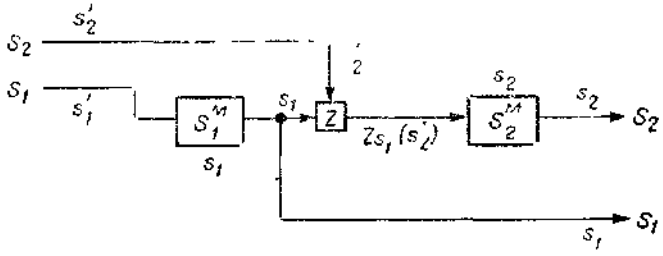


Рис. 4

Допустим, что в определении каскадного соединения двух автоматов 1 мы выбрали $\tilde{X} = S_2 \times S_1$, $\eta(s_2, s_1) = s_1$ и предположили, что

отображение Z не зависит от элемента s_1 в $\tilde{x}$, тогда вместо $Z\{(s_2', s_1'), s_1\}$ можно писать $Z_{S_1}(s_2')$.

Более того, заменим автомат M полугрупповым автоматом S_1^M, а автомат M' — полугрупповым автоматом S_2^M.

Тогда имеем

$$\delta_Z[(s_2, s_1), (s_2', s_1')] = [s_2 Z_{s_1}(s_2'), s_1 s_1'].$$

Операция δ_z становится ассоциативной, если потребовать, чтобы отображение Z обладало свойствами

$$Z_{s_1 s_1'}(s_2) = Z_{s_1}[Z_{s_1'}(s_2)],$$

$$Z_{s_1}(s_2 s_2') = Z_{s_1}(s_2) Z_{s_1}(s_2'), \quad s_1, s_1' \in S_1, \quad s_2, s_2' \in S_2.$$

Этот пункт весьма естественно приводит нас к определению полупрямого произведения полугрупп S_1 и S_2.

3. Определение. Пусть S_1 и S_2 есть некоторые полугруппы, а Z — гомоморфизм S_1 в End (S_2) (моноид эндоморфизмов полугруппы S_2 относительно композиции отображений), т. е. $s_1 \to Z_{s_1}$.

Тогда *полупрямым произведением полугрупп* S_1 и S_2 с гомоморфизмом связи Z называется полугруппа $S_2 \times_Z S_1$, элементами которой служат элементы декартового произведения множеств $S_2 \times S_1$, а умножение определяется соотношением

$$(s_2, s_1)(s_2', s_1') = [s_2 Z_{s_1}(s_2'), s_1 s_1'].$$

Сразу приходим к следующему определению.

4. Определение. Полугруппа S называется *неприводимой,* если для всех полупрямых произведений $S_2 \times_Z S_1$, таких, что $S \mid S_2 \times_Z S_1$, необходимо, чтобы

$$S \mid S_2 \text{ или } S \mid S_1.$$

Два введенных определения сформулированы таким образом, что кажется правдоподобным, что автомат будет неприводим тогда и только тогда, когда неприводима его полугруппа. Однако, как мы увидим дальше, это неверно.

Предположим, что задан каскадный автомат $M' \times_Z M$ с полугруппой $\mathfrak{S}$. Может показаться, что всегда найдется подходящий гомоморфизм $\tilde{Z}$, такой, что

$$S \mid S' \times_{\tilde{Z}} S,$$

где S'—полугруппа автомата M, а S — полугруппа автомата M. Однако это, как правило, неверно, поскольку исходное отображение Z может полностью разрушать мультипликативную структуру полугруппы и поэтому ни одно $\tilde{Z}$ не будет обладать требуемыми гомоморфными свойствами.

Моделируя автомат M с помощью S^M, а автомат M'—с помощью S'^M, мы сможем представить каскадное соединение $M' \times_z M$ схемой, показанной на рис. 5.

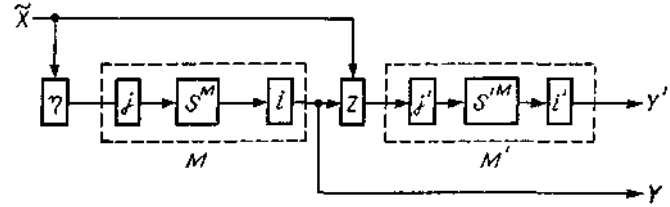


Рис. 5

Пространство состояний этого каскадного соединения (не обязательно приведенное) есть $S' \times S$. Рассмотрим теперь $\hat{S}$, полугруппу преобразований пространства $S' \times S$, индуцированных входными последовательностями.

Пусть t — некоторая входная последовательность, т. е. элемент множества $\tilde{X}^*$. Действие последовательности t на автомат S^M состоит в простом умножении на элемент $Is(t)$ полугруппы S. Однако действие t на второй автомат S'^M будет зависеть от состояния автомата S^M в начале операции. Предположим, что это состояние было s и обозначим искомое действие через $I_{S'}(t)\ (s)$. Таким образом, $I_{S'}\ (t)$ будет элементом полугруппы $F\ (S, S')$ отображений (а не только гомоморфизмов) множества S в S' со знаком композиции вида

$$(f_1 \circ f_2)\ (s) = f_1\ (s) \cdot f_2\ (s).$$

Следовательно, $\hat{S}$ образуется в результате замены элемента t свободной полугруппы входных последовательностей на элемент $(I_{S'}(t),\ I_S\ (t))$ множества $F\ (S,\ S') \times S$. Каково же полугрупповое умножение, индуцированное в этом множестве действием входных последовательностей? Действие последовательности t_1, за которой идет последовательность t_2, дает

$$[I_{S'}\ (t_1),\ I_S\ (t_1)] \cdot [I_{S'}\ (t_2),\ I_S\ (t_2)] = \\ = [I_{S'}\ (t_1 t_2),\ I_S\ (t_1 t_2)]. \qquad (1)$$

Но действие элемента t_1t_2 на автомат S^M определяется просто действием $t_{1,t}$ умноженным на действие t_2, т. е.

$$I_S\ (t_1 t_2) = I_S\ (t_1) \cdot I_S\ (t_2), \quad \textbf{(2)}$$

в то время как действие последовательности t_1t_2 на S'^M при условии, что в начальный момент автомат S^M находился в состоянии s, определяется действием t_1 на автомат S'^M, когда S^M начинает работать из

состояния s, умноженным на действие элемента t_2, когда автомат S^M начинает работать из состояния $s \cdot I_S(t_1)$. Следовательно,

$$I_{S'}(t_1 t_2)(s) = [I_{S'}(t_1)(s)] \cdot [I_{S'}(t_2)(s \cdot I_S(t_1))]. \quad \textbf{(3)}$$

5. Если теперь вернуться к определению полупрямого произведения полугрупп и рассмотреть отображение

$$W : S \to \mathrm{End}\,[F(S, S')],$$

определяемое соотношением

$$W_s(l')(s_1) = l'(s_1 s),$$

то станет ясно, что умножение, определенное на множестве $F(S, S') \times S$, приводит к полупрямому произведению $F(S, S') \times_W S$. Это позволяет сделать вывод, что последняя полугруппа заслуживает особого внимания; мы будем называть ее *узловым произведением полугрупп* S и S' и обозначим через $S' \mathrm{w} S^*$. Заметим, что отображение *W* определено однозначно при заданных полугруппах *S* и *S'*, поэтому его не обязательно указывать в явном виде.

Мы видим, что $\hat{S}$ является подполугруппой узлового произведения полугрупп S и *S'*. Но поскольку полугруппа автомата $M' \times_Z M$ представляет собой действие множества X^* на приведенное пространство состояний, эта полугруппа — гомоморфный образ полугруппы $\hat{S}$. Следовательно, полугруппа каскадного соединения $M' \times_Z M$ необходимо делит узловое произведение полугрупп автоматов *M* и *M'*. Таким образом, хотя в общем случае не существует полупрямого произведения полугрупп S' и *S*, которое делила бы полугруппа $\tilde{S}$, однако справедливо, что $\tilde{S}$ всегда делит одно полупрямое произведение полугрупп *F (S, S')* и S, а именно их узловое произведение.

Обратное утверждение также почти справедливо. Мы можем построить автомат $(S' \mathrm{w} S)^M$ с помощью автоматов S^M и S'^M (не обязательно даже при помощи автоматов *M* и *M')* и тогда он легко моделируется любым приведенным автоматом, полугруппа которого делит $S' \mathrm{w} S$ — полугруппу некоторого каскадного соединения автоматов S'^M и S^M. В самом деле, рассмотрим каскадное соединение $S'^M \times_Z S^M$, для которого $\tilde{X} = S \times F(S, S')$ и $Z_s(s', l') = l'(s)$ (см. рис. 6).

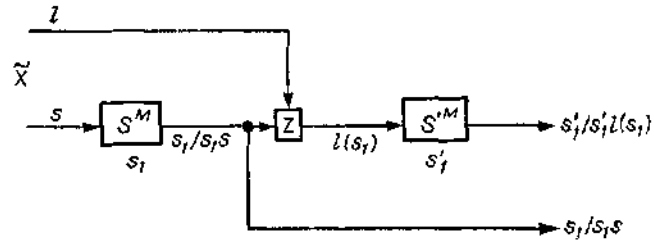


Рис 6

Действие последовательности *(l,* s), за которой идет последовательность (*l′, s′*), на состояние $(s_1,\ s_1')$ приводит к $[s_1'l\ (s_1)\ l'\ (s_1s_2),\ s_1ss']$, поэтому входные последовательности должны перемножаться, как в узловом произведении. Получаем следующий важный результат.

6. Теорема. Рассмотрим конечные полугруппы S и *S′*. Полугруппа $\tilde{S}$ представляет собой полугруппу соединения $S'^M \times_Z S^M$ для некоторого отображения связи *Z* тогда и только тогда, когда $\tilde{S}$ делит узловое произведение $S'\mathrm{w}S$.

В действительности обратное утверждение в теореме 6 может быть сформулировано одновременно в более сильной форме (с использованием только полупрямых произведений) и более слабой форме (с использованием автоматов).

7. Теорема. Для любых полугрупп S_1 и S_2 автомат $(S_2\mathrm{w}S_1)^M$ их узлового произведения может быть получен с помощью полупрямого произведения некоторого числа автоматов S_1^M и S_2^M.

Доказательство. Воспользуемся соотношением

$$S_2\mathrm{w}S_1 = F\ (S_1,\ S_2) \times \mathrm{w}S_1$$

и тем, что $F\ (S_1,\ S_2) = S_2 \times \ldots \times S_2\ [\#\ (S_1)\ \text{раз}]$. На рис. 7 каждому автомату приписано его состояние в момент *t.* Символ s над стрелкой означает, что в автомат подается сигнал s в момент *t,* а символ *s/ss′* над стрелкой означает, что в момент времени *t* выдан сигнал *s,* а в момент времени $t + 1$ — сигнал *ss′*.

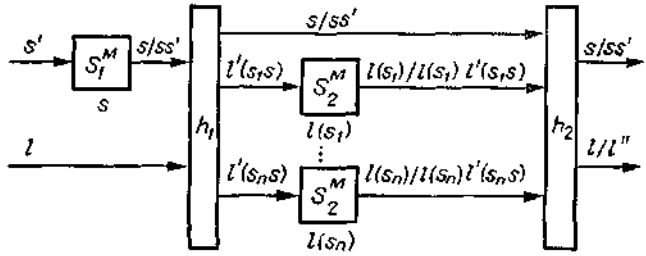


Рис. 7

Отображения h_1 и h_2 определяются с помощью следующих соотношений: $h_1(s, l') = [s, l'(s_1s), \ldots, l'(s_ns)]$, где $S_1 = \{s_1, s_2, \ldots, s_n\}$ (h_2 определено на рис. 7).

Следовательно, полугрупповой автомат $(S_2 w S_1)^M$ можно построить так, как это показано на рис. 7, где

$$s' \in S_1, \quad l' \in F(S_1, S_2), \quad (l', s') \in S_2 w S_1$$

и умножение в узловом произведении определяется соотношением $l''(s_k) = l(s_k)l'(s_ks)$.

Выходной элемент l'' определяется согласно таблице значений, описывающей элемент полугруппы $F(S_1, S_2)$. Входному сигналу (l', s') в момент времени t соответствует выходной сигнал (l'', ss') в момент времени $t + 1$.

Теперь докажем следующую теорему.

8. Теорема. Пусть автомат M неприводим. Тогда его полугруппа S будет неприводимой.

Доказательство. Пусть задан неприводимый автомат M. Утверждение, что полугруппа S делит полупрямое произведение $S_2 \times_Z S_1$, аналогично тому, что автомат S^M моделируется каскадным соединением $S_2^M \times_Z S_1^M$. Так как $M \mid S^M$, то $M \mid S_2^M \times_Z S_1^M$ и в силу неприводимости автомата $M \mid S_1^M$ или $M \mid S_2^M$. Но из делимости автоматов вытекает делимость их полугрупп, поэтому $S \mid S_t$ или $S \mid S_2$. Следовательно, полугруппа S неприводима.

9. Теорема. Если полугруппа S автомата M неприводима, то автомат M является s-неприводимым (автомат M называется s-неприводимым, если каждый раз, когда он моделируется каскадным соединением двух автомагов M_1 и M_2 с полугруппами S_1 и S_2, либо $M \mid S_1^M$, либо $M \mid S_2^M$).

Доказательство. Пусть полугруппа S неприводима. Теперь если $M \mid M_2 \times_Z M_1$, то полугруппа S должна делить узловое произведение $S_2 w S_1$ и, следовательно, делить S_1 или $F(S_1, S_2)$. Если S делит S_1, то

автомат S^M делит S^{M_1}. Если же S делит $F(S_1, S_2)$, то, так как $F(S_1, S_2) \cong S_2 \times \ldots \times S_2$, где число сомножителей равно $\#(S_1)$, полугруппа S в силу неприводимости должна делить S_2, но тогда S^M делит S_2^M. Таким образом, имеем $M \mid S_1^M$ или $M \mid S_2^M$. Доказательство закончено.

10. Итак, *автомат s-неприводим тогда и только тогда, когда неприводима его полугруппа.* Крон и Роудз описали неприводимые полугруппы (соответственно s-неприводимые автоматы) и указали, как другие полугруппы (соответственно автоматы) можно построить из неприводимых с помощью полупрямых произведений и делимости (соответственно каскадных соединений и моделируемости).

Отметим, что неприводимость автоматов не эквивалентна неприводимости полугрупп. Причина этой неэквивалентности заключается в том, что выходные отображения автоматов M_1 и M_2 могут делать их множества выходных сигналов столь малыми, что ни M_1, ни M_2 отдельно не могут моделировать автомат M, в то время как один из автоматов S_1^M и S_2^M может быть достаточно большим и будет моделировать M.

11. Пример. Рассмотрим автомат M, который производит сложение по mod 3; его выходные сигналы совпадают с его состояниями, а переходы в пространстве состояний определяются с помощью следующей таблицы:

	1	2	3
1	2	3	1
2	3	1	2
3	1	2	3

Пусть автомат M_1 получается из M в результате добавления выходной кодирующей функции, которая определяется с помощью следующих соотношений:

$$h(1) = 1,\ h(2) = s,\ h(3) = s.$$

Попробуем смоделировать автомат M, начинающий работать из состояния 1, каскадным соединением автомата M_1, начинающего работать из состояния 1, и автомата M_2, который есть в точности автомат M_1, но только он начинает работать из состояния 2 (см. рис. 8).

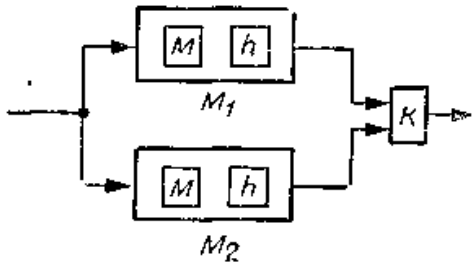


Рис. 8

Читатель может проверить, что если мы определим k с помощью соотношений

$$k\begin{pmatrix}1\\s\end{pmatrix}=1,\quad k\begin{pmatrix}s\\s\end{pmatrix}=2,\quad k\begin{pmatrix}s\\1\end{pmatrix}=3,$$

то каскадное соединение ведет себя так же, как автомат M. Следовательно, $M \mid M_2 \times M_1$ и $s = s_1 = s_2$, но ни одно из соотношений $M|M_2$ или $M \mid M_1$ невозможно, так как множество выходных сигналов одного автомата M_1 слишком мало для того, чтобы его можно было декодировать во все множество выходных сигналов автомата M. Рекомендуем также запомнить, что моделируемость автомата M полугрупповым автоматом S^M была доказана для случая, когда автомат M циклический, т. е. все его состояния достижимы из некоторого начального состояния q.

[Мы используем термин «циклический» в алгебраическом смысле: существует одна образующая $Q = \delta(q, X^*)$, это означает, что граф состояния содержит только циклы.] Тогда если состояние $q' = \delta(q,x)$, то его можно кодировать как состояние *[x]* полугруппового автомата, причем оно не зависит от выбора элемента x, необходимо только, чтобы x удовлетворял соотношению $q' = \delta(q, x)$. Если же автомат M не является достижимым, т. е. пространство состояний представляет собой объединение нескольких орбит, то рассмотренный метод кодирования годится только для одной орбиты. Поэтому для каждой из них потребуется только автомат S^M, а также селектирующий переключатель, показывающий, во что кодируется состояние, принадлежащее орбите соответствующего начального состояния.

Мы воспользовались каскадным соединением для построения композиции автоматов, не содержащей петель. Крон и Роудз рассматривают композицию при помощи последовательных и параллельных со-

единений — частные случаи каскадного соединения. Обратное не совсем верно (см. рис. 9).

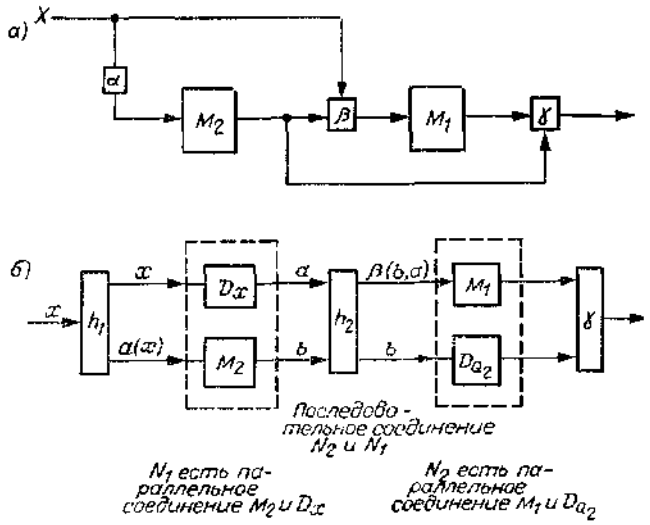


Рис. 9

Дело в том, что в автомат M_1 как элемент каскадного соединения автоматов M_2 и M_1 поступают одновременно выходные сигналы автомата M_2 и элементы множества X, тогда как в случае последовательного соединения автоматов M_2 и M_1 элементы множества X не могут быть получены непосредственно автоматом M_1, сначала они перерабатываются автоматом M_2. Возникающую здесь трудность можно обойти. На рис. 9 видно, что функция вход—выход последовательно-параллельного соединения (б) совпадает с функцией вход—выход каскадного соединения *(а),* сдвинутой на одну единицу времени. Квадратики, обозначенные символами D_X и D_{Q_2}, есть элементы задержки с множествами состояний X и Q_2 соответственно (Q_2— множество состояний автомата M_2). В таком автомате входной сигнал поступает на выход, не изменяясь, но с задержкой на единичный момент времени. Легко видеть, что эти **автоматы** являются **возвратными** (в другой терминологии—**основными комбинаторными автоматами**).

В действительности можно показать, что автоматы задержки непременно появляются при переходе от каскадных соединений к последовательно-параллельным соединениям.Отметим, что разложение

автомата в каскадное соединение заданных автоматов получить значительно легче, чем разложение в последовательно-параллельное соединение. Если неясно, как непосредственно произвести разложение в последовательно-параллельное соединение, то сначала можно произвести разложение в каскадное соединение, а затем воспользоваться приемом, показанным на рис. 9. Таким образом будет получен желаемый результат.

2.3.2. Расширение полугрупп и каскадное соединение автоматов

Известно, что если на полугруппе S задано отношение эквивалентности ρ , обладающее свойством, что из соотношений $x\rho x'$ и $y\rho y'$ вытекает $xy\rho x'y'$, то ρ называется конгруэнтностью. Множество классов эквивалентности S/ρ является полугруппой с законом композиции, определяемым соотношением $[x]\,[y] = [xy]$. Заметим, что когда S есть моноид с единицей 1, то класс $[1]_\rho$ — подмоноид в S, а множество S/ρ — также моноид.

Говорят, что полугруппа *S* есть *расширение полугруппы T при помощи конгруэнтности* ρ, если ρ есть конгруэнтность на полугруппе S и существует элемент *t* в *S,* такой, что $[t]_\rho \cong T$. Если $\Sigma = S/\rho$, то говорят, что S есть *расширение полугруппы T при помощи полугруппы* Σ.

Предположим теперь, что *G* — группа, а ρ — отношение конгруэнтности на *G.* Легко проверить, что класс $H = [1]_\rho$ представляет собой нормальную подгруппу в С и это единственная подполугруппа в G вида $[g]_\rho$. Тогда $K = G/\rho$ есть в точности фактор-группа *G/H.* Если ранее предложенные определения по аналогии перенести на группы, то получим: группа *G* — расширение группы *H* при помощи группы *K,* если $H \lhd G$ и $G/H \cong K$.

Охарактеризуем кратко все расширения группы *H* при помощи группы *K.*

Предположим, что $K \cong G/H$. Тогда каждому элементу группы *K* соответствуют $|H|$ представителей класса *[g],* где *g*—элемент группы *G.* Выберем и зафиксируем по одному такому представителю $\bar{g}$ для каждого класса *[g].*

Так как *H*—нормальная подгруппа, то $\bar{g}h \in H\bar{g}$ для любых элементов $g \in G$ и $h \in H$. Следовательно, отображение

$h \to \bar{g}h\bar{g}^{-1} = h^{[g]}$ (4) является автоморфизмом группы *H.*

Так как элемент $\bar{g}_1\bar{g}_2$ принадлежит классу

$[g_1]\,[g_2] = [g_1 g_2]$,

мы можем определить $([g_1],\ [g_2])$ при помощи соотношения

$$\bar{g}_1\,\bar{g}_2 = ([g_1],\ [g_2])\,\overline{g_1 g_2}. \quad (5)$$

Благодаря этому умножение в группе можно ввести при помощи соотношения

$$(h_1\,\bar{g}_1)\,(h_2\,\bar{g}_2) = h_1\,h_2^{[g_1]}\,\bar{g}_1\,\bar{g}_2 = \\ = h_1\,h_2^{[g_1]}([g_1],\ [g_2])\,\overline{g_1 g_2}. \quad (6)$$

Если определить

$$\beta(h_2\,\bar{g}_2,\ [g_1]) = h_2^{[g_1]}([g_1],\ [g_2]),$$

то это правило позволяет представлять автомат G^{M1} как каскадное соединение автоматов H^M и K^M (см. рис. 10), где

$$\alpha : G \to K : g \to [g],$$

$$\beta : G \times K \to H : (g_2,\ [g_1]) \to (\bar{g}_2\,g_1)\,\overline{[\bar{g}_1\,g_2]}^{-1},$$

$$\gamma : H \times K \to G : (h,\ [g]) \to h\bar{g}.$$

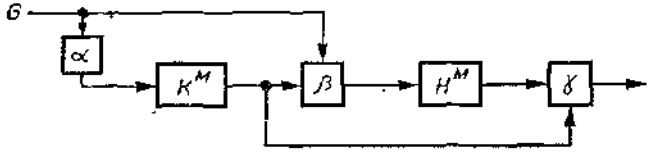


Рис. 10

Соотношения (4)—(6) после ряда дополнительных вычислений приводят к следующему результату (условие (7) требуется для того, чтобы гарантировать ассоциативность расширения).

1. Теорема Шрейера. Предположим, что H—нормальный делитель группы G, а K — фактор-группа, $K \cong G/H$; для каждого смежного класса $u \in K$ выбран представитель $\bar{u} \in G$, тогда автоморфизмы группы H $h \to h^u$ для каждого элемента $u \in K$ и системы факторов (u, u) для $u, v \in K$ определяются при помощи соотношений

$$(h^u)^v = (v, u)\,h^{vu}\,(v, u)^{-1},$$

$$(v, w)^u\,(u, vw) = (u, v)\,(uv, w). \quad \textbf{(7)}$$

И наоборот, если для каждого элемента $u \in K$ задан автоморфизм $h \to h^u$ группы H и если для этих автоморфизмов и системы факторов $\{(u, v) \in H \mid u, v \in K\}$ справедливы условия (7), то элементы $h\bar{u}$, $u \in K$, $h \in H$ с законом умножения

$$h_1\,\bar{u}_1 \cdot h_2\,\bar{u}_2 = h_1\,h_2^{u_1}\,(u_1,\ u_2)\,\overline{u_1 u_2}$$

определяют группу G с нормальной подгруппой (изоморфной) H, кроме того, выполнено соотношение $G/H \simeq K$.

Отметим, что в теории групп удобно использовать элемент $\bar{u}h$ в качестве типичного представителя элемента группы G (т. е. рассматривать левые смежные классы), в то время как в теории автоматов для нас удобнее элементы вида $h\bar{u}$.

Теперь хотелось бы полученные результаты перенести на расширение полугрупп. Но расширения групп обладают важным свойством, которое не выполняется для большинства расширений полугрупп. Речь идет о том, что элементы группы G однозначно определяются заданием элемента из H и элемента из *K* и для каждой такой пары существует элемент группы *G*.

2. Пример. Рассмотрим полугруппу $S = \{1, r_1, \ldots, r_n\}$, содержащую $n+1$ элементов, причем элемент 1 является единичным, а остальные n — правые нули. Пусть ρ — отношение конгруэнтности со следующими классами эквивалентности:

$[1] = \{1\}$; $[r_1] = \{r_1, \ldots, r_n\}$. Тогда $S/\rho \cong \{1, r_1\}$,

но пара полугрупп $T = [1]$ и $\Sigma = \{1, r_1\}$ ничего не позволяет сказать о расширении S полугруппы *T* при помощи Σ. Полугруппы *T* и Σ дают слишком малую часть элементов, которые можно было бы рассматривать как элементы полугруппы S. Неясно, следует ли ограничиваться этой малой частью.

3. Пример. Пусть S — полугруппа, а I — собственный идеал в S (т. е. $I \subset S$, $SIS \subset I$), определим отношение конгруэнтности ρ на полугруппе S при помощи соотношений

$[s] = \{s\}$, если $s \in S - I$;

$[s] = I$, если $s \in I$.

S/ρ обычно называют фактор-полугруппой Риса полугруппы S по mod *I* и записывают как *S/I*. Полугруппа S является расширением полугруппы $T = I$ при помощи $\Sigma = S/I$. В этом случае имеется слишком много пар элементов из *I* и из Σ. Теория расширений, рассматриваемых в этом примере, была развита Клиффордом и Манном.

Следовательно, прежде всего необходимо ограничить класс расширений полугрупп, на которые можно было бы распространить результаты Шрейера.

Соответствующий класс расширений и обобщение на него шрейеровской теории групповых расширений были даны Редей. Несколько модифицированное изложение результатов Редей дано далее.

4. Определение. Моноид S называется *шрейеровским расширением моноида T при помощи моноида* Σ тогда и только тогда, когда:

1) T—подмоноид в S;

2) существует отображение (не обязательно гомоморфизм) $\sigma \to \bar{\sigma}$ моноида Σ в S, такое, что отображение $(t, \sigma) \to t\bar{\sigma}$— взаимно однозначное отображение декартового произведения $T \times \Sigma$ на S (если $s = t\bar{\sigma}$, то можно писать $\bar{s}$ для $\bar{\sigma}$);

3) отношение ρ, при котором тогда $s_1 \rho s_2$ и только тогда, когда $\bar{s}_1 = \bar{s}_2$, является отношением конгруэнтности на S и $S/\rho \cong \Sigma$; изоморфизм задается соотношением η (s) = σ тогда и только тогда, когда $\sigma = \bar{s}$, $[1]_\rho = T$.

Читатель без труда убедится в справедливости следующей леммы.

5. Лемма. Полупрямое произведение $S_2 \times {}_z S_1$ двух моноидов S_2 и S_1 представляет собой шрейеровское расширение S_1 при помощи S_2.

Пусть S — шрейеровское расширение моноида T при помощи Σ. Тогда для каждого элемента σ_1 из Σ элемент $s = \vec{t}\bar{s}$ принадлежит S, элемент $\bar{\sigma}_1 t\bar{s}$ принадлежит $T\overline{\sigma_1 s}$ и поэтому существует единственный элемент $\beta(\sigma_1, s)$, такой, что

$$\bar{\sigma}_1 s = \beta(\sigma_1, s)\,\overline{\sigma_1 s}.$$

Следовательно, если $s_1 = t_1\bar{s}_1$ и $s_2 = t_2\bar{s}_2$, то

$$s_1 s_2 = t_1 \beta(\bar{s}_1, s_2) \cdot \overline{s_1 s_2}.$$

Это приводит к автомату, реализующему S^M в виде каскадного соединения автоматов Σ^M и T^M. Читатель легко убедится в этом, если на рис. 10 заменит K на Σ и H на T, отображения α, β и γ определяются при помощи соотношений

$$\alpha : S \to \Sigma : s \to [s],$$

$$\beta : \Sigma \times S \to T : (\sigma, s) \to \beta(\sigma, s),$$

$$\gamma : \Sigma \times T \to S : (\sigma, t) \to t\sigma.$$

Не ограничивая общности дальнейших рассуждений, можно предположить, что $\bar{1} = 1$.

Для каждых элементов $\sigma\tau \in \Sigma$ по условию (3) $\bar{\sigma}\bar{\tau} \in T\overline{\sigma\tau}$, поэтому должен существовать единственный элемент $(\sigma, \tau) \in T$, такой, что $\bar{\sigma}\bar{\tau} = (\sigma, \tau)\,\overline{\sigma\tau}$.

Так как $\bar{1} = 1$, ясно, что $(\sigma, 1) = (1, \sigma) = 1$.

Вновь для любого элемента t из T и любого σ из Σ элемент $\vec{\sigma}$ принадлежит $T\bar{\sigma}$ и элемент t принадлежит $T\bar{1}$, поэтому элемент $\bar{\sigma}t$ принадлежит $T\bar{\sigma}$.

Следовательно, существует единственный элемент t^σ моноида T, такой, что

$$t^\sigma \bar{\sigma} = \bar{\sigma} t.$$

Заметим, что $1^\sigma = 1$ для всех элементов σ из моноида Σ и $t^1 = t$, так как $\bar{1} = 1$. Можно иначе записать операцию умножения:

$$(t_1 \bar{\sigma_1})(t_2 \bar{\sigma_2}) = t_1 t_2^{\sigma_1} \bar{\sigma}_1 \bar{\sigma}_2 = t_1 t_2^{\sigma_1} (\sigma_1, \sigma_2) \overline{\sigma_1 \sigma_2}.$$

Из закона ассоциативности вытекает, что

$$t_1 t_2^{\sigma_1} (\sigma_1, \sigma_2) t_3^{\sigma_1 \sigma_2} (\sigma_1 \sigma_2, \sigma_3) =$$

$$= t_1 [t_2 t_3^{\sigma_2} (\sigma_2, \sigma_3)]^{\sigma_1} (\sigma_1, \sigma_2 \sigma_3). \quad (8)$$

Положив $t_1 = t_2 = t_3 = 1$, мы получим соотношение

$$(\sigma_1, \sigma_2)(\sigma_1 \sigma_2, \sigma_3) = (\sigma_2, \sigma_3)^{\sigma_1} (\sigma_1, \sigma_2 \sigma_3). \quad (9)$$

Положив $t_1 = t_2 = 1$, $\sigma_3 = 1$, получим

[так как $(\sigma, 1) = (1, \sigma) = 1$]

$$(\sigma_1, \sigma_2) t_3^{\sigma_1 \sigma_2} = (t_3^{\sigma_2})^{\sigma_1} (\sigma_1, \sigma_2). \quad (10)$$

Наконец, положив $\sigma_2 = \sigma_3 = 1$, $t_1 = 1$, получим

$$t_2^{\sigma_1} t_3^{\sigma_1} = (t_2 t_3)^{\sigma_1}, \quad (11)$$

т. е. каждый элемент 0 определяет эндоморфизм моноида T, при котором элемент t переходит в элемент t^σ. В действительности из соотношений (9)—(11) можно получить соотношение (8). Теперь уже имеется все необходимое для того, чтобы стала ясной следующая теорема.

6. Теорема Редей—Шрейера. Пусть задано шрейерово расширение S моноида T при помощи моноида Σ, а соответствие $\sigma \to \bar{\sigma}$ (и $\bar{1} = 1$) определяет выбор представителя Σ класса в моноиде S. Тогда эндоморфизмы моноида T, $t \to t^\sigma$ при каждом $\sigma \in \Sigma$ и система факторов (σ, σ') для $\sigma, \sigma' \in \Sigma$ определяются соотношениями

$$(\sigma_2, \sigma_3)^{\sigma_1} (\sigma_1, \sigma_2 \sigma_3) = (\sigma_1, \sigma_2)(\sigma_1 \sigma_2, \sigma_3);$$

$$(t_3^{\sigma_2})^{\sigma_1} (\sigma_1, \sigma_2) = (\sigma_1, \sigma_2) t_3^{\sigma_1 \sigma_2}; \quad (12)$$

$$(\sigma_1, 1) = (1, \sigma_2) = 1.$$

И наоборот, если для каждого элемента $\sigma \in \Sigma$ задан эндоморфизм $t \to t^\sigma$ моноида T и если все эти эндоморфизмы и система факторов $\{(\sigma, \sigma') \in T \mid \sigma, \sigma' \in \Sigma\}$ удовлетворяют условиям (12), то элементы $t\bar{\sigma}$, $\sigma \in \Sigma$, $t \in T$ и закон композиции

$$t_1 \bar{\sigma}_1 \cdot t_2 \bar{\sigma}_2 = t_1 t_2^{\sigma_1} (\sigma_1, \sigma_2) \overline{\sigma_1 \sigma_2}$$

определяют моноид S, представляющий собой шрейеровское расширение моноида T при помощи моноида Σ.

2.4. Каскадная декомпозиция автоматов при помощи покрытий

2.4.1. Автоматы

Мы обсудим некоторые абстрактные задачи, возникающие при проектировании таких приборов, как счетчики, вычислительные устройства, телефонные переключающие цепи, автоматы для продажи штучных товаров. Для примера опишем действие поста по сбору пошлины на шоссейной дороге. Служащий по посту включает сигнал «движение закрыто» пока не будет внесена плата 5 центов, затем на время проезда автомобиля он включает сигнал «движение открыто». Если автомобиль пытается проехать на сигнал «движение закрыто», служащий поднимает тревогу. Более явно инструкции для служащего приведены на рис. 1. Для того чтобы их выполнять, ему не следует думать и считать, он должен только обнаружить, в какой ситуации находится, прочитать, что написано в соответствующей клетке, соответствующим образом поступить и запомнить, к какой строке следует перейти. Любое условие, заставляющее служащего обращаться к некоторой строке, назовем состоянием служащего. Для того чтобы избежать путаницы, связанной с хронометрированием, установим единицу времени (например, 1/2 с) и потребуем, чтобы служащий выполнял одну инструкцию через каждый единичный интервал времени.

	Ничего не происходит	Внесена плата 5 центов	Автомобиль проходит пост
0	Движение закрыть, перейти к строке 0	Движение закрыть, перейти к строке 1	Поднять тревогу, перейти к строке 0
1	Движение закрыть, перейти к строке 1	Движение закрыть, перейти к строке 2	Поднять тревогу, перейти к строке 0
2	Движение закрыть, перейти к строке 2	Движение открыть, перейти к строке 3	Поднять тревогу, перейти к строке 0
3	Движение открыть, перейти к строке 3	Движение открыть, перейти к строке 3	Движение закрыть, перейти к строке 0

Рис. 1. Управление постом производится следующим образом. В нулевой строке выбирают прямоугольник, определяемый одной из ситуаций, заданной верхней строкой. Дальнейшее действие описывается содержанием этого прямоугольника. В результате мы переходим к некоторой строке, вновь выбираем прямоугольник, определяемый одной из ситуаций верхней строки, и т. д. Работа осуществляется через интервалы времени, равные 1/2 с. (Номер строки

есть в точности количество внесенных пятицентовых монет, номер 3 следует интерпретировать как 3 или более,)

По-видимому, читатель уже подозревает, что мы перешли от служащего к автомату. Это действительно так. Определим **автомат как объект, для которого существуют внешние воздействия, некоторые состояния, некоторые реакции и операция, преобразующая каждую пару состояние—внешнее воздействие в состояние и реакцию в последующий момент времени** (на рис. 1 операция задается при помощи таблицы). Таким образом, эта операция преобразует пару состояние— внешнее воздействие (x_1, u) в пару реакция в последующий момент времени—состояние в последующий момент времени (y, x_2). Мы назовем *у реакцией на воздействие при заданном состоянии* x_1, а состояние x_2 *— преемником* x_1 *относительно и.* Иногда для описания автомата в дополнение к введенным понятиям требуется добавить начальное состояние (например, на рис. 1 это строка 0).

Перейдем от описания автомата, представленного с помощью рис. 1, к описанию его как некоторого электрического прибора. В этом случае удобно, чтобы внешние воздействия, реакции и состояния изображались как строки напряжений в узлах линий. Каждой линии соответствует одно из двух возможных напряжений. На рис. 2 и 3 дано новое описание первоначального примера, представленное как некоторый электрический прибор.

Входные воздействия в вольтах электрического прибора на паре линий	Входные воздействия первоначального автомата
00	Ничего не происходит
01	Внесена плата 5 центов
10	Автомобиль проходит пост

а

Состояния в вольтах электрического прибора на паре линий	Состояние первоначального автомата как номер строки на рис. 1
00	0
01	1
10	2
11	3

б

Реакция в вольтах электрического прибора на паре линий	Реакция первоначального автомата
00	Закрыть пост
01	Открыть пост
10	Поднять тревогу

в

Рис. 2. Пусть электрические входные воздействия в таблице *а* соответствуют расположенным в той же строке входным воздействиям первоначального автомата, аналогичные предположения справедливы для состояний в таблице б и реакций в таблице *в*

		Вход		
		00	01	10
Состояние	00	00, перейти к 00	00, перейти к 00	10, перейти к 00
	01	00, перейти к 01	00, перейти к 10	10, перейти к 00
	10	00, перейти к 10	01, перейти к 11	10, перейти к 00
	11	01, перейти к 11	01, перейти к 11	00, перейти к 00

последующая реакция, последующее состояние

Рис. 3. С помощью рис. 2 мы получаем, что оператор, указанный здесь, соответствует оператору, представленному на рис. 1

Для моделирования действий человека, которые иллюстрирует рис. 1, нам необходим электрический прибор, который, во-первых, преобразует каждую возможную пару состояние—внешнее воздействие (заданную как напряжения) в пару реакция в последующий момент времени—состояние в последующий момент времени (заданные также в терминах напряжений) и, во-вторых, воспринимает второй элемент этой пары как начальное состояние для следующего момента времени. Такой прибор изображен схематически на рис. 4.

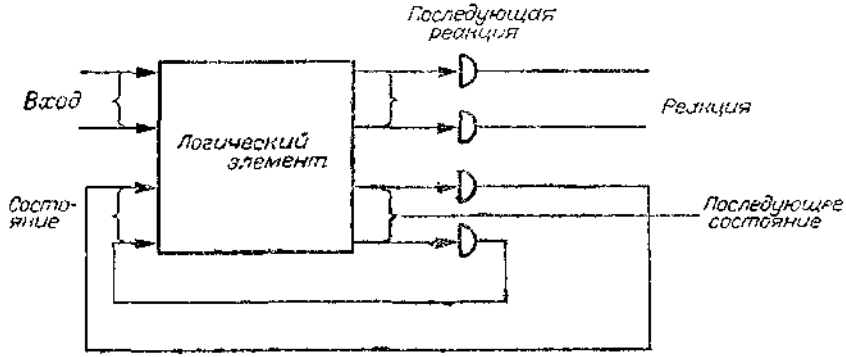


Рис. 4. Прямоугольник, обозначенный символом «логический элемент» при заданных в данный момент времени состоянии и входном воздействии производит (в соответствии с рис. 3) состояние и реакцию в следующий момент времени. Каждый полукруг па рисунке обозначает задержку на единичный момент времени вырабатываемых реакций и состояния по сравнению с заданными (они называются последующей реакцией и последующим состоянием)

Подробное описание элементов этой схемы можно найти в любом учебнике по синтезу логических схем.

Несмотря на то, что в настоящей главе мы стремимся избегать слишком абстрактных понятий, следует определить и как-то назвать переход, осуществленный при помощи рис. 2 и 3, от автомата, программирующего действия оператора, к электрическому автомату. ***Гомоморфизмом* автомата *A* в автомат *A′* называется оператор *h*, который преобразует каждое внешнее воздействие, состояние, реакцию и операцию автомата *A* во внешнее воздействие, состояние, реакцию и операцию автомата *A′* (соответственно) так, что для каждого внешнего воздействия и состояния автомата *A* состояние и реакция в последующий момент времени автомата *A′* получаются в результате применения сначала *h*, а затем операции автомата *A′* и совпадают с состоянием и реакцией автомата *A′*, полученными в**

результате применения сперва операции автомата *A*, а затем оператора *h*. Это устное определение иллюстрируется рис. 5 и 6. (Заинтересованный читатель сможет найти более простое определение.)

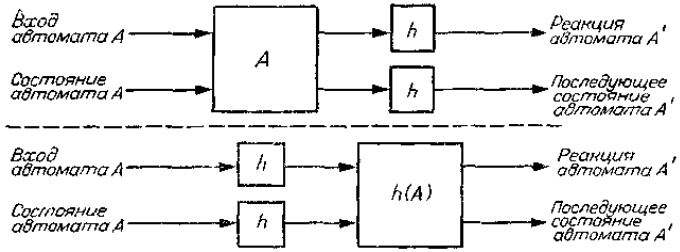


Рис. 5. Прямоугольник, помеченный символом *A,* обозначает действие автомата *A,* прямоугольник, помеченный символом *h(A),*—действие автомата Л' Прямоугольник, помеченный символом /г, обозначает оператор, который преобразует каждое входное воздействие, состояние, реакцию или оператор автомата *A* во входное воздействие, состояние, реакцию или оператор автомата *A' h* представляет собой гомоморфизм тогда и только тогда, когда операторы автоматов,, разделенных пунктирной линией, равны

Символ	Обозначает
$A \to B$	Оператор, преобразующий каждый элемент множества A в некоторый элемент множества B
$A \to B \to C$	Оператор, получаемый в результате применения сначала $A \to B$, а затем $B \to C$
$A \times B$	Множество всех пар элементов (a, b), где a — элемент множества A и b — элемент множества B
$A \times B$ $\downarrow \quad \downarrow$ $C \times D$	Оператор, преобразующий каждый элемент множества $A \times B$ в элемент множества $C \times D$, получаемый в результате применения отображения $A \to C$ на первой координате и $B \to D$ на второй

а

$$\begin{array}{c} X \times U \to Y \times X \\ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ X' \times U' \to Y' \times X' \end{array}$$

Символ	Обозначает
X	Множество состояний автомата A
U	Множество входных воздействий A
Y	Множество реакций для A
$X \times U \to Y \times X$	Оператор автомата A
	и аналогично для автомата A'

б

Рис. 6. Система обозначений, описанная в таблице *а,* все время будет использоваться в дальнейшем. Отображения в диаграмме *б* дают гомоморфизм из А в А' тогда и только тогда, когда диаграмма *б* коммутативна, т. е.

$$\begin{array}{ccc} X \times U \to Y \times X & & X \times U \\ \downarrow \quad \downarrow & = & \downarrow \quad \downarrow \\ Y' \times X' & & X' \times U' \to Y' \times X' \end{array}$$

Легко видеть, что гомоморфизм автомата, программирующего действия оператора, в электрический автомат является обращением гомоморфизма электрического автомата в автомат, программирующий действие оператора (этот последний и описан при помощи рис. 2 и 3).
Любой гомоморфизм, для которого существует обратный гомоморфизм, называется изоморфизмом.
Из рисунков видно, что часто используется композиция операторов (сначала применяется *f*, а затем *g),* что не всегда следует непосредственно вычислять операторы (находить образ элемента *x* при действии оператора *f*) и что редко два оператора, определенные на одном множестве, принимают значения на одинаковых множествах. Для того чтобы не обременять читателя лишними обозначениями, будем оператор,определенный на множестве *A* и принимающий значения в множество *B,* обозначать как $A \rightarrow B$.
Далее нам придется рассматривать необратимые гомоморфизмы, которые несколько состояний автомата *A* переводят в одно состояние автомата *A'.* Это происходит потому, что каждая задача проектирования начинается с изучения автомата A', а заканчивается изучением автомата *A,* обладающего рядом желаемых свойств (например, он имеет двоичные внутренние переменные), и гомоморфизма из *A* на *A'* (здесь «на» означает, что каждое внешнее воздействие, состояние, реакция и операция автомата A' являются образом относительно *h* некоторого внешнего воздействия, состояния, реакции и операции автомата A). Часто требуемое свойство присуще не произвольному автомату A, изоморфному A ', но некоторому автомату A , который больше, чем A ', в том смысле, что существует (необратимый) гомоморфизм из A на A '.

2.4.2. Независимость координат

В этом параграфе мы продолжим рассмотрение автомата A, изоморфного заданному автомату A ' и обладающего некоторыми свойствами, а именно независимостью некоторых координат состояния от его других координат. Иллюстрация того, что мы понимаем под независимостью координат состояния, приведена на рис. 7—9, детальное определение независимости координат состояния читатель найдет на рис. 10 и 11.

		Вход		
		00	01	10
Состояние	000	00; 000	00; 100	10; 000
	100	00; 100	00, 110	10; 000
	110	00; 110	01; 111	10; 000
	111	01; 111	01; 111	00; 000
		последующая реакция, последующее состояние		

Рис. 7. Здесь представлен еще один автомат, изоморфный автомату, описанному в параграфе 1. Заметим, что первая координата последующего состояния определяется входным воздействием и первой координатой текущего состояния; вторая координата последующего состояния определяется входным воздействием и двумя первыми координатами текущего состояния и соответственно третья координата последующего состояния — входным воздействием и тремя первыми координатами текущего состояния

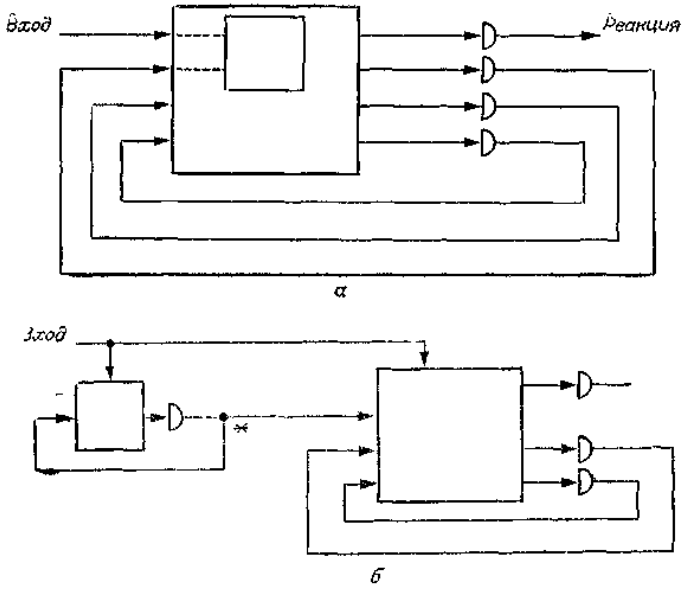


Рис. 8. Так как первая координата последующего состояния определяется входным воздействием и первой координатой текущего состояния, можно считать, что первая координата производится маленьким внутренним квадратом, изображенным на рис. *а.* Этот квадрат можно вынести наружу, как показано на рис. *б.* Линия, ' помеченная звездочкой, необходима, поскольку другие координаты последующего состояния могут зависеть от первой координаты текущего состояния

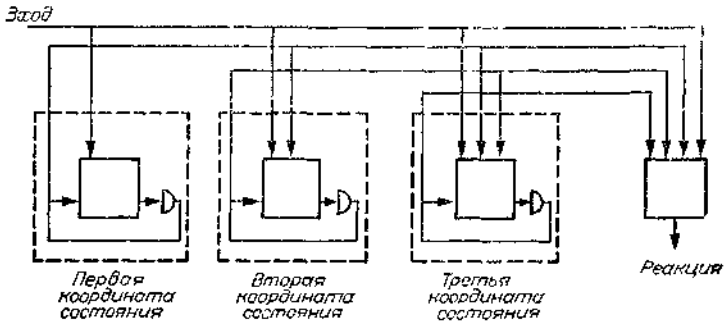


Рис. 9. Продолжая рассуждения, начатые в примечании к рис. 8, мы получаем изображенное здесь представление автомата, заданного на рис 7, Каждый кусок, заключенный в пунктирную линию, сам представляет собой автомат, называемый компонентой. Каждое входное воздействие компоненты составляется из внешнего входного воздействия и состояний всех предыдущих компонент. Каждая последующая реакция компоненты — ее последующее состояние. В результате получается каскадное соединение

n	$f(n)$
1	5
2	2
3	4
4	2
5	5
6	2

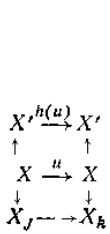
Рис. 10. С этого места *{a, b, c)* обозначает множество с элементами *a, b* и с. Разбиение {{1, 5}, {2, 4, 6), {3}} ранее заданного оператора *f* есть множество прообразов всех элементов из *B* относительно отображения $f : A \rightarrow B$. Прообраз подмножества $C \subset B$ есть множество всех таких элементов $x \in A$, что $f(x) \in C$

$$\begin{array}{ccc} X & \xrightarrow{u} & X \\ \downarrow & & \downarrow \\ X_j & \longrightarrow & X_h \end{array}$$

Символ	Обозначает
u	Произвольное входное воздействие автомата A
X	Множество всех состояний автомата A
$X \xrightarrow{u} X$	Оператор, преобразующий каждое состояние в последующее при входном воздействии u
$X \longrightarrow X_h$	Отображение проекции, преобразующее каждое состояние в его k-ю координату
$X \longrightarrow X_j$	Отображение проекции, переводящее состояние в строку координат, занумерованных строкой j целых чисел

Рис. 11. *k*-я координата последующего состояния автомата *A* определяется набором координат текущего состояния, заиндексированным множеством *J*, если при каждом входном воздействии *u* существует отображение $X_J \dashrightarrow X_k$, для которого диаграмма коммутативна (т. е. $X \to X_J \dashrightarrow X_k = X \xrightarrow{u} X \to X_k$). Следовательно, для каждого *u* образ относительно *u* любого блока разбиения, определяемого $X \to X_J$, является подмножеством некоторого блока разбиения отображения $X \to X_k$ (см. рис. 10)

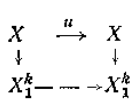
Из рис. 12 мы видим, что отношение зависимости между координатами автомата *A* связано с отношением среди разбиений на множестве состояний автомата *A'*.



Символ	Обозначает
$X' \xrightarrow{h(u)} X'$	Оператор, преобразующий каждое состояние автомата A' в последующее состояние при входном воздействии $h(u)$ (h — гомоморфизм из A в A')
$X \longrightarrow X'$	Оператор, переводящий каждое состояние автомата A в его образ относительно h в автомате A'
другие	См. рис. 11

Рис. 12. Если *h* — изоморфизм, то его можно обратить. Тогда (как на рис. 11) образ относительно *h (u)* любого блока разбиения отображения $X' \to X \to X_J$ есть подмножество некоторого блока разбиения отображения $X' \to X \to X_k$. Следовательно, мы можем разыскивать автомат *A*, изоморфный автомату *A'*, обладающий независимостью некоторых координат от других, подбирая пару разбиений множества *X'*, для которых образ при любом отображении состояния $X' \xrightarrow{h(u)} X'$ любого блока первого разбиения является подмножеством некоторого блока второго разбиения

Эта связь используется в декомпозиции автоматов. Сначала мы разыскиваем соответствующие разбиения на множестве состояний автомата A', а затем пытаемся с их помощью построить координаты для каждого состояния автомата A. При помощи подобного приема мы найдем автомат *A* со специальным видом координатной зависимости: каскадное соединение иллюстрируют рис. 7—9. На рис. 13 показано, как определяется *вложенная последовательность сохраняемых разбиений.*



Символ	Обозначает
$X \to X_1^k$	Отображение проекции, переводящее каждое состояние автомата A в его первые k координат
другие	См. рис 11

Рис. 13. Первые *k* координат состояния преобразуются независимо от остальных координат тогда и только тогда, когда образ относительно $X \xrightarrow{u} X$ любого блока разбиения отображения $X \to X_1^k$ представляет собой подмножество некоторого блока того же самого разбиения. В этом случае мы говорим, что разбиение сохраняется отображением состояния $X \xrightarrow{u} X$. Автомат является каскадным соединением, если для каждого *k* и для каждого *u* разбиение $X \to X_1^k$ сохраняется отображением $X \xrightarrow{u} X$. Между прочим, для каждого *k* разбиение отображения $X \to X_1^{k+1}$ мельче разбиения отображения $X \to X_1^k$. Следовательно, как и в случае, показанном на рис. 12, мы можем искать автомат *A,* изоморфный автомату *A′* и являющийся каскадным соединением, подбирая вложенную последовательность сохраняемых разбиений множества состояний автомата *A′*

Например, для того чтобы найти каскадное разложение для автомата, заданного на рис. 7, представляющего другое описание автомата по сравнению с рис. 1 (программа действия оператора), первое сохраняемое разбиение множества состояний выбирается как {{0}, {1, 2, 3}}, а второе — как {{0}, {1}, {2,3}}.

Мы пришли, таким образом, к следующим задачам:

1) найти вложенную последовательность сохраняемых разбиений;

2) построить каскадный автомат *A,* координатные проекции которого соответствуют этим разбиениям, как показано на рис. 13. Мы отложим рассмотрение первой из этих задач и посвятим параграф 2.4.3 второй.

2.4.3. Назначение координат

Предположим, что заданы автомат *A′* и вложенная последовательность сохраняемых разбиений. Мы хотим построить автомат *A,* гомоморфно отображающийся на автомат *A′* так, что проекции $X \to X_1^k$ соответствуют заданным разбиениям. Временно рассмотрим ситуацию, когда имеется только одно разбиение и, следовательно, две компоненты автомата *A*. Прежде всего заметим, что вопрос с входными воздействиями и реакциями автомата *A* решается очень просто, мы положим их равными входным воздействиям и реакциям автомата *A′* и будем

считать, что гомоморфизм определяет тождественное отображение этих множеств. Однако при рассмотрении множества состояний возникают следующие проблемы:

1) нетрудно построить отображение $X \to X_1$, соответствующее первому разбиению, но спрашивается, как надо задать $X \to X_2$, чтобы проекция $X \to X_1^2$ соответствовала бы разбиению (как правило, отображений $X \to X_2$ существует несколько);

2) если блоки разбиения имеют разную мощность, то операция автомата A определена не полностью, следует как-то ее доопределить.

Рис. 14 иллюстрирует эти проблемы.

Вход

	1	2
1	2	3
2	4	4
3	5	1
4	5	6
5	6	1
6	2	4

последующее состояние

а

Состояние автомата A'	Первая координата автомата A	Вторая координата автомата A
1	0	1
3	0	3
4	0	4
6	0	6
2	1	x
5	1	y

б

Вход

		1	2
Состояние	01	1 x	03
	03	1 y	01
	04	1 y	06
	06	1 x	04
	11	0	0
	13	0	0
	14	0	0
	16	0	0

в

		Состояние первой компоненты, вход			
		0,1	0,2	1,1	1,2
Состояние второй компоненты	1	x	3		
	3	y	1		
	4	y	6		
	6	x	4		
		последующее состояние второй компоненты			

г

Рис. 14. Отображение состояния автомата A' задано в таблице *а*, рассмотрение последующих реакций не входит в нашу задачу и в данном случае опущено.

Ищется автомат *A,* являющийся каскадным соединением двух компонент и гомоморфно отображающийся на *A′,* причем такой, что первая координатная проекция соответствует разбиению {{1, 3, 4, 6}, {2, 5}}. Более простая часть — определение координатных проекций — дана в таблице *б.* Ясно, что не имеет значения, какую из двух величин (0 или 1) мы выберем в соответствии с разбиением для первой координаты. Также не имеет значения, какую из четырех величин мы выберем для второй координаты внутри одного блока, *x* и *y* мы должны выбрать из множества {1, 3, 4, 6}. На основе *б* мы можем расширить частичную таблицу для оператора автомата *A,* показанную в *в.* То, как пополняется эта таблица, определяет оператор второй компоненты, показанный в таблице *г*

Для дальнейшего необходимо ввести несколько определений. Если *U* — множество входных воздействий автомата, то *U** обозначает множество всех (конечных) последовательностей входных воздействий. Заметим, что на множестве *U** можно задать ассоциативную операцию $U^* \times U^* \to U^*$, произведение двух последовательностей есть результат их сочленения. Если *A* — автомат с множеством состояний *X* и множеством входных воздействий *U,* то *отображение переходов* автомата *A* есть операция $X \xrightarrow{v} X$ (где *v* принадлежит *U*),* переводящая каждое состояние s в состояние автомата *A′,* когда на его вход поступает воздействие *v,* а его начальное состояние есть *s.* Если *B* есть подмножество в *X,* переходящее в себя при отображении $X \xrightarrow{v} X$, то соответствующее отображение множества *B* так же обозначается, как
$B \xrightarrow{v} B$.
Вернемся к двум нашим проблемам. Заметим, то если *B* представляет собой подмножество множества состояний автомата *A,* объединяющее все элементы, первая координата которых задана и фиксированна, то гомоморфный образ *B′* множества *B* является блоком сохраняемого разбиения. Если $X' \xrightarrow{v} X'$ отображает множество *B′* в себя, то отображение $X \xrightarrow{v} X$ должно оставлять фиксированной соответствующую первую координату и поэтому *действие отображения* $X \xrightarrow{v} X$ на *вторые координаты должно быть таким же сложным, как* $B' \xrightarrow{v} B'$. Следовательно, верной представляется гипотеза, утверждающая, что для выбранного блока сохраняемого разбиения каждое отображение состояния второй компоненты будет изоморфно некоторому преобразованию $B' \xrightarrow{v} B'$ этого блока в первоначально заданном автомате. Мы уточним это положение в конце данного параграфа.

Ясно, чю в ситуации, изображенной на рис. 14, для второй компоненты автомата *А* требуется одно отображение состояния, соответствующее каждому переходному отображению состояния автомата *А',* при котором множество {1, 3, 4, 6} переходит в себя.
Рис. 15 частично объясняет, почему *не требуется большего числа отображений переходов,* чем это необходимо для второй компоненты. Точный критерий сформулирован в лемме, приведенной далее.

1		3		*y*		6		4		*y*		1
3	$\xrightarrow{2}$	1	$\xrightarrow{1}$	*x*	$\xrightarrow{1}$	4	$\xrightarrow{2}$	6	$\xrightarrow{1}$	*x*	$\xrightarrow{2}$	4
4		6		*x*		4		6		*x*		4
6		4		*y*		6		4		*y*		1

а

		Состояние первой компоненты, вход			
		0,1	0,2	1,1	1,2
Состояние второй компоненты	1	4	3	(6)	(1)
	3	6	1	(4)	(4)
	4	6	6	4	4
	6	4	4	6	1
		последующее состояние второй компоненты			

б

Рис. 15. На диаграмме *а* представлены некоторые из отображений состояния второй компоненты, определяемые последовательностью входных воздействий. Эти преобразования порождаются независимо от того, какие значения мы дадим *x* и *y;* новые преобразования не будут порождены, если мы положим *x* = 4 и *y* = 6. Результирующая операция для второй компоненты показана в табл. *б.*

Мы примем следующие обозначения:
A' —автомат;
X' — множество состояний автомата A';
U — множество входных воздействий автомата *A';*
*U** — множество всех конечных последовательностей входных воздействий;
P — разбиение множества *X',* сохраняемое каждым входным воздействием (см. параграф 2.4.2);
D — подмножество множества *X';*
B — блок разбиения *P (B* может иметь индексы);

$D \xrightarrow{v} B$ — оператор, который переводит каждое состояние *s* из *D* в состояние автомата *A′,* когда он находится в начальном состоянии *s,* а на вход поступает последовательность входных воздействий *v*;
ретракт — B_i есть такое подмножество в D, для которого существуют две такие последовательности w_i и v_i в U^*, что композиция отображений $B_i \xrightarrow{w_i} D \xrightarrow{v_i} B_i$ является тождественным преобразованием множества B_i.

Лемма о ретракте. Если существует множество *D,* для которого каждый блок из *P* представляет собой ретракт в *D,* то автомат *A′*— гомоморфный образ двухуровневого каскадного автомата A. Действие *A* на первые координаты соответствует действию *A′* на *P,* а отображения состояния второй компоненты соответствуют действию на *D* тех отображений состояния автомата *A′,* которые переводят *D* в себя.

Доказательство. Пусть множество входных воздействий автомата *A* есть *U,* а множество состояний — P × *D.* Пусть гомоморфизм *h* определяет тождественное отображение множества *U,* а *P* × *D* отображается в *X′* так, что каждая пара (B_h, s) переходит в образ s относительно $D \xrightarrow{v_k} B_k$. Мы должны теперь для каждого *u* дополнить диаграмму (отображением, обозначенным пунктирной стрелкой)

$$\begin{array}{ccc} P \times D & \dashrightarrow & P \times D \\ \downarrow & & \downarrow \\ X' & \xrightarrow{u} & X' \end{array}$$

так, чтобы она была коммутативной.

На первой координате определить это отображение легко. Положим, что первая координата преобразуется в соответствии с действием, которое отображение $X' \xrightarrow{u} X'$ определяет на разбиении *P.* Для того чтобы продолжить действие отображения на вторую координату, рассмотрим диаграмму, являющуюся ограничением первой (первая координата задается как блок B_j*):*

$$\begin{array}{ccc} \{B_j\} \times D & \dashrightarrow & \{B_k\} \times D \\ \downarrow & & \downarrow \\ B_j & \xrightarrow{u} & B_k \end{array}$$

Здесь $X' \xrightarrow{u} X'$ отображает B_j в B_h. Если мы далее ограничим эту диаграмму на вторую координату, то в силу определения отображения $P \times D \to X'$ получим диаграмму.

$$\begin{array}{ccc} D & \dashrightarrow & D \\ v_j \downarrow & & \downarrow v_k \\ B_j & \xrightarrow{u} & B_k \end{array},$$

которую мы должны дополнить при помощи отображения переходов автомата A', переводящего множество *D* в себя так, чтобы она была коммутативной. Немедленно убеждаемся, что композиция $D \xrightarrow{v_j} B_j \xrightarrow{u} B_k \xrightarrow{w_h} D$ — искомое отображение.

Если каждый блок разбиения *P* конечен, то лемму можно улучшить, потребовав только, чтобы отображение $B \xrightarrow{w} D \xrightarrow{v} B$ было перестановкой (т. е. не обязательно тождественным), так как найдется такое целое число *n,* что отображение $(B \xrightarrow{w} D \xrightarrow{v} B)^n$ будет тождественным. Теперь можно удовлетворить условиям леммы, выбрав новые отображения

$$B \xrightarrow{w'} D = (B \xrightarrow{w} D \xrightarrow{v} B)^{n-1} (B \xrightarrow{w} D).$$

Читатель, должно быть, уже обратил внимание на то, что реакции автомата никак не связаны с назначением координат, а различие между входными воздействиями и последовательностями входных воздействий никак не проявляется. Поэтому можно заменить автомат *A* на так называемый *автомат преобразований* **A**, который получается следующим образом из автомата A:

1) реакция автомата **A** на входное воздействие *u* в состоянии s есть преемник состояния s относительно *u* (т. е. реакция в последующий момент времени совпадает с состоянием в последующий момент времени);

2) множество входных воздействий автомата **A** есть множество всех конечных последовательностей входных воздействий автомата A. (Вполне очевидно, каким образом следует расширить переходное отображение состояния автомата A на новое множество входных воздействий.) Преимущество автоматов преобразований состоит в том, что операция, при которой каждый элемент множества *U** переходит в себя, каждое состояние автомата *A* переходит в блок *P,* его содержащий, отображению переходов автомата A соответствует отображение $P \times U^* \to P$, делающее коммутативной диаграмму

$$\begin{array}{ccc} P \times U^* & \to & X \\ \downarrow & & \downarrow \\ P \times U^* & \to & X \end{array},$$

является гомоморфизмом. Будем называть образ автомата **A** относительно этого гомоморфизма фактор-автоматом $\mathbf{A}/P$ (читается: «**A** mod P»).

Рассмотрим теперь автомат A' и вложенную последовательность сохраняемых разбиений $P_1, P_2, \ldots, P_k$. Для того чтобы построить (если это возможно) каскадный автомат A, гомоморфно отображающийся на автомат A', произведем следующие операции:

1) перейдем к автомату преобразований **A**';
2) построим последнюю компоненту $\mathbf{A}_k$ при помощи (если это возможно) леммы о ретракте, примененной к **A**' и P_k);
3) $\mathbf{A}_k$ теперь предшествует автомату $\mathbf{A}'/P_k$; опять применим (если это возможно) лемму о ретракте, но теперь к $\mathbf{A}'/P_k$ и P_{k-1} для того, чтобы получить последующую компоненту $\mathbf{A}_{k-1}$;
4) продолжим эти действия до тех пор, пока не придем к первой компоненте $\mathbf{A}_0 = A'/P_1$;
5) перейдем от полученного каскадного автомата преобразований к автомату общего вида, ограничивая множество входных воздейст вий и присоединяя реакции. Для того чтобы более кратко описать компоненты, появляющиеся при этой конструкции, для каждого автомата преобразований **A** и подмножества *D* множества состояний **A** зададим подавтомат $\mathbf{A}_D$, множество состояний которого есть D, множество входных воздействий состоит из всех входных воздействий автомата **A**, переводящих *D* в себя, а отображение переходов является, очевидно, результатом ограничения отображения переходов автомата **A**. Пусть P_i/P_{i+1} — разбиение блоков из P_{i+1}, индуцированное разбиением P_i. Если тогда D_i — подмножество разбиения P_{i+1}, для которого каждый блок разбиения P_i/P_{i+1} является ретрактом, то компонента $\mathbf{A}_i$ есть $(\mathbf{A}'/P_{i+1})_{D_i}$. Теперь можно найти любую компоненту $\mathbf{A}_i$ независимо от других компонент.

До сих пор для заданного автомата A' и вложенной последовательности сохраняемых разбиений $P_1, \ldots, P_k$ мы могли построить в любом порядке компоненты каскадного автомата, который гомоморфно отображается на A' при условии, что справедливы предположения леммы о ретракте. Наше положение нельзя признать завидным, так как задан в действительности только автомат A' и, как правило, трудно или даже невозможно найти любое сохраняемое разбиение. Хуже обстоит дело с вложенной последовательностью разбиений, удовлетворяющих условию леммы. В следующем параграфе мы улучшим эту ситуацию, отбросив требование, что автомат *A* строго изоморфен автомату A'; в заключение этого параграфа приведем метод, который иногда полезен при нахождении сохраняемых разбиений.

Рассмотрим автомат преобразований **A** с множеством состояний *X* и множеством входных воздействий *U**. Начнем с разбиения произвольного отображения состояния $X \xrightarrow{v} X$ (см. рис. 11); рассмотрим множество всех разбиений для отображений $X \xrightarrow{w} X \xrightarrow{v} X$, где отображение $X \xrightarrow{v} X$ фиксированно, а *w* пробегает все множество U*; образуем пересечение всех этих разбиений. Полученное в результате раз-

биение будет сохраняемым, если блок состояний B стягивается к единственному состоянию при отображениях $X \xrightarrow{w} X \xrightarrow{v} X$ для всех w, т. е. он является образом B при отображении $X \xrightarrow{u} X$, где u — любой элемент из U^* (если это не так, могут возникнуть неприятности при отображении $X \xrightarrow{u} X \xrightarrow{w} X \xrightarrow{v} X$). Для примера, который иллюстрирует рис. 16, Лиу показал, что, когда автомат можно построить в виде каскадного соединения элементов задержки, разбиения этого типа будут обеспечивать построение.

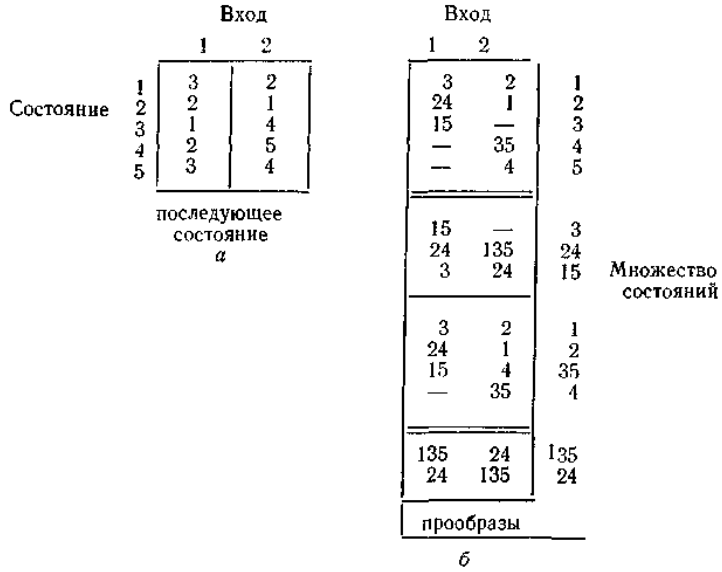


Рис. 16. Разбиения преобразований состояния автомата в таблице *а* вычисляются с помощью возвращения к таблице *б*. Пересечение разбиений, порожденных из {{1, 3, 5), (2, 4)}, даст сохраняемое разбиение {{1, 3, 5}, {2, 4)); пересечение разбиений, порожденных из {{1, 5}, {2, 4}, {3}}, дает сохраняемое разбиение {{1, 5}, {2, 4}, {3}}.

В других случаях метод обычно приводит к разбиению X на отдельные элементы.

2.4.4. Покрытия

В следующих двух параграфах мы рассматриваем только автоматы преобразований (это предположение не ограничивает общность

результатов); однако мы сохраним обозначение *U* для множества входных воздействий, а *U** — для множества последовательностей входных воздействий. Предположим, требуется расширить возможности нахождения каскадного автомата, гомоморфно отображающегося на заданный автомат, с тем чтобы они распространялись и на ситуацию, описанную рис. 17.

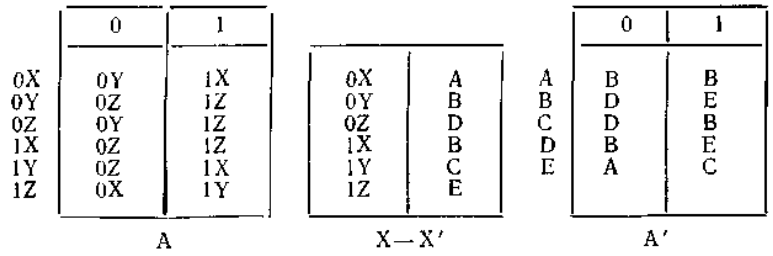


Рис. 17. Гомоморфизм, который действует на состояния посредством отображения $X \to X'$, указывает преобразование каскадного автомата *A* на автомат *A′*. Если, как в параграфе 2.4.3, провести разбиение первого координатного отображения $X \to X_1$ через гомоморфизм, то получим не разбиение, а совокупность перекрывающихся блоков $C = \{(A, B, C), (B, C, E)\}$. Заметим, что поскольку при отображении $X \to X'$ множество *X* переходит на *X′*, каждое состояние из *X′* будет элементом некоторого блока из С, и поскольку диаграмма

$$\begin{array}{ccc} X' & \xrightarrow{u} & X' \\ \uparrow & & \uparrow \\ X & \xrightarrow{u} & X \\ \downarrow & & \downarrow \\ X_1 & \xrightarrow{u} & X_1 \end{array}$$

коммутативна (напомним рис. 12), образ относительного любого отображения состояния любого блока из С представляет собой подмножество некоторого блока из *С*. Любая совокупность блоков состояний, удовлетворяющая первому положению, называется **покрытием**. Любая совокупность, удовлетворяющая также второму положению, называется **сохраняемым покрытием** (также, как сохраняемые разбиения на рис. 13)

Пример и определения, данные здесь, предполагают обобщение имеющейся процедуры на сохраняемые покрытия вместо сохраняемых разбиений. Оказывается, что лемма о ретракте без всяких изменений переносится на этот, более общий случай, в то же время, хотя каждое отображение состояния индуцирует однозначное преобразование блоков для сохраняемого разбиения, это неверно для сохраняемого покрытия (см. рис. 18).

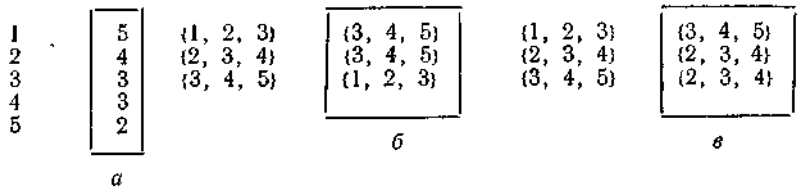


Рис. 18. Если *C* — сохраняемое покрытие множества состояний *X*, мы говорим, что отображение $C \to C$ соответствует отображению состояния $X \to X$, если $C \to C$ переводит каждый блок B из C в блок, который содержит образ *B* при отображении $X \to X$. Преобразования блоков, показанные в таблицах *б* и *в*, соответствуют отображению состояния в таблице *а*. В диаграммах мы пишем $C \sim C \to X \to X$ для $C \to C$, соответствующего $X \to X$.

Следующая лемма позволяет перейти к покрытиям.

Лемма о сохранении однозначности. Если *G* — группа перестановок множества *X*, а C — покрытие множества *X*, сохраняемое группой *G* и такое, что ни один блок из *C* не будет собственным подмножеством никакого другого блока из C, то каждому элементу *g* из *G* соответствует только одно преобразование *g'* покрытия *C*, причем *g'* является перестановкой.

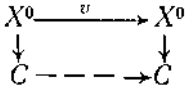
Доказательство. Достаточно показать, что для каждого блока *B* из *C* множество *g(B)* действительно представляет собой элемент *C*, а не подмножество некоторого блока. Предположим противное, т. е. что *g(B)* — собственное подмножество блока *B'*, тогда $B = g^{-1} g(B)$ — собственное подмножество g^{-1} (B'). Но так как *C* — сохраняемое покрытие, $g^{-1}(B')$ будет подмножеством некоторого блока *B''*, но тогда и *B* будет представлять собой собственное подмножество блока *B''*, что противоречит условию леммы. Можно пользоваться теми же соображениями для доказательства того, что преобразование *g'* является перестановкой.

Как известно, мы отказались от сохраняемых разбиений из-за того, что их слишком мало, но в то же время сохраняемых покрытий как-будто бы слишком много. В самом деле, любой набор блоков *C* можно превратить в сохраняемое покрытие, добавив к нему образы относительно всех отображений состояния всех блоков покрытия, а также в качестве блоков те состояния, которые не вошли в построенное покрытие, в этом случае следует отбросить те блоки, которые являются собственными подмножествами других. Наша задача состоит в том, чтобы найти для сохраняемых покрытий аналог вложенной последовательности сохраняемых разбиений и с его помощью построить каскадное соединение с компонентами, по возможности наиболее простыми. Один тип компоненты подсказывается примером,

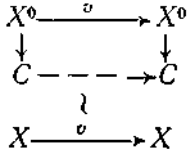
изображенным на рис. 17. Здесь каждое отображение состояния первой компоненты переводит все ее состояния (1 и 0) в одно состояние (1 или 0). Такое отображение состояния называется **возвратным**. Однако из компонент, все отображения состояния которых возвратные, нельзя построить (при помощи каскадного соединения) даже счетчиков. Тем не менее для построения любого автомата с конечным множеством состояний достаточно компонент, все отображения состояния которых возвратные или перестановочные (далее такие автоматы называются перестановочно-возвратными). Вся оставшаяся часть параграфа будет посвящена доказательству этого факта.

По аналогии с разбиениями мы назовем покрытие C_2 *измельчением* покрытия C_1 (соответственно C_1 — *укрупнением* покрытия C_2), если каждый блок покрытия C_2 представляет собой подмножество некоторого блока покрытия C_1; если, кроме того, C_1 не является измельчением C_2, то C_2 называется *собственным измельчением покрытия* C_1.

Для покрытий также существует объект, аналогичный фактор-автомату в случае разбиений. Пусть заданы автомат преобразований A и его покрытие C, сохраняемое при отображениях состояния A. Тогда говорят, что *автомат преобразований A° сообщает положение автомата A в покрытии C,* если A° имеет такое же множество входных воздействий, как A, и если существует оператор $X^\circ \to C$ из множества состояний автомата A° на C, для которого при любом входном воздействии v диаграмму



можно дополнить отображением $C \dashrightarrow C$ до коммутативной. Причем отображение $C \dashrightarrow C$ соответствует отображению состояния $X \xrightarrow{v} X$ автомата A. Таким образом, мы получим суммарную диаграмму



Теперь у нас есть все необходимое для доказательства следующей теоремы.

Основная теорема. Для каждого конечного автомата A' существует каскадный автомат A, гомоморфно отображающийся на A'. Компоненты автомата A перестановочно-возвратные, кроме того, группа перестановок каждой компоненты, порождаемая перестановочными

отображениями состояния, является гомоморфным образом подгруппы из полугруппы отображений состояния автомата A'.

Доказательство. (Первая часть теоремы доказывается легко. Читатель может восстановить доказательство самостоятельно, рассмотрев последовательность покрытий C_k, где C_k есть совокупность всех подмножеств из X, содержащих менее k состояний. Для того чтобы доказать более трудную часть теоремы, нужно получить условия, при которых можно применить лемму о ретракте, позволяющую исследовать группы перестановок. Из-за этого приходится вступить на довольно сложный путь выбора покрытий.) Доказательство следует вести по индукции, но на каждом стандартном шаге ее возникают серьезные технические трудности и поэтому приходится прибегать к искусственным методам. Покажем, что для каждого автомата A' с сохраняемым покрытием C_1 и автомата A_1, который сообщает положение автомата A' в покрытии C_1, можно построить перестановочно-возвратный автомат A_2, группа которого будет гомоморфным образом подгруппы из полугруппы отображений состояния автомата A', и автомат A, являющийся каскадным соединением автоматов A_1 и A_2 и сообщающий положение автомата A' в некотором сохраняемом покрытии C_2. Причем C_2 есть собственное измельчение покрытия C_1 (при условии, что покрытие C_1 не состоит целиком из одноэлементных множеств). Наметим план доказательства. Во-первых, построим собственное измельчение C_2 покрытия C_1, во-вторых, найдем множество состояний автомата A_2 и оператор $X \to C_2$ так, чтобы автомат A сообщал положение автомата A' в C_2, в-третьих, определим отображение состояния автомата A_2, чтобы он был перестановочно-возвратным, и так, чтобы A сообщал положение A' в C_2.

Мы построим покрытие C_2, выбрасывая некоторые блоки из покрытия C_1. Для того чтобы определить, какие блоки следует выбрасывать, упорядочим подмножества из X' следующим образом: B_1 предшествует B_2, если B_2 — образ множества B_1 при некотором отображении состояния. Будем называть блоки B_1 и B_2 *подобными,* если каждый из них предшествует другому; блок B_1 будем называть *начальным элементом* покрытия C_1, если каждый блок покрытия C_1, предшествующий B_1, подобен элементу B_1 покрытия C_1 и если никакой блок из C_1 не содержит большего числа элементов, чем B_1. Теперь построим покрытие C_2 при помощи C_1, заменяя каждый блок B в начальном классе подобия на покрытие блока B, которое получается следующим образом:

1) выбираем каждое собственное подмножество блока B, которое одноэлементно или представляет собой образ некоторого блока покрытия C_1,

2) выбрасываем блоки — собственные подмножества других блоков, принадлежащих *B*.

Заметим, что все выброшенные блоки покрытия C_1 — ретракты друг друга; далее каждый блок покрытия C_2, не являющийся элементом C_1, представляет собой собственное подмножество выброшенного блока покрытия C_1. Если блок A покрытия C_2 есть подмножество выброшенного блока *P* покрытия C_1, а *R* — некоторый другой выброшенный блок, то существует блок *B* покрытия C_2, принадлежащий *R* и такой, что отображения, делающие *P* и *R* ретрактами друг друга, делают также блоки *A* и *B* ретрактами друг друга.

Выберем теперь один из выброшенных блоков B_1 и обозначим через *C* набор блоков, которые заменили B_1 в новом покрытии. Положим в качестве пространства состояний автомата A_2 множество C, тогда *X* есть $C_1 \times C$. Так же, как в лемме о ретракте, для каждого выброшенного блока B_i выберем отображения состояния $B_1 \xrightarrow{v_i} B_i$ и $B_i \xrightarrow{w_i} B_1$, чтобы композиция $B_i \xrightarrow{w_i} B_1 \xrightarrow{v_i} B_i$ была тождественным отображением.

Пусть A — автомат, сообщающий положение автомата A' в покрытии C_2 посредством оператора $C_1 \times C \to C_2$, который переводит каждую пару (B_i, E) из $C_1 \times C$:

1) в B_i, если B_i принадлежит C_2 или 2) в образ множества *E* при отображении $X \xrightarrow{v_i} X$ в противном случае. Заметим, что $C_1 \times C$ переходит при этом отображении на C_2 (что следует из замечания, сделанного в предыдущем параграфе).

Множество входных воздействий автомата A_2 есть $C_1 \times U^*$ (где U^* — множество входных воздействий автомата A'); для каждого элемента (B_j, u) из множества $C_1 \times U^*$ мы должны указать перестановочное или возвратное отображение пространства состояний. Пусть автомат A_1 переходит из состояния B_j в состояние B_k при входном воздействии *u*, а автомат A_2 — в состояние *F* из состояния *E* при входном воздействии (B_j, u).

Случай 1. B_j и B_k представляют собой выброшенные блоки, тогда лемма о ретракте гарантирует существование единственного отображения (обозначенного пунктиром), делающего следующую диаграмму коммутативной:

$$\begin{array}{ccc} B_1 & \dashrightarrow & B_1 \\ {\scriptstyle v_j}\downarrow & & \downarrow{\scriptstyle v_k} \\ B_j & \xrightarrow{u} & B_k \end{array}.$$

В силу леммы о единственности этому отображению соответствует однозначно определенная перестановка или возвратное отображение.

Благодаря этому отображению состояния автомата A_2 мы видим, что автомат А сообщает положение автомата А' в покрытии C_2.
Случай 2. B_j принадлежит покрытию C_2, а B_k — выброшенный блок; пусть F содержит образ блока B_j относительно отображения $X' \xrightarrow{u} X' \xrightarrow{\omega_k} X'$. Благодаря этому возвратному отображению автомата A_2 мы видим, что автомат А сообщает положение автомата A' в покрытии C_2.
Случай 3. B_k принадлежит покрытию C_2; положим F равным E. Так как один B_k сообщает положение автомата A' в покрытии C_2, выбор элемента F не имеет значения.
Это завершает доказательство одного шага индукции. Начиная индукцию с тривиального покрытия $\{X'\}$, мы получаем последовательность покрытий и перестановочно-возвратных автоматов, которая обрывается, когда мы приходим к покрытию, состоящему из одноэлементных множеств, тем самым автомат A' представлен как каскадное соединение перестановочно-возвратных автоматов. Вновь рассматривая случай 1 и леммы о ретракте и единственности, мы видим, что каждая появляющаяся здесь группа перестановок представляет собой гомоморфный образ группы всех перестановок некоторого подмножества D множества состояний автомата A, порожденных элементами из U^*. На этом заканчивается доказательство основной теоремы.
Конструкция, предложенная в теореме, оказывается неэффективной в том смысле, что каскадное соединение будет слишком длинным. Читатель, желающий повысить эффективность, должен переработать доказательство шага индукции, изменив требование, чтобы все невозвратные отображения переходов автомата A_2 были однозначно определены. (Указание: вместо того чтобы отбрасывать все блоки, подобные начальному D при переходе от C_1 к C_2, попробуйте отбросить все ретракты D). Предложенная конструкция допускает более быстрое разложение в каскадное соединение, у которого только первые несколько компонент такие же, как раньше. Конечные компоненты также перестановочно-возвратные, но в них нет лишних подгрупп.

2.4.5. Перестановочные автоматы

Теорема, доказанная в предыдущем параграфе, позволяет разложить любой автомат в каскадное соединение перестановочно-возвратных компонент. Теперь попытаемся представить произвольный перестановочно-возвратный автомат в виде каскадного соединения более

простых элементов. Сначала исследуем возможность каскадной декомпозиции только перестановочных автоматов, а затем—как обобщить полученные результаты на перестановочно-возвратные автоматы. В дальнейшем мы будем пользоваться такими понятиями, как группа, нормальный делитель, смежный класс, фактор-группа, гомоморфизм групп .

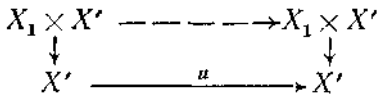
Если A′ — перестановочный автомат, отображения состояния которого порождают группу G, и если H — нормаиьный делитель группы G, а G/H — соответствующая фактор-группа, то существует автомат A, представляющий собой каскадное соединение автоматов A_1 и A_2 и гомоморфно отображающийся на A′. Отображения переходов первой компоненты A_1 этого соединения порождают группу G/H, отображения переходов второй компоненты A_2 порождают группу H.

Доказательство Для того чтобы доказать это утверждение, рассмотрим автомат A_2 с таким же множеством состояний, как у автомата *A′,* по отображения переходов автомата A_2 будут принадлежать H. Всякий раз, когда автомат A_2 не сможет имитировать действие автомата *A′* (а это возможно, так как отображения переходов *A′* могут не принадлежать *H),* положение будет спасать автомат A_1. Его состояния— это такие перестановки из группы *G,* которые преобразуют «неверные» состояния автомата A_2 в верные состояния автомата A'. Таким образом, в качестве множества состояний автомата A_1 следует взять набор представителей смежных классов в *G/H,* отображения переходов автомата A_1 будут порождать фактор-группу *G/H.*

Пусть множество входных воздействий первой компоненты A_1 автомата A совпадает с множеством входных воздействий *U** автомата A', а множество состояний первой компоненты A_1 есть совокупность представителей по одному из каждого левого смежного класса группы *G* по *H.* Отображение переходов

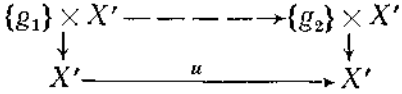
$$X_1 \times U^* \to X_1$$

автомата A_1 преобразует каждую пару $(X' \xrightarrow{g} X', u)$ в представитель класса смежности, содержащего композицию $X' \xrightarrow{g} X' \xrightarrow{u} X'$. Как уже отмечалось, множество состояний второй компоненты A_2 автомата A совпадает с множеством состояний *X′* автомата A'. Пусть гомоморфизм из *A* в *A′* состоит из тождественного отображения на *U** и отображения $X_1 \times X' \to X'$, которое переводит каждую пару $(X' \xrightarrow{g} X', s)$ в *g (s).* Теперь мы должны для любого элемента *u* из *U** найти такое отображение $X_1 \times X' \dashrightarrow X_1 \times X'$, чтобы диаграмма

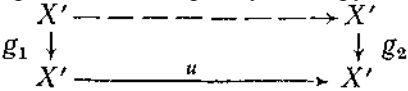


была коммутативна.

Так же, как в доказательстве леммы о ретракте, рассмотрим сначала диаграмму, являющуюся ограничением первой, и определим действие на вторую координату (первая координата есть g_1):



Пользуясь определением отображения $X_1 \times X' \to X'$ и далее ограничивая диаграмму на вторую координату, получим



Для последней диаграммы требуемое отображение легко находится, оно представляет собой композицию $X' \xrightarrow{g_1} X' \xrightarrow{u} X' \xrightarrow{g_2^{-1}} X'$. Так как по определению отображения переходов автомата A_1 элемент g_2 представляет класс смежности, содержащий отображение $X' \xrightarrow{g_1} X' \xrightarrow{u} \xrightarrow{u} X'$, предшествующее отображение, делающее диаграмму коммутативной, принадлежит *H.* Тем самым сформулированное утверждение полностью доказано.

Рассмотренная конструкция впервые была разработана Фробениусом на рубеже XX века в теории представлений групп. В 1962 г. Крон установил, что она может быть применена в теории автоматов.

Заметим, что каскадный автомат *A* возвращается в состояние, соответствующее состоянию s автомата *A'* (при любом начальном состоянии), когда: 1) первая координата остается постоянной и 2) автомат A_2 переходит в состояние, образ которого относительно преобразования, определяемого автоматом A_1, есть s. В частности, если мы выберем в качестве *H* группу, состоящую из одного тождественного преобразования, то увидим, что произвольный перестановочно-возвратный автомат можно построить как каскадное соединение группового автомата и автомата, каждое отображение переходов которого либо возвратное, либо тождественное. Групповой автомат можно разложить, последовательно применяя конструкцию Фробениуса, на групповые автоматы, группы которых не имеют собственных гомоморфизмов (**гомоморфизм называется собственным, если он не является тождественным или тривиальным**). Тождественно-возвратный автомат можно разложить при помощи кодирования его состояний

двоичными числами в параллельное соединение тождественно-возвратных автоматов с двумя состояниями. Эти соображения приводят к теореме Роудза и Крона.

Каждый конечный автомат можно разложить в каскадное соединение автоматов простых групп и тождественно-возвратных автоматов с двумя состояниями.

Позднее будет показано, что нельзя обойтись меньшим множеством компонент. Заметим, что любое каскадное соединение перестановочных автоматов снова будет перестановочным автоматом, поэтому для построения неперестановочного автомата требуется по крайней мере одна неперестановочная компонента. Определим теперь, какие перестановочные компоненты нужны для построения произвольного заданного автомата. Дадим некоторые предварительные определения.

Если $G = A_0 \supset A_1 \supset A_2 \supset \ldots \supset A_n$ есть цепочка групп, в которой каждая A_i представляет собой максимальный нормальный делитель в группе A_{i-1}, то такая цепочка называется композиционным рядом группы G, а фактор-группы A_{i-1}/A_i называются композиционными факторами группы G. (Согласно теореме Жордана — Гельдера между факторами любых двух композиционных рядов можно установить такое взаимно однозначное соответствие, что соответствующие факторы будут изоморфны.)

Если А — произвольная каскадная декомпозиция автомата A' и К — произвольный композиционный фактор группы перестановок подавтомата в A', то К является композиционным фактором группы перестановок подавтомата некоторой компоненты автомата А.

Мы не будем давать подробного доказательства этого важного результата, а лишь укажем его основные моменты и потом их обсудим. Пусть автомат A' — гомоморфный образ каскадного автомата А. Выберем некоторый перестановочный подавтомат B' автомата A' и найдем (1) перестановочный подавтомат B автомата A, гомоморфный образ которого—автомат B'; затем отметим, что B представляет собой каскадное соединение и каждая его компонента есть подавтомат соответствующей компоненты в автомате A. Пусть автомат B^- получается из автомата B в результате отбрасывания последней компоненты. Рассмотрим соответствующий гомоморфизм из группы автомата B в группу автомата B^-. Заметим (2), что полугруппа последней компоненты автомата B должна содержать подгруппу, представляющую собой ядро этого гомоморфизма. Пользуясь индукцией, получаем последовательность групп $B = A_0 \supset A_1 \supset \ldots \supset A_n$, причем полугруппа i-й компоненты содержит в качестве подгруппы фактор A_{i-1}/A_i. Переходя от этой по-

следовательности к композиционному ряду, находим, что каждый композиционный фактор группы автомата *B* — композиционный фактор перестановочного подавтомата в некоторой компоненте автомата *B*. Но каждая компонента автомата *B* — подавтомат в соответствующей компоненте автомата *A*. Каждый композиционный фактор группы автомата *B′* — также композиционный фактор группы автомата *B* (3). Следовательно, каждый композиционный фактор группы автомата *B′* является также композиционным фактором группы перестановочного подавтомата в некоторой компоненте автомата *A,* что и требовалось доказать.

1. Пусть С будет прообразом автомата B' относительно отображения $A \to A'$. Найдем перестановочный подавтомат *B* в *C,* образ которого при $A \to A'$ также есть *B′*. Для каждого отображения переходов $X \xrightarrow{u} X$ автомата *A* определим *область значений* $X \xrightarrow{u} X$ как образ множества *X* при отображении $X \xrightarrow{u} X$, разбиение отображения $X \xrightarrow{u} X$ — как множество прообразов одноэлементных множеств относительно отображения $X \xrightarrow{u} X$, ранг отображения $X \xrightarrow{u} X$—как число состояний в образе (равное числу блоков в разбиении). Пусть *C′* — подавтомат автомата *C,* имеющий такое же множество состояний, как *C,* и имеющий в качестве входных воздействий такие *u,* для которых ранг отображения $X \xrightarrow{u} X$ минимален. Среди отображений переходов автомата *C* существует такое отображение *e,* что его последовательное применение снова дает *e,* т. е. $(e \cdot e = e)$. Для любого отображения переходов *t* автомата С найдутся такие степени, что $t^p = t^{p+k}$ и тогда для всех положительных целых *m* и для всех $r > p$ $t^r = t^{r+mk}$. Если мы выберем $r = mk$ = наименьшее кратное *k,* превосходящее *p,* то $t^r = t^r \cdot t^r$. Отображение $A \to A'$ должно переводить *e* в единицу группы автомата *B′,* поскольку никакие другие перестановки не могут быть равны своему квадрату. Следовательно, для каждого отображения перехода *t* автомата С $A \to A'$ должно отображать *t* и $e \cdot t \cdot e$ в одно и то же отображение переходов автомата *B′*. Обозначим через *C″* подавтомат автомата С, для которого входными воздействиями являются такие *u,* что отображение $X \xrightarrow{u} X$ представляется в виде $e \cdot t \cdot e$, тогда $A \to A'$ все еще отображает *C″* на *B′*. Все отображения переходов автомата *C″* имеют такое же разбиение, как у отображения *e,* и такую же область значений, как у отображения *e.* Поэтому можно построить перестановочный подавтомат *B,* множество состояний которого есть эта область значений, такой, что $A \to A'$ отображает *B* на *B′*.

2. Пусть *B* управляется всеми такими входными воздействиями *u,* для которых $X \xrightarrow{u} X$ принадлежит ядру гомоморфизма (группа автомата *B* → группа автомата *B*⁻*).* Несмотря на то что отображения переходов автомата *B* «выметают» *H,* состояния всех компонент, кроме последней, остаются неизменными. Следовательно, одна последняя компонента должна иметь достаточно отображений переходов для того, чтобы устранить Н.
3. Пропускаем композиционный ряд группы *G* через ядро гомоморфизма $G \to G'$.

Некоторые методы, рассмотренные в настоящей главе, удалось разработать благодаря существованию аналогичных классических результатов. Так, сохраняемые разбиения, когда действия преобразований на них перестановочны, являются, по существу, системами *импримитивности,* которые Э. Галуа рассматривал примерно в 1830 г. Основная конструкция параграфа 2.4.5 в матричной форме известна с 1900 г. как *фробениусовское мономиальное представление.* Аналогом параллельного соединения для линейных динамических систем является приведение к нормальной форме. Для координат это будут главные колебания. Соответствующая теория была развита К. Жорданом примерно в 1880 г. Для нас, однако, эти нормальные координаты — причина некоторой неудовлетворенности. Казалось бы, они должны появляться в линейном аналоге теории каскадной декомпозиции, но пока еще не ясно, как их можно получить с помощью методов, рассмотренных ранее. Трудность состоит в том, что в случае *параллельной* декомпозиции приходится прибегать к более мощному аппарату.

Обсудим теперь вопрос об эффективности каскадной декомпозиции. Для заданного каскадного разложения на перестановочно-возвратные компоненты определим сложность разложения как максимальную длину цепочек координат, для которых компонента каждой координаты в цепочке имеет нетривиальную группу перестановок, зависящую от предыдущих координат в цепочке. Определим сложность автомата как минимум по всем каскадным разложениям сложностей разложений. Легко показать, что сложность автомата зависит только от его. полугруппы.

2.5. Основная теорема декомпозиции в алгебраической теории автоматов

2.5.1. Основные определения

В этой главе мы рассматриваем теорию автоматов с чисто алгебраической точки зрения.

1. Определение. Пусть A и B — некоторые множества. *Автоматом* называется любая функция $f: \Sigma A \to B$. *Полугруппой автомата f* называется полугруппа $\hat{f}^S = \Sigma A / \equiv_f$, где для $t_1, t_2 \in \Sigma A$ $t_1 \equiv_f t_2$ тогда и только тогда, когда для всех $\alpha, \beta \in (\Sigma A)^1$ $f(\alpha t_1 \beta) = f(\alpha t_2 \beta)$. Очевидно, что $\equiv_f$ является отношением конгруэнтности.

Если S — полугруппа, то *автоматом полугруппы* S называется функция S^f: $\Sigma S \to S$, для которой $S^f(s_1, \ldots, s_n) = s_1 \ldots s_n$. Полугруппа автомата полугруппы S записывается как S^{f^S} и $S^{f^S} \cong S$. Автомат полугруппы автомата f записывается как f^{Sf}. Существуют функции $h_1 : A \to f^S$ и $h_2 : f^S \to B$, такие, что $f = h_2 f^{Sf} h_1^\Gamma$, где отображение $h_1^\Gamma : \Sigma A \to \Sigma \hat{f}^S$ задается при помощи соотношения $h_1^\Gamma(a_1, \ldots, a_n) = [h(a_1), \ldots, h(a_n)]$.

Это равенство функций легко доказывается, если положить $h_1(a) = [a]_{\equiv_f}$ и $h_2([a_1, \ldots, a_n]_{\equiv_f}) = f(a_1, \ldots, a_n)$.

Пусть $\mathcal{F}$ будет некоторая совокупность автоматов. Определим совокупность $\mathcal{F}^S = \{f^S : f \in \mathcal{F}\}$. Аналогично пусть $\mathcal{S}$ — некоторая совокупность полугрупп. Определим $\mathcal{S}^f = \{S^f : S \in \mathcal{S}\}$.

2. Определение. Пусть S — полугруппа. *Эндоморфизмом* полугруппы S называется гомоморфизм $\varphi : S \to S$. Пусть $\mathrm{End}_L(S)$ обозначает множество эндоморфизмов полугруппы S. $\mathrm{End}_L(S)$ представляет собой подмножество и даже подполугруппу в $F_L(S)$.

Пусть задан некоторый гомоморфизм $\varphi : S_1 \to \mathrm{End}_L(S_2)$. Тогда *полупрямым произведением полугрупп* S_2 и S_1 с гомоморфизмом связи φ называется полугруппа $S_2 \times_\varphi S_1$, элементами которой служат элементы декартового произведения множеств $S_2 \times S_1$. Умножение определяется условием

$$(s_2, s_1) \cdot (s_2', s_1') = [s_2 \varphi(s_1)(s_2'), s_1 s_1'].$$

Читатель может без труда проверить свойство ассоциативности этого закона умножения. Если мы обозначим элемент $\varphi(s_1)(s_2')$ как

$^{s_1}(s_2')$, то

$$\left(s_1 s_1'\right)_{(s_2')} = {}^{s_1}\left[{}^{s_1'}(s_2')\right].$$

Так как отображение φ (S) — эндоморфизм полугруппы S_2 для всех элементов $s \in S_1$, то справедливо равенство

$^{s_1}(s_2 s_2') = {}^{s_1}(s_2)^{s_1}(s_2')$.

В новых обозначениях умножение определяется при помощи соотношения

$(s_2, s_1)\cdot(s_2', s_1') = [s_2{}^{s_1}(s_2'), s_1 s_1']$.

Следовательно, умножение координат из S_1 производится как обычно, в то время как умножение координат из S_2 выполняется при помощи «скручивания», зависящего от координаты из полугруппы S_1.

Прямое произведение представляет собой частный случай полупрямого произведения, так как $S_2 \times S_1 = S_2 \times_{\varphi} S_1$, где φ (s) есть тождественный эндоморфизм для всех элементов $s \in S_1$, т. е. $^{s_1}(s_2) = s_2$ для всех $s_1 \in S_1$ и $s_2 \in S_2$.

3. Определение. *Полугруппой правых отображений или правых преобразований,* называется пара *(X, S),* где *X* — непустое множество, а S — подполугруппа полугруппы $F_R(X)$. Для каждого $x \in X$ и каждого $s \in S$ положим *xs = (x) s.* Тогда выполняются следующие соотношения:

1) $x(s_1 s_2) = (x s_1) s_2$.

2) Если $s_1, s_2 \in S$ и $s_1 \neq s_2$, то $xs_1 \neq xs_2$ для некоторого $x \in X$.

Для каждой полугруппы *T* можно построить полугруппу правых отображений, это будет пара $[T^1, R(T)]$ (см. пункт 1.4и гл. 1), причем $R(T) \cong T$. Здесь для любых $t \in T^1$, $t' \in T$ имеет место соотношение

$tR(t') = tt'$.

Наоборот, если *X* — множество, S — полугруппа и задано отображение $\theta: X \times S \to X$, для которого записываем элемент 0 (x, s) как *xs,* то оно называется *правым действием* полугруппы S на множество *X* тогда и только тогда, когда выполняется условие 1. Действие называется *точным,* когда выполняется и условие 2. Если теперь тройка *(X, S,* 6) является действием, определим отображение $\hat{\theta}(s): X \to X$ при помощи соотношения $(x)\,\hat{\theta}(s) = xs$. Тогда пара $[X, \hat{\theta}(S)]$, называется *полугруппой правых отображений, ассоциированной с действием (X, S,* 6). Отображение $s \to \hat{\theta}(s)$ будет гомоморфизмом полугруппы S на образ $\hat{\theta}(S) \subseteq F_R(X)$ в соответствии с условием 1. Это отображение

будет взаимно однозначным тогда и только тогда, когда действие точное. Далее мы не будем различать точное действие и ассоциированную полугруппу правых отображений.

Две полугруппы правых отображений (X_1, S_1) и (X_2, S_2) называются *изоморфными* тогда и только тогда, когда существуют взаимно однозначное отображение $j : X_1 \to X_2$ (поэтому, в частности, $|X_1| = |X_2|$) и изоморфизм $\varphi : S_1 \to S_2$, такие, что $j(xs) = j(x)\,\varphi(s)$ для всех $x \in X_1$ и $s \in S_1$. Изоморфизм между ними обозначается как $(X_1, S_1) \cong (X_2, S_2)$. Очевидно, что отношение ^ представляет собой отношение эквивалентности на совокупности всех полугрупп правых отображений. Из условия $S_1 \cong S_2$, вообще говоря, не вытекает изоморфизм $(X_1, S_1) \cong (X_2, S_2)$.

Абстрактная полугруппа, определяемая парой (X, S), есть любая полугруппа *T,* изоморфная полугруппе S.

1.4. Определение. Пусть заданы полугруппы правых отображений (X_j, S_j), где / = 1, ..., *n.* Положим $X = X_n \times \ldots \times X_1$. Пусть 5 — подполугруппа в $F_R(X)$, состоящая из всех отображений $\psi : X \to X$, которые удовлетворяют следующим двум условиям:

1 (треугольное действие). Если $p_k : X \to X_k$"" "" обозначает отображение проектирования на *k-ю* компоненту, то для каждого $k = 1, ..., n$ существует отображение $f_k : X_k \times \ldots \times X_1 \to X_k$, такое, что $p_k \psi(t_n, \ldots, t_{k+1}, t_k, \ldots, t_1) = f_k(t_k, \ldots, t_1)$ для всех $t_i \in X_i$, $k = l, ..., n,$ т. е. новая *k-я* координата, получаемая в результате действия я|), зависит только от первых *k* координат элемента, на который действует отображение \p. Мы записываем это при помощи соотношения

$$\psi = w(f_n, \ldots, f_1).$$

2. *(k-я* компонента действия принадлежит полугруппе S_k)-Мы требуем, чтобы $f_1 \in S_1$ и для всех $k = 2, ..., n$ и всех $\alpha = (t_{k-1}, \ldots, t_1) \in X_{k-1} \times \ldots \times X_1$ отображение $g_\alpha \in F_R(X_k)$, определяемое с помощью соотношения $g_\alpha(y_k) = f_k(y_k, t_{k-1}, \ldots, t_1)$, принадлежало полугруппе S_h.

Тогда пара $(X, S) = (X_n, S_n) \wr \ldots \wr (X_1, S_1)$ называется *узловым*

произведением полугрупп правых отображений $(X_n, S_n), \ldots, (X_1, S_1)$ и

(X_n, S_n) w ... w (X_1, S_1) является абстрактной полугруппой, определяемой полугруппой правых отображений *(X, S).* Можно проверить, что

$$(X_3, S_3) \wr [(X_2, S_2) \wr (X_1, S_1)] \cong (X_3, S_3) \wr (X_2, S_2) \wr (X_1, S_1) \text{ и}$$

$$[(X_3, S_3) \wr (X_2, S_2)] \wr (X_1, S_1) \cong (X_3, S_3) \wr (X_2, S_2) \wr (X_1, S_1),$$

т. е. узловое произведение представляет собой ассоциативную операцию на полугруппах правых отображений.

Узловое произведение и полупрямое произведение тесно связаны друг с другом. Действительно, справедливо соотношение

$$(X_2, S_2) \wr (X_1, S_1) \cong [X_2 \times X_1, F(X_1, S_2) \times_Y S_1],$$

где $Y(s_1)(f)(x_1) = f(x_1 s_1)$ для $s_1 \in S_1$, $f : X_1 \to S_2$, $x_1 \in X_1$ и действие на декартово произведение $X_2 \times X_1$ определяется из соотношения $(x_2, x_1)(f, s) = [x_2 f(x_1), x_1 s]$. Тогда требуемый изоморфизм устанавливается при помощи функции, отображающей пару (x_2, Д) в пару (Д s), где $f_1(x) = xs$ и $f_2(x_2, x_1) = x_2 f(x_1)$. Следовательно, полугруппа изоморфна полупрямому произведению $F(X_1, S_2)$ на S_X. Отметим, что имеется изоморфизм $F(X_1, S_2) \cong S_2 \times \ldots \times S_2$ ($|X_1|$ раз).

2.5.2. Элементарные свойства автоматов

2.1. Определение. Пусть $f: \Sigma A \to B$ — автомат. Определим *естественное расширение* $f^\sigma : \Sigma A \to \Sigma B$ отображения / при помощи соотношения

$$f^\sigma(a_1, a_2, \ldots, a_n) = [f(a_1), f(a_1, a_2), \ldots, f(a_1, \ldots, a_n)].$$

Пусть задано отображение $h : A \to B$. Определим *единственное расширение* $h^\Gamma : \Sigma A \to \Sigma B$ отображения *h* при помощи соотношения $h^\Gamma(a_1, \ldots, a_n) = [h(a_1), \ldots, h(a_n)]$. Очевидно, что отображение h^r— гомоморфизм из полугруппы 2 *А* в полугруппу 26.

Говорят, что гомоморфизм $H : \Sigma A \to \Sigma B$ *сохраняет длину* тогда и только тогда, когда Я ($a_1, \ldots, a_n$) представляет собой последовательность длины *n* в 26 для всех последовательностей ($a_x \ldots .a_n$) длины

n в 2Л и всех целых положительных *n*. Заметим, что определенное нами естественное расширение h^T сохраняет длину. Кроме того, каждый сохраняющий длину гомоморфизм Я может быть однозначно представлен в виде h^T. Легко увидеть, что
отображение $h : A \to B$ задается равенством *h (a)* = Я (а) для всех элементов *а £ А*.

2.2. Система обозначений. Обозначения, введенные в определении 1.1, могут быть, очевидно, расширены. Так, например, $f^{S|S|S}$ будет автоматом полугруппы $f^{S|S}$; последний правый индекс указывает, является ли объект полугруппой или автоматом. Но в силу изоморфизма $S \cong S^{|S}$ длина последовательности верхних индексов не имеет существенного значения, справедливы, например, равенства $f^{S|S} = f^S$ и $f^{S|S|} = f^{S|}$ (после отождествления полугрупп $f^{S|S}$ и f^S).
По аналогии с этими обозначениями мы будем писать $f^{S|}$, имея в виду полугруппу $(f^S)^1$, и — $f^{S|}$, имея в виду $(f^S)^I$ (см. пункт 1.4 згл. 1). Соответственно $f^{S|I}$ будет употребляться вместо $(f^{S|})^I$ и т. д. Общее правило для определения природы объекта, записанного с помощью указанной системы обозначений, заключается в прочтении последовательности верхних индексов слева направо.

2.3. Замечание. Пусть $f : \Sigma A \to B$ — произвольный автомат. Напомним, что полугруппа $/^S$ определяется посредством отношения конгруэнтности на полугруппе 2/4, причем для элементов $\gamma, \delta \in \Sigma A$ $\gamma \equiv {}_f\delta$ тогда и только тогда, когда $f(\alpha\gamma\beta) = f(\alpha\delta\beta)$ для всех
$\alpha, \beta \in (\Sigma A)^1$.
Обозначим класс эквивалентности (относительно $\equiv {}_f$), содержащий элемент $\alpha \in \Sigma A$, через [а]$_y$. Пусть $h_f : A \to f^S$ — отображение, при котором $h_f(a) = [a]_f$. Тогда
отображение $f^S/h_f^1 : \Sigma A \to\!\!\to f^S$
будет каноническим гомоморфизмом, соответствующим конгруэнтности $\equiv {}_f$. Пусть отображение $j_f : f^S \to B$ определяется как $j_f [\alpha]_f = f(\alpha)$. Тогда / можно переписать в виде $f = j_f \cdot f^S/h_f^\Gamma$. Это равенство называется *фундаментальным разложением* автомата /.
В дальнейшем мы предполагаем, что все рассматриваемые автоматы обладают тем свойством, что их полугруппы имеют конечный порядок.

2.4. Определение, а) Пусть заданы автоматы $f : \Sigma A \to B$ и $g : \Sigma C \to D$. Говорят, что *f делит g* (это записывается как /| *g)* тогда и только тогда, когда существуют гомоморфизм $H : \Sigma A \to \Sigma C$ и отображение
$h : D \to B$, такие, что $f = hgH$. Говорят, что / *делит g (с сохранением*

длины) [это записывают как $f \mid g\ (lp)$] тогда и только тогда, когда Я — гомоморфизм, сохраняющий длину. Отметим, что отношения делимости автоматов и *(1р)* делимости автоматов являются рефлексивными и транзитивными отношениями.

б) Пусть заданы полугруппы *S* и *Т.* Говорят, что S делит *Т* (это записывается как *S* \ *T)* тогда и только тогда, когда 5 — гомоморфный'образ подполугруппы из Т [т. е. существуют подполугруппа $T' \subseteq \subseteq T$ и гомоморфизм $\varphi : T' \to S$, такой, что $\varphi\ (T') = S$].

Отметим, что отношение делимости полугрупп рефлексивное, антисимметричное и транзитивное (антисимметричность мы понимаем в том смысле, что из S | *Т* и *Т* | 5 вытекает, что $S \cong T$).

2.5. Замечание, а) Существуют такие автоматы /, *g,* что *f**g,* но / не делит *g* с сохранением длины. Пусть автомату определяется как Z|, ограниченный на 2 {1}. Тогда $\mathbf{Z}_2^l \mid g$, но уравнение $h_2 g h_1^{\Gamma} = \mathbf{Z}_2^l$ не имеет решения ни при каких h_x и h_2.

б) Пусть *Р* — разбиение полугруппы 5 и пусть *Q* — отношение конгруэнтности, порожденное *Р,* т. е. $s_1 \equiv s_2 \pmod{Q}$" тогда и
только тогда, когда $\alpha s_1 \beta \equiv \alpha s_2 \beta \pmod{P}$ для всех $\alpha, \beta \in S^1$. В этом случае при упорядочении отношений $(\mathrm{mod}\ Q) \subseteq (\mathrm{mod}\ P)$. Если $(\equiv)$— произвольная конгруэнтность на *S*, такая, что $(\equiv) \subseteq (\mathrm{mod}\ P)$, то легко доказать, что $(\equiv) \subseteq (\mathrm{mod}\ Q)$. В терминах гомоморфизмов мы можем переформулировать это свойство следующим образом. Назовем эпиморфизм $\varphi : S \twoheadrightarrow T$ *Р гомоморфизмом* и
запишем: $\varphi : S \twoheadrightarrow T$, если
из равенства $\varphi\ (s_1) = \varphi\ (s_2)$ вытекает, что $s_1 \equiv s_2\ (\mathrm{mod}^{P} P)$. Тогда существует *единственный минимальный Р'-гомоморфный образ* полугруппы 5, а именно $\eta : S \twoheadrightarrow S/Q$. Это означает, что
если $\varphi : S \twoheadrightarrow T$
есть *Р* гомоморфизм, то существует эпиморфизм $\psi : T \twoheadrightarrow^{P} S/Q$, такой, что $\psi\, \varphi = \eta : S \twoheadrightarrow S/Q$ (см. параграф 1 гл. 8).

Теперь очевидно, что если $f : \Sigma A \to B$ — некоторый автомат, то $f^S f h_f^{\Gamma} : \Sigma A \twoheadrightarrow f^S$ будет единственным минимальным (mod f) гомоморфным образом полугруппы 2Л, здесь $\alpha \pmod{f} \beta$ тогда и только тогда, когда $f\ (\alpha) = f\ (\beta)$.

Пусть $\theta : \Sigma A \to S$ будет некоторым (mod /) гомоморфизмом *ЪА.* Тогда существует эпиморфизм $\varphi : \theta\ (\Sigma A) \twoheadrightarrow f^S$. Следовательно, $f^S \xleftarrow{\varphi} \theta\ (\Sigma A) \subseteq S$ и поэтому $f^S \mid S$. Эти замечания понадобятся
для доказательства следующего предложения.

2.6. Предложение, а) Пусть / — автомат. Тогда $f \mid f^{S/}$ (lp).
б) Пусть S *и* T — полугруппы. Тогда из свойства $S \backslash T$ вытекает, что $S^f \mid T^f$ (lp).
в) Пусть *fug*— автоматы. Тогда из свойства $f \backslash g$ вытекает, что $f^S \mid g^S$.

Доказательство, а) Справедливость этой части предложения вытекает из существования фундаментального разложения для /.
б)Предположим, что 5 |- *Т.* Пусть 7" — подполугруппа в Г, а $\varphi : T' \to S$ — гомоморфизм, такой, что $\varphi\ (T') = S$. Для каждого элемента $s \pounds S$ выберем представителя $\bar{s}$ в множестве $\varphi^{-1}\ (s) \subseteq T'$. Определим отображение $h_1 : S \to T$ как $h_1\ (s) = \bar{s}$. Определим отображение $h_2 : T \to S$ как $h_2\ (t) = \varphi\ (t)$ для t g 7" и произвольным образом для $t \notin T'$.
Тогда $S^f = h_2 T^f h_1^{\Gamma}$, следовательно, $S^f \mid T^f (lp)$.
в)Предположим, что $f \backslash g$. Тогда в силу пункта а)и $f \mid g^{S/}$ поэтому $f = hg^{S/} H$, где H — гомоморфизм. Следовательно, отображение $g^{S/} H : \Sigma A \to g^S$ является (mod /) гомоморфизмом и в силу замечания 2.5 мы получаем, что $f^S \mid g^S$.Предложение полностью доказано.
2.7. Замечание. Пусть / — автомат, предположим, что S — полугруппа, удовлетворяющая равенству $f = hS^f H$ при соответствующих отображении *h* и гомоморфизме *Н.* Мы знаем, что по крайней мере одно решение этого уравнения существует — S = /^{s}. В общем случае имеем $f \mid S^f$ и поэтому $f^S \mid S^{fS} = S$. Из этого следует, что полугруппа $S = f^s$ — единственное (с точностью до изоморфизма) и минимальное решение этого уравнения и в случае, когда $S — f^s$, само соотношение становится фундаментальным разложением.
Мы видели, что для произвольного заданного автомата $f : \Sigma A \to B$ существует каноническая полугруппа f^s, ассоциированная с автоматом *f,P* — единственный минимальный гомоморфный образ полугруппы2Л относительно (mod/) гомоморфизма. Естественно, возникает следующий вопрос. Предположим мы построили какой-то новый автомат с помощью заданных автоматов *fug.* Как в этом случае полугруппа нового автомата связана с полугруппами f^S и g^{S1}?
Если мы возьмем автомат $g : \Sigma C \to D$ и кодируем его входные и выходные сигналы (т. е. определим гомоморфизм *Н* из свободной полугруппы 2Л, порожденной множеством *А* в полугруппу 2С и отображение $h : D \to B$ в другое множество), то получим автомат $f : \Sigma A \to$
$\to B$, гдеМы видим, что в этом случае $f^S \mid g^S$, так как $f \backslash g$.

$f = hgH.$

Существуют два очевидных способа композиции автоматов — это последовательное и параллельное соединения. (Напомним, определение"2.Гдля/<[1].)

2.8. Определение. Пусть $f : \Sigma A \to B$ и $g : \Sigma C \to D$ — автоматы.

а) *Последовательным соединением* (или *последовательной композицией)* автоматов / и *g* с *гомоморфизмом связи* $H: \Sigma B \to \Sigma C$ называется автомат $gHf^{\sigma} : \Sigma A \to D.$

б) *Параллельным соединением* (или *параллельной композицией)* автоматов *fug* называется автомат $f \times g : \Sigma (A \times C) \to B \times D,$ определяемый соотношением

$$(f \times g)[(a_1, c_1),..., (a_n, c_n)] = [f (a_1,..., a_n), \ g (c_1, ..., c_n)].$$

Последовательное и параллельное соединения любого конечного числа автоматов определяются аналогично.

2.9. Утверждение. Пусть $f_1, \ ..., \ f_n$ — автоматы, $f_i : \Sigma A_i \to B_i, \ i = 1, ..., n.$ Тогда $(f_n \times ... \times f_1)^S | f_n^S \times ... \times f_1^S.$ В общем случае $(f_n \times ... \times f_1)^S \not\cong f_n^S \times ... \times f_1^S.$

Доказательство. Используя фундаментальное разложение для каждого автомата, получаем соотношение

$$f_n \times ... \times f_1 = (j_{f_n} \times ... \times j_{f_1}) \cdot (f_n^{S_f} h_{f_n}^{\Gamma} \times ... \times f_1^{S_f} h_{f_1}^{\Gamma}),$$

в котором произведение отображений определяется очевидным образом. Отображение

$$\theta = f_n^{S_f} h_{f_n}^{\Gamma} \times ... \times f_1^{S_f} h_{f_1}^{\Gamma} : \Sigma (A_n \times ... \times A_1) \to f_n^S \times ... \times f_1^S$$

является $(\mathrm{mod}\ f_n \times .. \times f_1)$ гомоморфизмом. Из этого в соответствии с пунктом б) замечения 2.5 вытекает первая часть утверждения.

Приведем теперь пример, доказывающий неизоморфность $(f_n \times \times ... \times f_1)^S$ и $f_n^S \times ... \times f_1^S.$ Рассмотрим автомат $\mathbf{Z}_2^{f} \times \mathbf{Z}_2^{f},$ ограниченный на $\Sigma (\{1\} \times \{1\}).$

Полугруппа автомата $\mathbf{Z}_2^{f},$ ограниченного на 2 {1}, есть Z_2, полугруппа автомата $\mathbf{Z}_2^{f} \times \mathbf{Z}_2^{f},$ ограниченного на $\Sigma(\{1\} \times \{1\}),$ также есть Z_2 и она не совпадает с полугруппой Z_2 x Z_2.

Изучение последовательной композиции не является столь же легким делом. Напомним определение *узлового произведения* полугрупп правых отображений и связь его с полупрямым произведением,

(X_2, S_2) w $(X_1, S_1) \cong F(X_1, S_2) \times {}_Y S_1$,

где гомоморфизм связи *Y* указан вслед за определением 1.4.

Рассмотрим полугруппу отображений, которая играет исключительно важную роль в дальнейшем изложении. Предположим, что 5 — полугруппа, пусть *R (S)* обозначает правое регулярное представление полугруппы S (см. пункт 1.4и гл. 1). 'Тогда пара $[S^1, R(S)]$ — полугруппа правых отображений; так как имеется изоморфизм $S \cong R(S)$, мы будем для краткости обозначать введенную полугруппу правых отображений как $S \cong R(S)$.

2.10. Определение. Пусть заданы две полугруппы правых отображений *(X, S)* и *(Y, T).* Мы будем говорить, что *(X, S) делит (Y, T)* [это обозначается как $(X, S) \mid (Y, T)$] в том и только в том случае, когда выполняются следующие условия.

1. Существуют подмножество *Y'* в *Y* и подполугруппа *V* в *T,* такие, что *Y'* инвариантно относительно действия 7'", т. е. $Y'T' \subseteq Y'$.

2.Существуют отображение и $\theta : Y' \to\!\!\!\!\to X$ эпиморфизм $\varphi : T' \to\!\!\!\!\to S$, такие, что $\theta(yt) = \theta(y)\,\varphi(t)$ для всех элементов $y \in$

$\in Y'$ и $t \in T'$.

Отметим, что мы не требуем, чтобы полугруппа 7" действовала точно на *Y'*. Если условия 1 и 2 справедливы, будем писать $(Y', T') \subseteq$

$\subseteq (Y, T)$ и $(Y', T') \xrightarrow{(\theta, \varphi)} (X, S)$.

Введем теперь понятие прямого произведения для полугрупп отображений и докажем некоторые соотношения, связывающие узловые произведения, прямые произведения и понятие делимости полугрупп отображений.

2.11. Определение. Назовем *прямым произведением (X, S)* X X *(Y, T)* полугрупп отображений *(X, S)* и *(Y, T)* полугруппу отображений $(X \times Y, S \times T)$,~ для которой $(x, y)(s, t) = (xs, yt)$. Прямое произведение любого конечного числа полугрупп отображений определяется аналогично.

2.12. Замечание. Легко устанавливается справедливость следующих фактов:

а) свойство делимости полугрупп отображений рефлексивно, антисимметрично и транзитивно;

б) если $(X, S) \mid (Y, T)$,...,.., то S | *T;*

в)51 Г тогда и только тогда, когда $(S^1, S) \mid (T^1, T)$;

г) $(X_i, S_i) \mid (X_2, S_2) \times (X_1, S_1)$, $i = 1, 2$;_ .....

д) пусть $(X_i, S_i) \mid (Y_i, T_i)$, $i = 1, 2$, тогда

$(X_2, S_2) \times (X_1, S_1) \mid (Y_2, T_2) \times (Y_1, T_1)$.

2.13. Замечание. Предположим, что $(X, S) \mid (Y, T)$^.....и пара (V, *T')*

удовлетворяет условиям определения 2.10. Введем отношение эквивалентности $\equiv$ на полугруппе 7", полагая $t_1 \equiv t_2$ тогда и только тогда, когда для всех элементов $y \in Y'$ выполняется равенство $yt_1 = yt_2$. Легко проверить, что отношение $\equiv$ будет конгруэнтностью на полугруппе 7" и фактор-полугруппа $T'/\equiv$ представляет собой единственный максимальный гомоморфный образ $\overline{T}$ полугруппы 7", для которого пара $(Y', \overline{T})$ является полугруппой отображений, т. е. $\overline{T}$ действует точно на Y'. Легко видеть тогда, что $(Y', T'/\equiv) \to\to (X, S)$.

2.14. Утверждение,

а) $(X_2, S_2) \times (X_1, S_1) \mid (X_2, S_2) \wr (X_1, S_1)$.

б) $(X_i, S_i) \mid (X_2, S_2) \wr (X_1, S_1), i = 1,2;$

в) пусть X — произвольное множество, такое, что пара (X, S) — полугруппа отображений.

Тогда $(S^1, S) \mid (X, S) \times \ldots \times (X, S)$ ($|X|$ раз);

г) пусть $(X_i, S_i) \mid (Y_i, T_i)$, $i = 1,2$. Тогда

$(X_2, S_2) \wr (X_1, S_1) \mid (Y_2, T_2) \wr (Y_1, T_1)$.

Доказательство, а) Необходимо только доказать, что

$$S_2 \times S_1 \subseteq F(X_1, S_2) \times {}_Y S_1,$$

но это очевидно в силу мономорфизма $\varphi(s_2, s_1) = (f_{s_2}, s_1)$, где $f_{s_2}(x) = s_2$ для всех элементов $x \in X_x$.

б) Это соотношение следует из соотношения а) и пункта г) замечания 2.12 в силу транзитивности свойства делимости полугрупп отображений.

в) Так как пара (X, S) представляет собой полугруппу отображений, 5 есть подполугруппа полугруппы $F_R(X)$. Кроме того, S^1 есть подполугруппа в $F_R(X)$, так как, если $1 \notin S$, мы можем присоединить к S тождественное отображение из $F_R(X)$'. Поскольку множество X конечно, перенумеруем его элементы числами 1, 2.....n. Тогда каждый элемент $f \in F_R(X)$ можно переписать как $(i_1, \ldots, i_n)$, где $(k)\, f = i_k$. Если $g = (j_1, \ldots, j_n)$, то по определению $(i_1, \ldots, i_n) * (j_1, \ldots, j_n) = (j_{i_1}, \ldots, j_{i_n})$. Поэтому $F_R(X)$ можно отождествить с полугруппой $(X \times \ldots \times X, *)$.

Пусть $\hat{S}$ — образ полугруппы S^1 в полугруппе $X \times \ldots \times X$. Тогда

$$\left[\hat{S}, (s, \ldots, s) : s \in S\right] \subseteq (X \times \ldots \times X, S \times \ldots \times S).$$

Кроме того, легко проверить, что

$$\left[\hat{S},\ \{(s,\ldots,s): s \in S\}\right] \cong (S^1, S).$$

Следовательно,

$$(S^1, S) \mid (X \times \ldots \times X,\ S \times \ldots \times S) = (X, S) \times \ldots \times (X, S).$$

г) Доказательство этого пункта разобьем на четыре части.

1. Пусть $(Z_1, W_1) \subseteq (Y_1, T_1)$ и пусть пара (X, S) — произвольная полугруппа отображений. Тогда

$$(X, S) \wr (Z_1, W_1/\equiv) \mid (X, S) \wr (Y_1, T_1).$$

Доказательство. Множество $V = F(Y_1, S) \times_Y W_1$ является подполугруппой в $F(Y_1, S) \times_Y T_1$ и поэтому

$$(X \times Z_1, V) \subseteq \left[X \times Y_1, F(Y_1, S) \times_Y T_1\right].$$

Определим эпиморфизм $\theta: V \twoheadrightarrow F(Z_1, S) \times_Y W_1/\equiv$ при помощи соотношения $\theta(\hat{f}, w) = (\hat{f}, [w])$, где $\hat{f} = f$. . ограниченное на Z_x и $[w]$ есть образ элемента w относительно эпиморфизма, ассоциированного с конгруэнтностью $\equiv$ (см. замечание 2.13). Следовательно,

$$[X \times Z_1, F(Z_1 S) \times_Y W_1/\equiv] \leftarrow\!\!\!\leftarrow (X \times Z_1, V).$$

Тем самым первая часть доказана.

2. Пусть имеет место включение $(Z_2, W_2) \subseteq (Y_2, T_2)$. Тогда

$$(Z_2, W_2/\equiv) \wr (X, S) \mid (Y_2, T_2) \wr (X, S).$$

Доказательство.

$$[Z_2 \times X, F(X, W_2) \times_Y S] \subseteq [Y_2 \times X, F(X, T_2) \times_Y S].$$

Определим отображение $\theta: F(X, W_2) \times_Y S \twoheadrightarrow F(X, W_2/\equiv) \times_Y S$, полагая $\theta(f, s) = \{[f(\cdot)], s\}$.

Очевидно, 0 есть эпиморфизм. Таким образом, доказана вторая часть.

3. Предположим, что (Y_1, T_1) — полугруппа отображений и $(X_1, S_1) \leftarrow\!\!\!\leftarrow (Y_1, T_1)$. Тогда справедливо соотношение

$$(X, S) \wr (X_1, S_1) \mid (X, S) \wr (Y_1, T_1).$$

Доказательство. Пусть $\theta: Y_1 \twoheadrightarrow X_1$ и $\varphi: T_1 \twoheadrightarrow S_1$ — отображения, соответствующие $(X_1, S_1) \leftarrow\!\!\!\leftarrow (Y_1, T_1)$. Определим подполугруппу

$$V = \{(f, t) \in F(Y_1, S) \times_Y T_1 : f(y_1) = f(y_2)\}$$

вся" кий раз, как только $\theta(y_1) = \theta(y_2)$.

Тогда

$$(X \times Y_1, V) \subseteq [X \times Y_1, F(Y_1, S) \times_Y T_1].$$

Определим отображение $\psi: V \twoheadrightarrow F(X_1, S) \times_Y S_1$, положив

$\psi(\bar{f}, t) = [\hat{f}, \varphi(t)]$,
где $\hat{f}(x) = f(\bar{x})$, $\bar{x} \in \theta^{-1}(x)$. Отображение $\hat{f}$ полностью определено в силу определения полугруппы *V*. Отображение яр будет эпиморфизмом и легко проверить, что

$$[X \times X_1, F(X_1, S) \times_Y S_1] \leftarrow\!\!\leftarrow (X \times Y_1, V),$$

где $\hat{\theta} : X \times Y_1 \twoheadrightarrow X \times X_1$ задается при помощи соотношения $\hat{\theta}(x, y_1) = [x, \theta(y_1)]$. Тем самым доказана третья часть.

4. Пусть пара $(У_2, T\%)$ — полугруппа отображений и предположим, что $(X_2, S_2) \leftarrow\!\!\leftarrow (Y_2, T_2)$.

Тогда $(X_2, S_2) \wr (X, S) \mid (Y_2, T_2) \wr (X, S)$.

Доказательство. Пусть отображения Э и ф такие же, как в пункте 3. Определим отображение $\hat{\theta} : Y_2 \times X \twoheadrightarrow X_2 \times X$ при помощи соотношения $\hat{\theta}(y_2, x) = [\theta(y_2), x]$. Определим отображение

$\psi : F(X, T_2) \times$
$\times_Y S \twoheadrightarrow F(X, S_2) \times_Y S$, полагая $\psi(f, s) = (\varphi \cdot f, s)$.

Отображение г|? есть эпиморфизм и легко проверить, что

$$[X_2 \times X, F(X, S_2) \times_Y S] \leftarrow\!\!\leftarrow [Y_2 \times X, F(X, T_2) \times_Y S].$$

Мы доказали четвертую часть.

Теперь в силу 1—4, замечания 2.13 и транзитивности свойства делимости имеем $(X_2, S_2) \wr (X_1, S_1) \mid (Y_2, T_2) \wr (Y_1, T_1)$ всякий раз, как только $(X_i, S_i) \mid (Y_i, T_i)$, $i = 1,2$. Тем самым утверждение 2.14 полностью доказано.

Перейдем теперь к случаю последовательного соединения. Пусть задан автомат $f : \Sigma A \to B$. Тогда для всех элементов $\alpha, \beta \in (\Sigma A)^1$ определим правую конгруэнтность $\equiv_{Q_f}$, полагая $\alpha \equiv_{Q_f} \beta$" тогда и только тогда, когда $f(\alpha\gamma) = f(\beta\gamma)$ для всех $\gamma \in (\Sigma A)^1$; причем с самого начала считается, что $f(1) \notin B$. Пусть Q_f обозначает множество классов эквивалентности отношения $\equiv_{Q_f}$. Через $[\alpha]_{Q_f}$ обозначим класс, содержащий элемент $\alpha \in (\Sigma A)^1$.

Как и раньше, теперь $[\alpha]_f$ обозначает элемент полугруппы f^S, в который элемент $\alpha \in \Sigma A$ переходит при отображении $f^S i h_f$ (см. замечание 2.3).

Тогда пара (Q_f, f^S) будет полугруппой правых отображений относительно действия $[\alpha]_{Q_f} \cdot [\beta]_f = [\alpha\beta]_{Q_f} \in Q_f$. Нетрудно проверить, что это действие определено корректно и является точным.

Следующий результат представляет собой важное звено в дальнейших исследованиях.

2.15. Предложение. Пусть $F = f_n H_{n-1} f^{\sigma}_{n-1} \ldots f^{\sigma}_2 H_1 f^{\sigma}_1$, $n \geqslant 2$, где отображения f_i определяют автоматы, H_i есть гомоморфизм связи. Тогда

а) $(Q_F, F^S) \mid (Q_{f_n}, f^S_n) \wr \ldots \wr (Q_{f_1}, f^S_1)$,

б) $F^S \mid (f^{S1}_n, f^S_n) \mathrm{w} \ldots \mathrm{w} (f^{S1}_1, f^S_1)$.

Доказательство. Пункт а) необходимо доказать только для $n = 2$, так как общий случай тогда сразу же доказывается по индукции при помощи транзитивности свойства делимости полугрупп отображений и пункта г) утверждения 2.14. Пусть $F = gHf^{\sigma} : \Sigma A \to D$, где $f : \Sigma A \to B$ и $g : \Sigma C \to D$ — автоматы, а $H : \Sigma B \to \Sigma C$ — гомоморфизм связи. Мы должны показать, что

$(Q_F, F^S) \mid (Q_g, g^S) \wr (Q_f, f^S)$.

Для каждого элемента $a \in A$ определим элемент $\hat{a} \in (Q_g, g^S) \mathrm{w} (Q_f, f^S)$, полагая $\hat{a} = ([\beta]_{Q_g}, [\alpha]_{Q_f}) = ([\beta]_{Q_g} \cdot [Hf(\alpha \cdot a)]_g, [\alpha]_{Q_f} \cdot [a]_f)$, где $\alpha \in (\Sigma A)^1$, $\beta \in (\Sigma C)^1$. Тогда $\alpha \cdot a \in \Sigma A$ и $Hf(\alpha \cdot a) \in \Sigma C$.

Пусть символ $\hat{A}$ обозначает подполугруппу узлового произведения $(Q_g, g^S) \mathrm{w} (Q_f, f^S)$, порожденную множеством элементов $\{\hat{a} : a \in A\}$, и пусть символ φ обозначает единственное расширение отображения $a \to \hat{a}$ до эпиморфизма полугруппы ΣA в полугруппу $\hat{A}$. Мы покажем, что φ является (mod F) гомоморфизмом и поэтому он устанавливает отображение

$$F^S \xleftarrow{\psi} A \subseteq (Q_g, g^S) \mathrm{w} (Q_f, f^S)$$

согласно пункту б) замечания 2.5, ψ представляет собой эпиморфизм и $\psi(a_1 \ldots a_n) = [a_1, \ldots, a_n]_F$.

Пусть $\alpha = (a_1, \ldots, a_n) \in \Sigma A$. Тогда $\varphi(\alpha) = \hat{a}_1 \ldots \hat{a}_n$. Пусть $\varphi(\alpha)$ действует на $([1]_{Q_g}, [1]_{Q_f})$.

$$([1]_{Q_g}, [1]_{Q_f})\,\varphi(\alpha) = ([1]_{Q_g}, [1]_{Q_f})\,\hat{a}_1 \ldots \hat{a}_n = ([1]_{Q_g} \cdot [Hf(a_1)]_g \cdot [Hf(a_1, a_2)]_g$$

$$\ldots [Hf(a_1, \ldots, a_n)]_g, [1]_{Q_f} \cdot [(a_1, \ldots, a_n)]_f) = ([Hf(a_1) \cdot Hf(a_1, a_2) \ldots Hf(a_1,$$

$$\ldots, a_n)]_{Q_g}, [(a_1, \ldots, a_n)]_{Q_f}) = ([Hf^{\sigma}(\alpha)]_{Q_g}, [\alpha]_{Q_f}). \quad (2.1)$$

Теперь предположим, что $\varphi(\alpha) = \varphi(\beta)$, $\alpha, \beta \in \Sigma A$. Тогда в силу соотношения (2.1)

$$[Hf^{\sigma}(\alpha)]_{Q_g} = [Hf^{\sigma}(\beta)]_{Q_g},$$

откуда вытекает, что $g(Hf^{\sigma}(\alpha)) = g(Hf^{\sigma}(\beta))$. Так как $F = gHf^{\sigma}$, получаем, что $F(\alpha) = F(\beta)$, и, следовательно, отображение φ является (mod *F)* гомоморфизмом.

Для того чтобы установить делимость полугрупп отображений, определим подмножество *X* в декартовом произведении, полагая, что $X = 1 \cdot \hat{A} \cup \{1\}$, где $1 = ([1]_{Q_g}, [1]_{Q_f}) \in Q_g \times Q_f$. Тогда полугруппа $\hat{A}$ оставляет множество *X* инвариантным. Предположим, что $1 \cdot (\hat{a}_1 \ldots \hat{a}_n) = 1 \cdot (\hat{b}_1 \ldots \hat{b}_m)$. Тогда в силу уравнения (2.1) получаем

$$[(a_1, \ldots, a_n)]_{Q_f} = [(b_1, \ldots, b_m)]_{Q_f},$$

откуда вытекает

$$[(a_1, \ldots, a_n)]_{Q_F} = [(b_1, \ldots, b_m)]_{Q_F}.$$

Следовательно, можно определить отображение $\theta : X \twoheadrightarrow Q_F$ при помощи соотношений

$$\theta(x) = \begin{cases} [(a_1, \ldots, a_n)]_{Q_F}, & \text{если } x = 1 \cdot (\hat{a}_1 \ldots \hat{a}_n), \\ [1]_{Q_F}, & \text{если } x = 1. \end{cases}$$

Теперь получаем

$$(Q_F, F^S) \xleftarrow{(\theta, \psi)} (X, \hat{A}) \subseteq (Q_g, g^S) \wr (Q_f, f^S),$$

где θ и ψ удовлетворяют определению делимости полугрупп отображений. Тем самым пункт а) доказан.

б) Этот пункт вытекает из пункта а) и из того факта, что $(Q_f, f^S) | (f^{S1}, f^S)$ для любых автоматов *f*. В самом деле, $(Q_f, f^S) \xleftarrow{(\theta, \varphi)} (f^{S1}, f^S)$, где φ — тождественный изоморфизм, а отображение $\theta : f^{S1} \twoheadrightarrow Q_f$ задается при помощи соотношения $\theta([\alpha]_f) = [\alpha]_{Q_f}$ для $\alpha \in \Sigma \hat{A}$ и $\theta(1) = [1]_{Q_f}$. Воспользовавшись упорядоченностью отношений на полугруппах $(\equiv_f) \subseteq (\equiv_{Q_f})$, легко проверить, что отображение θ определено корректно. Следовательно, $(Q_f, f^S) | (f^{S1}, f^S)$. Тогда в силу свойства транзитивности делимости полугрупп отображений и пункта г) утверждения 2.14 получаем требуемый результат.

Теперь будем строить композиции автоматов при помощи комбинирования последовательных и параллельных соединений, исключая при этом структуры с петлями. Пусть $\mathcal{F}$ — совокупность автоматов. Определим семейство автоматов, содержащее $\mathcal{F}$ и замкнутое относительно операций последовательного и параллельного соединений и делимости, требуя, однако, для всех включаемых в эти понятия гомоморфизмов выполнения свойства сохранения длины. Это требование желательно с практической точки зрения, так как свойство

сохранения длины означает наличие реальной связи (например, при помощи проводов) между автоматами. Указанное семейство определяется так, чтобы в доказательствах было удобно пользоваться методом математической индукции.

2.16. Определение. Пусть $\mathcal{F}$ — некоторое множество автоматов. Определим *последовательно-параллельное* замыкание $SP(\mathcal{F})$ множества f как

$SP(\mathcal{F}) = \bigcup \{SP_i(\mathcal{F}): i = 1, 2\ldots\}$, где $SP_1(\mathcal{F}) = \mathcal{F}$ и $SP_i(\mathcal{F}) =$ $= \{f_2 \times f_1,\ f_2 h^{\Gamma} f_1^{\sigma},\ h_2 f h_1^{\Gamma} : f_1,\ f_2 \in SP_{i-1}(\mathcal{F})$ и $h,\ h_1,\ h_2$ отображения$\}$.

Что можно сказать о совокупности полугрупп автоматов из $SP(\mathcal{F})$, т. е. о множестве $SP(\mathcal{F})^S$? Из полученных ранее результатов следует, что $SP(\mathcal{F})^S$ должно быть замкнуто относительно операций прямого и узлового произведений, а также относительно некоторых случаев делимости. Это заключение приводит к следующим определениям.

2.17. Определение. Пусть $\mathcal{S}$ — некоторая совокупность полугрупп. Определим семейство полугрупп отображений $\overline{W}(\mathcal{S})$, называемое *замыканием $\mathcal{S}$ относительно узловых произведений и делимости,* полагая $\overline{W}(\mathcal{S}) = \bigcup \{\overline{W}_i(\mathcal{S}) : i = 1,2, \ldots\}$, где

$\overline{W}_1(\mathcal{S}) = \{(S^1,\ S) : S \in \mathcal{S}\}$ и

$\overline{W}_i(\mathcal{S}) = \{(X,\ S)$: $(X,\ S) | (X',\ S')$ для некоторой пары $(X',$ $S') \in \{\overline{W}_{i-1}(\mathcal{S}) \cup \{[(X_2,\ S_2) \wr (X_1,\ S_1) : (X_j,\ S_j) \in \overline{W}_{i-1}(\mathcal{S})]\}\}$.

Обозначим через $\overline{W}(\mathcal{S})$ семейство абстрактных полугрупп, ассоциированных с полугруппами отображений из $\overline{W}(\mathcal{S})$.

2.18. Утверждение. Пусть $\mathcal{S}$ — семейство полугрупп. Если $(X,\ S) \in \overline{W}(\mathcal{S})$, то $(X,\ S) | (S_n^1,\ S_n) \wr \ldots \wr (S_1^1,\ S_1)$ для некоторых $S_i \in \mathcal{S}$, $i = 1, \ldots, n$ и $1 \leqslant n$. В частности, если $S \in W(\mathcal{S})$, то $S | (S_n^1,\ S_n)$ w $\ldots$ w$(S_1^1,\ S_1)$, каждая $S_i \in \mathcal{S}$ или $S | T \in \mathcal{S}$.

Доказательство. Очевидно, вторая часть утверждения следует из первой. Мы докажем первое соотношение с помощью индукции по i для $\overline{W}_i(\mathcal{S})$. Для $i = 1$ утверждение, как легко видеть, справедливо. Допустим, что оно верно для $i = n$, и предположим, что $(X,\ S) \in$ $\in \overline{W}_{n+1}(\mathcal{S})$. Тогда выполняется одно из соотношений:

1. $(X,\ S) | (Y,\ T) \in \overline{W}_n(\mathcal{S})$

ИЛИ

2. $(X,\ S) | (Y_2,\ T_2) \wr (Y_1,\ T_1)$, $(Y_i,\ T_i) \in \overline{W}_n(\mathcal{S})$, $i = 1, 2$.

В первом случае утверждение оказывается верным в силу предположения индукции и транзитивности свойства делимости. Во втором случае получаем требуемое в силу пункта г) утверждения 2.14, а также предположения индукции и транзитивности.

2.19. Замечание. Для удобства обозначений мы введем укороченный символ $S_2 \mathrm{w} S_1$ для полугруппы $(S_2^1,\ S_2)\mathrm{w}\ (S_1^1,\ S_1)$. Однако это новое обозначение должно использоваться с некоторой осторожностью, так как операция «w» на *полугруппах* не является *ассоциативной,* в то время как операции « ≀ » и «w» на полугруппах отображений ассоциативны.

Эта трудность возникает из-за того, что в определении операции «w» моноиды S_2^1 и S_1^1 рассматриваются как множества, на которые действуют соответственно полугруппы S_2 и S_1. Поэтому

$$(S_3 \mathrm{w}\, S_2) \mathrm{w}\, S_1 = F\,[S_1^1,\ F(S_2^1,\ S_3) \times_Y S_2] \times_Y S_1,$$

в то время как

$$S_3 \mathrm{w}\,(S_2 \mathrm{w}\, S_1) = F\,([F(S_1^1,\ S_2) \times_Y S_1]^1, S_3) \times_Y [F(S_1^1,\ S_2) \times_Y S_1].$$

И поскольку в общем случае мощности появляющихся здесь множителей не равны, операция «w» не будет ассоциативной.

Однако полугруппа $(X_2,\ S_2)\ \mathrm{w}\ (X_1,\ S_1)$ зависит от $S_1,\ S_2$ и X_1, но не зависит от X_2. Учитывая это соображение, легко показать, что

$$(S_3 \mathrm{w}\, S_2) \mathrm{w}\, S_1 = (S_3^1,\ S_3) \mathrm{w}\, (S_2^1,\ S_2) \mathrm{w}\, (S_1^1,\ S_1).$$

Последнее соотношение как раз то, которое нам требовалось. Оно приводит к следующему определению.

2.20. Определение. Пусть имеются полугруппы $S_1,\ \ldots,\ S_n$. Определим произведение $S_1 \mathrm{w} \ldots \mathrm{w}\, S_n = R_n$ по индукции, полагая $R_1 = S_1,\ R_n = R_{n-1} \mathrm{w}\, S_n$. (Для удобства мы изменили порядок индексов.) Тогда

$$S_n \mathrm{w} \ldots \mathrm{w}\, S_1 = (S_n^1,\ S_n) \mathrm{w} \ldots \mathrm{w}\, (S_1^1,\ S_1).$$

2.21. Утверждение. а) Если $T \in W(\mathscr{S})$, то $(T^1,\ T) \in \overline{W}(\mathscr{S})$.

б) Если $S \mid T$ и $T \in W(\mathscr{S})$, то $S \in W(\mathscr{S})$.

в) Если $S_1,\ S_2 \in W(\mathscr{S})$, то $S_2 \mathrm{w}\, S_1 \in W(\mathscr{S})$.

Доказательство, а) Условие $T \in W(\mathscr{S})$ означает, что существует множество X, такое, что $(X,\ T) \in \overline{W}(\mathscr{S})$. В силу пункта в) утверждения 2.14 $(T^1,\ T) \mid (X,\ T) \times \ldots \times (X,\ T)$ (произведение берется $|X|$ раз). Но в силу пункта а) семейство $\overline{W}(\mathscr{S})$ замкнуто относительно операции прямого произведения и поэтому $(T^1,\ T) \in \overline{W}(\mathscr{S})$.

б) Из условия $S \mid T$ вытекает, что $(S^1,\ S) \mid (T^1,\ T)$, но тогда в соответствии с пунктом а) получаем $(S^1,\ S) \in \overline{W}(\mathscr{S})$ и $S \in W(\mathscr{S})$.

в) Справедливость этой части утверждения немедленно вытекает из пункта а).

Приведем теперь другое определение семейства $W(\mathcal{S})$, более удобное для последующей работы.

2.22. Определение. Пусть $\mathcal{S}$ — семейство полугрупп. Определим семейство $\hat{W}(\mathcal{S})$ полугрупп, полагая $\hat{W}(\mathcal{S}) = \bigcup \{\hat{W}_i(\mathcal{S}) : i = 1, 2, \ldots\}$, где $\hat{W}_1(\mathcal{S}) = \mathcal{S}$ и $\hat{W}_i(\mathcal{S}) = \{S : S \mid T$ для некоторой полугруппы $T \in \hat{W}_{i-1}(\mathcal{S}) \cup [S_2 \text{ w } S_1 : S_1,\ S_2 \in \hat{W}_{i-1}(\mathcal{S})]\}$

2.23. Утверждение. $\hat{W}(\mathcal{S}) = W(\mathcal{S})$.

Доказательство. Очевидно, в силу утверждения 2.18 справедливо включение $W(\mathcal{S}) \subseteq \hat{W}(\mathcal{S})$. Для того чтобы доказать обратное включение, предположим, что $\hat{W}_n(\mathcal{S}) \subseteq W(\mathcal{S})$, пусть, кроме того, $S \in \hat{W}_{n+1}(\mathcal{S})$. Тогда или 1) $S \mid T \in \hat{W}_n(\mathcal{S})$, или 2) $S \mid T_2$ w T_1, где $T_1, T_2 \in \hat{W}_n(\mathcal{S})$. В первом случае $T \in W(\mathcal{S})$ по предположению индукции, но тогда $S \in W(\mathcal{S})$ в силу пункта б) утверждения 2.21. Во втором случае обе полугруппы $T_1,\ T_2 \in W(\mathcal{S})$ и тогда в соответствии с пунктом в) утверждения 2.21 T_2 w $T_1 \in W(\mathcal{S})$. Следовательно, $S \in W(\mathcal{S})$ и поэтому $\hat{W}_{n+1}(\mathcal{S}) \subseteq W(\mathcal{S})$. Отсюда вытекает, что $\hat{W}(\mathcal{S}) \subseteq W(\mathcal{S})$.

Предупреждение. Хотя имеет место равенство $\hat{W}(\mathcal{S}) = W(\mathcal{S})$, это не означает, что $\hat{W}_i(\mathcal{S}) = W_i(\mathcal{S})$ для каждого *i;* поэтому, если проводится доказательство при помощи индукции по *i,* надо придерживаться какого-нибудь одного определения семейства $W(\mathcal{S})$.

Как правило, мы будем пользоваться определением семейства $W(\mathcal{S})$, данным в 2.22. (Обозначение $\hat{W}(\mathcal{S})$ далее не применяется.)

Для удобства переформулируем результаты утверждения 2.14 в терминах операции «w».

2.24. Утверждение. а) $S_2 \times S_1 \mid S_2$ w S_1.

б) $S_i \mid S_2$ w S_1, $i = 1,2$.

в) Если $S_i \mid T_i$, $i = 1,2$, то S_2 w $S_1 \mid T_2$ w T_1.

г) Если $S \in W(\mathcal{S})$, то $S \mid S_n$ w ... w S_1, где каждая полугруппа $S_i \in \mathcal{S}$, $i = 1, \ldots, n$ и $1 \leqslant n$.

2.25. Утверждение. Пусть — $\mathcal{F}$ совокупность автоматов. Тогда $SP(\mathcal{F})^S \subseteq W(\mathcal{F}^S)$.

Доказательство. Проведем доказательство индукцией по *i* относительно SP_i. Конечно, для $i = 1$ $SP_1(\mathcal{F})^S \subseteq W(\mathcal{F}^S)$, так как $SP_1(\mathcal{F}) = \mathcal{F}$. Предположим, что $SP_n(\mathcal{F})^S \subseteq W(\mathcal{F}^S)$. Пусть $f \in SP_{n+1}(\mathcal{F})$.

Тогда или 1) $f = h_2 g h_1^r$, где $g \in SP_n(\mathcal{F})$, или 2) $f = f_2 \times f_1$, где f_2, $f_1 \in SP_n(\mathcal{F})$, или 3) $f = f_2 h^r f_1^\sigma$, где $f_2, f_1, \in SP_n(\mathcal{F})$.
В случае 1) $f^S | g^S$ и по индукции $g^S \in W(\mathcal{F}^S)$, следовательно, $f^S \in W(\mathcal{F}^S)$. В случае 2) $f^S | f_2^S \times f_1^S$ и согласно пункту а) утверждения 2.24 $f_2^S \times f_1^S | f_2^S \mathrm{w} f_1^S \in W(\mathcal{F}^S)$ по индукции. Следовательно, $f^S \in W(\mathcal{F}^S)$. В случае 3) $f^S | f_2^S \mathrm{w} f_1^S$ согласно предложению 2.15, поэтому $f^S \in W(\mathcal{F}^S)$. Следовательно, $SP_{n+1}(\mathcal{F})^S \subseteq W(\mathcal{F}^S)$.
Утверждение доказано.
Что можно сказать относительно обратного утверждения: задано семейство полугрупп $\mathcal{S}$, справедливо ли включение $W(\mathcal{S})^I \subseteq SP(\mathcal{S}^I)$? В общем случае ответ будет отрицательным, но если добавить определенные базисные автоматы в *SP* и определенные базисные полугруппы в *W*, то утверждение будет справедливо.
Для того чтобы развить эту идею, определим некоторые полугруппы и автоматы, играющие важную роль в нашей теории.

2.26. Определениею а) Пусть *A* — непустое множество. Напомним определение полугрупп A^l и A^r.
б) Определим полугруппы $U_3 = \{r_0, r_1\}^{r1}$, $U_2 = \{r_0\}^1$, $U_1 = \{r_0, r_1\}^r$, и $U_0 = \{1\}$. Полугруппы U_0, U_1 и U_2 представляют собой собственные подполугруппы полугруппы U_3 (с точностью до изоморфизма). Если S | U_3, то $S \cong U_i$ для некоторого i = 0, 1,2, 3.
в) *Автомат задержки* $D_A : \Sigma A \to A \cup \{*\}$ определяется при помощи соотношений $D_A(a_1) = *$ (произвольный символ) и $D_A(a_1, \ldots, a_n) = a_{n-1}$.

г) *Автомат задержки* D_1 — это автомат D_A, для которого $A = \{r_0, r_1\}$ и $* = 1$.
д) *Автомат* $2_A : \Sigma A \to (A \cup \{*\}) \times A$ определяется при помощи соотношений $2_A(a_1) = (*, a_1)$ и $2_A(a_1, \ldots, a_n) = (a_{n-1}, a_n)$.

2.27. Замечание. Пусть *A* — непустое множество. Естественным расширением автомата A^{rf} будет тождественное отображение на множестве ΣA. Полугруппа A^r может быть записана как подполугруппа прямого произведения подходящего числа полугруппы U_1. Следовательно, $A^{rf} | U_1^l \times \ldots \times U_1^l$ (*lp*), поэтому $A^{rf} \in SP(U_1^l)$.

2.28. Утверждение. Пусть $U_1^l \in \mathcal{F}$. Тогда $SP(\mathcal{F})$ есть множество всех автоматов *f*, представляемых в виде

$$f = h_{n+1} f_n h_n^r f_{n-1}^\sigma \ldots h_2^r f_1^\sigma h_1^r,$$

где каждый автомат f_i — конечная параллельная композиция элементов из $\mathcal{F}$, а h_i — отображения.

Доказательство. Вновь проводим доказательство по индукции по i относительно SP_i. Основной этап доказательства состоит в следующем. Предположим, что утверждение справедливо для $SP_n(\mathcal{F})$ и что автомат $f \in SP_{n+1}(\mathcal{F})$ записан в виде $f = f_2 \times f_1;\ f_1, f_2 \in SP_n(\mathcal{F})$. Пусть $f_1 : \Sigma A \to B,\ f_2 : \Sigma C \to D$. Тогда
$f = (f_2 \times B^{rf}) i^{\Gamma} (C^{rf} \times f_1)^{\sigma}$, где i — тождественное отображение на множестве $C \times B$. Так как автомат $f_1 \in SP_n(\mathcal{F})$, его можно представить в виде

$$f_1 = h_{n+1}\, g_n\, h_n^{\Gamma} \ldots h_2^{\Gamma}\, g_1\, h_1^{\Gamma}.$$

Благодаря этому нетрудно показать, что автомат $C^{rf} \times f_1$ можно переписать как $\overline{h}_{n+1}\,(C^{rf} \times g_n)\,\overline{h}_n^{i^r} \ldots \overline{h}_2^{\Gamma}\,(C^{rf} \times g_1)^{\sigma}\overline{h}_1^{\Gamma}$. Теперь автомат $C^{rf} \times g_i$ представляет собой конечную параллельную композицию элементов из $\mathcal{F}$, так как $U_1^l \in \mathcal{F}$. Проводя такие же рассуждения для автомата $f_2 \times B^{rf}$, мы видим, что автомат f удовлетворяет условию предложения, истальные этапы доказательства тривиальны.

2.29. Утверждение. Пусть A — непустое множество.
а) $A^{rf} \in SP(U_1^l)$.
б) $D_A \in SP(D_1)$.
в) $2_A \in SP(D_1, U_1^l)$.

Доказательство, а) Справедливость этого пункта была показана в замечании 2.27.
б) Как указано в замечании 2.27, запишем A как подмножество прямого произведения подходящего числа множеств $\{r_0, r_1\}$. Тогда легко видеть, что автомат D_A представляет собой параллельную композициютого же самого числа автоматов D_1. Следовательно, $D_A \in SP(D_1)$.
в) Проверим, что $2_A = (D_A \times A^{rf})h^{\Gamma}$, где отображение $h : A \to A \times A$ задается соотношением $h(a) = (a, a)$.
Следовательно, в силу пунктов а) и б) $2_A \in SP(D_1, U_1^l)$.
Теперь можно перейти к доказательству результата, обратного предложению 2.15, затем мы получим важное следствие.

2.30. Предложение. Пусть имеются полугруппы S_1 и S_2, а $S_2 \times {}_{\gamma}S_1$ — произвольное полупрямое произведение S_1 и S_2. Тогда $(S_2 \times {}_{\gamma}S_1)^l \in SP(S_1^l, S_2^l, D_1, U_1^l)$.

Доказательство. Проверим, что
$(S_2 \times {}_{\gamma}S_1)^l = (S_2^l \times S_1^{rl})\, h^{\Gamma}\, 2^{\sigma}_{S_2 \times S_1}\, (S_2^{rl} \times S_1^l)^{\sigma}$,
Где $h : [(S_2 \times S_1) \cup \{*\}] \times (S_2 \times S_1) \to S_2 \times S_1$, причем

$$h[*, (s_2, s_1)] = (s_2, s_1)$$

$$\text{и } h\ [(s_2, s_1), (t_2, t_1)] = [Y\ (s_1)\ t_2, t_1].$$

Следовательно, в силу предыдущих утверждений и замечаний получаем требуемый результат.

2.31. Следствие. Пусть (X_1, S_1) и (X_2, S_2)— полугруппы правых отображений. Тогда $[(X_2, S_2) \mathrm{w} (X_1, S_1)]^I \in SP\ (S_1^I, S_2^I, D_1, U_1^I)$.

Доказательство. Требуемый результат получается благодаря изоморфизмам $(X_2, S_2)\ \mathrm{w}\ (X_1, S_1) \cong F\ (X_1, S_2) \times {}_Y S_1$ и $F\ (X_1, S_2) \cong S_2 \times \ldots \times S_2$ ($|X_1|$ раз).

2.32. Утверждение. $D_1^S,\ U_1 \in W\ (U_3)$.

Доказательство. Очевидно, что $U_1 \in W\ (U_3)$, так как $U_1 | U_3$.

Проверим, что $D_1 = U_3^I h^\Gamma\ (U_3^I \times U_1^I)^\sigma H$, где:

1) $H : \Sigma\ U_1 \to \Sigma\ (U_3 \times U_1)$— гомоморфизм, определяемый соотношением $H\ (x) = [(1, r_0), (x, r_1)]$ для всех $x \in U_1$;

2) h: $U_3 \times U_1 \to U_3$, где $h\ (x, r_0) = x$ и $h\ (x, r_1) = 1$ для всех $x \in U_3$.

Следовательно, согласно предложению 2.15 имеем

$$D_1^S | U_3\ \mathrm{w}\ (U_3^I \times U_1^I)^S | U_3 \mathrm{w} (U_3 \times U_1).$$

Таким образом, $D_1^S \in W(U_3)$.

2.33. Замечание. Отметим, что из соотношения для автоматов

$$D_1 = U_3^I h^\Gamma\ (U_3^I \times U_1^I)^\sigma H$$

не вытекает, что $D_1 \in SP\ (U_3^I)$, поскольку гомоморфизм H не обязательно сохраняет) длину. В действительности справедливо, что $D_1 \notin SP\ (U_3^I)$. В результате становится ясной необходимость присутствия D_1 в приведенном далее соотношении 2.2.

2.34. Предложение. Пусть $\mathscr{S}$— совокупность полугрупп. Тогда

$$W\ (\mathscr{S} \cup \{U_3\}) = SP\ (\mathscr{S}^I \cup \{D_1, U_3^I\})^S. \quad (2.2)$$

Доказательство. На основе следствия 2.31, а также применяя метод математической индукции, покажем, что $W\ (\mathscr{S})^I \subseteq SP\ (\mathscr{S}^I \cup \cup\ \{D_1, U_1^I\})$. Из утверждения 2.25 и равенства $W\ (\mathscr{S})^{IS} = W\ (\mathscr{S})$ вытекают включения

$$W\ (\mathscr{S}) \subseteq SP\ (\mathscr{S}^I \cup \{D_1, U_1\})^S \subseteq W\ (\mathscr{S} \cup \{D_1^S, U_1\}).$$

Но в силу утверждения 2.32 $W\ (\mathscr{S} \cup \{D_1^S, U_1\}) \subseteq W\ (\mathscr{S} \cup \{U_3\})$. Поэтому, добавляя слева к $W\ (\mathscr{S})$ полугруппу U_3 и заменяя U_1^I на U_3^I в SP, получаем равенство

$$W\ (\mathscr{S} \cup \{U_3\}) = SP\ (\mathscr{S}^I \cup \{D_1, U_3^I\}).$$

2.5.3. Доказательство основной теоремы декомпозии для конечных полугрупп и автоматов

Пусть f — автомат, а $f^s = S$ — его полугруппа. Естественным представляется следующий вопрос: какие полугруппы должны принадлежать совокупности $\mathcal{S}$ для того, чтобы полугруппа S принадлежала $W(\mathcal{S})$, а точнее—каковы наименьшие ингредиенты (полугруппы), необходимые для $\mathcal{S}$, чтобы $S \in W(\mathcal{S})$? Аналогичный вопрос возникает для автоматов: какими должны быть элементарные автоматы $\mathcal{F}$, чтобы $f \in SP(\mathcal{F})$?

Очевидно, что $S \in W(S)$. Предположим, что существуют полугруппы S_1 и S_2, такие, что $S \mid S_2 \text{ w } S_1$, но S не делит ни S_1, ни S_2. Тогда $S \in W(S_1, S_2)$. Предположим далее, что $S_i \mid T_i \text{ w } W_i$, но S_1 не делит ни T_i, ни W_i, $i = 1,2$. Тогда, поскольку

$S \mid S_2 \text{ w } S_1 \mid (T_2 \text{ w } W_2) \text{ w } (T_1 \text{ w } W_1)$, имеем

$S \in W(T_1, T_2, W_1, W_2)$.

3.**1. Определение**. Полугруппа S называется *неприводимой,* если из соотношения $S \mid S_2 \text{w} S_1$ вытекает, что $S|S_1$ или $S \mid S_2$, каковы бы ни были полугруппы S_1 и S_2. Символ IRR обозначает множество всех неприводимых полугрупп.

Мы увидим, что для заданной полугруппы S существует совокупность полугруппы $\mathcal{S} \subseteq \text{IRR}$, такая, что $S \in W(\mathcal{S})$. Возникают следующие вопросы:

1) какие полугруппы являются неприводимыми?

2) пусть задана полугруппа S; каким необходимым и достаточным условиям должна удовлетворять совокупность неприводимых полугрупп $\mathcal{S}$ для того, чтобы $S \in W(\mathcal{S})$.

3.2. Определение. а) Символ PRIMES обозначает совокупность конечных нетривиальных простых групп. (Мы предполагаем, что читатель знаком с основными фактами теории групп, в частности, с основами теории нормальных делителей и с другим традиционным материалом до теоремы Жордана — Гельдера включительно.)

б) Символ UNITS обозначает множество всех делителей полугруппы U_3, т, е.

$$\text{UNITS} = \{U_0, U_1, U_2, U_3\}.$$

в) Пусть S — полугруппа. $\text{PRIMES}(S) = \{P \in \text{PRIMES} : P \mid S\}$.

(Отметим, что свойство простоты группы G не исключает ее делимости другой, простой группой. Если A_n знакопеременная группа на n символах, то $A_m \mid A_n$ при $m \leq n$, но A_n — простая группа при $n \neq 4$.)

Если $\mathcal{S}$ — совокупность полугрупп, определим $\text{PRIMES}(\mathcal{S}) =$

$= \bigcup \{\mathrm{PRIMES}\ (S) : S \in \mathscr{S}\}$.

г) $\mathrm{IRR}\ (S) = \{S' \in \mathrm{IRR} : S' \mid S\}$.

Оставшаяся часть этого параграфа будет посвящена доказательству основной теоремы декомпозиции.

3.3. Теорема Крона и Роудза. а) Пусть f— автомат с полугруппой f^S конечного порядка. В этом случае $f \in SP\ (\mathscr{S}^f \cup \{D_1, U_3^1\})$ тогда и только тогда, когда

$$\mathrm{PRIMES}\ (f^S) \subseteq \mathrm{PRIMES}\ (\mathscr{S}),$$

в частности

$$f \in SP\ [\mathrm{PRIMES}\ (f^S)^f \cup \{D_1, U_3^1\})]. \quad (3.1)$$

б) Пусть S — конечная полугруппа. В этом случае $S \in W\ (\mathscr{S} \cup \cup \{U_3\})$ тогда и только тогда, когда

$$\mathrm{PRIMES}\ (S) \subseteq \mathrm{PRIMES}\ (\mathscr{S}),$$

в частности

$$S \in W\ [\mathrm{PRIMES}\ (S) \cup \{U_3\})]. \quad (3.2)$$

в) $\mathrm{IRR} = \mathrm{PRIMES} \cup \mathrm{UNITS}$.

г) В общем случае $S \notin W\ [\mathrm{IRR}(S)]$.

3.4. Замечание. Пункт г) теоремы объясняет как разницу между элементами множеств PRIMES и UNITS, так и включение U_3 в предшествующие уравнения.

3.5. Следствие. Пусть S — конечная полугруппа. Тогда $S \mid S_n$ w $\ldots wS_1$ для некоторой последовательности $S_1, \ldots, S_n$, где $S_i \in \mathrm{PRIMES}\ (S) \cup \mathrm{UNITS},\ i = 1, \ldots, n$.

Доказательство. Это вытекает из соотношения (3.2) и утверждения 2.24.

Доказательству теоремы предшествует несколько лемм. Первая лемма показывает, что PRIMES и UNITS неприводимы.

3.6. Лемма. $\mathrm{PRIMES} \cup \mathrm{UNITS} \subseteq \mathrm{IRR}$.

Доказательство. Определим IRR_{SD} как множество полугрупп S, обладающих тем свойством, что из условия $S \mid S_2 \times {}_Y S_1$ вытекает, что S | S_1 или S| S_2 для всех полугрупп S_1, S_2 и всех гомоморфизмов связи Y. Легко видеть, что $\mathrm{IRR}_{SD} \subseteq \mathrm{IRR}$. Мы покажем, что $\mathrm{PRIMES} \cup \cup \mathrm{UNITS} \subseteq \mathrm{IRR}_{SD}$.

Сначала докажем, что $\mathrm{PRIMES} \subseteq \mathrm{IRR}_{SD}$. Пусть G — нетривиальная группа, предположим, что $G \mid S_2 \times {}_Y S_1$ для некоторых полугрупп S_1, S_2. Тогда пункт в) утверждения 1.19 гарантирует существование нетривиальной подгруппы G' в $S_2 \times {}_Y S_1$, гомоморфным об-

разом которой является группа *G*. Пусть p_1 — ограничение на G' отображения проекции на первую координату, т. е. $(s_2, s_1) \to s_1$. (Здесь нумерация координат соответствует нумерации полугрупп, а не месту, которое занимает координата в декартовом произведении.) Так как p_1 — гомоморфизм, $p_1(G') = G_1$ будет подгруппой полугруппы S_1. Пусть (e_2, e_1) — единица группы G'. Тогда $\ker p_1$ является нормальным делителем в G', состоящим из всех элементов в *G'*, которые имеют e_1 в качестве первой координаты. Пусть отображение $\varphi : \ker p_1 \to S_2$ задается соотношением $\varphi(s_2, e_1) = Y(e_1)\, s_2$. Тогда φ будет мономорфизмом; для того чтобы доказать взаимную однозначность отображения φ, предположим, что $\varphi(s_2, e_1) = \varphi(t_2, e_1)$. Тогда $(s_2, e_1) = (e_2, e_1)(s_2, e_1) = [e_2\, Y(e_1) s_2,\ e_1] = [e_2 Y(e_1)\ t_2,\ e_1] = (e_2, e_1)(t_2, e_1) = (t_2, e_1)$. Пусть $G_2 = \varphi(\ker p_1)$. Тогда группа *G'* является расширением подгруппы G_2 полугруппы S_2 при помощи подгруппы G_1 полугруппы S_1.

Предположим теперь, что $G \in$ TPRIMES и θ—такой гомоморфизм, что справедливо соотношение $\theta(G') = G$. Тогда $K = \ker \theta$ есть максимальный нормальный делитель подгруппы G' и $K \cdot \ker p_1$ совпадает или с *K*, или с G'. Если $K \cdot \ker p_1 = K$, то $G \mid G_1 \subseteq S_1$; если $K \cdot \ker p_1 = G'$, то $G \mid G_2 \subseteq S_2$. Тем самым доказано, что

$$\text{PRIMES} \subseteq \text{IRR}.$$

Докажем теперь, что $U_3 \in \text{IRR}_{SD}$. Доказательство неприводимости остальных элементов из множества UNITS аналогично, но проще. Прежде всего покажем, что из условия $U_3 \mid S$ вытекает, что U_3 представляет собой подполугруппу полугруппы S. Пусть $S' \subseteq S$ и φ — гомоморфизм, отображающий S' на U_3. Пусть элемент $s \in S'$ и $\varphi(s) = 1$. Тогда найдется идемпотент *e,* который будет некоторой степенью элемента s и, следовательно, $\varphi(e) = 1$. Вследствие этого $\varphi(eS'e) = U_3$ и элемент *e* является единицей для *eS'e*. Пусть S_1—подполугруппа в *e S'e* наименьшего порядка, для которой $\varphi(S_1) = U_1$. Тогда S_1 — простая справа полугруппа в силу пункта в) утверждения 1.19 и поэтому полугруппа S_2 изоморфна $G \times B^r$, где G — группа, а B — непустое множество. Множество *B* должно содержать по крайней мере два различных элемента b_1 и b_2, так как U_1 не является группой. Тогда

$$U_3 \cong \{e, (1, b_1), (1, b_2)\} \subseteq S.$$

Предположим теперь, что $U_3 \mid (S_2 \times_Y S_1)$. В силу ранее изложенного $U_3 = \{(b_1, a_1), (b_0, a_0), (b_1, a_1)\} \subseteq S_2 \times_Y S_1$. Как и раньше, обозначим через p_1 гомоморфизм вида $p_1(b, a) = a$. $p_1(U_3) = \{a_1, a_0, a_1\} \subseteq S_1$. Если $a_0 \neq a_1$, то полугруппа

$p_1 (U_3)$ изоморфна U_3 и $U_3 \mid S_1$. В самом деле, для $i = 0$ или $i = 1$ из $a_1 = a_i$ вытекает, что $za_1 = za_i$, откуда следует, что $z = a_i$ для всех $z \in \{a_1, a_0, a_1\}$.

Предположим теперь, что $a_0 = a_1$ тогда необходимо, чтобы $b_0 \neq b_1$ Пусть отображение $p_2 : U_3 \to S_2$ определяется как $p_2 (b, a) = Y (a_0) (b) \equiv {}^{a_0}b$. Заметим, что $Y (a_0)\ Y (a_1) = Y (a_0)$, тогда отображение p_2 будет гомоморфизмом. Кроме того, p_2 взаимно однозначно. В самом деле, из $(b_1, a_0) = (b_0, a_0)\ (b_1, a_0)$ вытекает, что $b_1 = b_0^{a_0}\ b_1$, а из $(b_0, a_0) = (b_0, a_0)\ (b_0, a_0)$ следует, что $b_0 = b_0^{a_0}\ b_0$.

Поэтому если $b_0 \neq b_1$, то ${}^{a_0}b_0 \neq {}^{a_0}b_1$. Далее из равенства $(b_0, a_0) = (b_0, a_0)\ (b_1, a_1)$ вытекает, что $b_0 = b_0^{a_0}\ b_1$, поэтому ${}^{a_0}b_1 \neq {}^{a_0}b_1$.

Аналогично ${}^{a_0}b_0 \neq {}^{a_0}b_1$. Следовательно, $U_3 \cong p_2 (U_3)$ и $U_3 \mid S_2$. Тем самым доказано, что $\mathrm{UNITS} \subseteq \mathrm{IRR}$.

Теперь перейдем к доказательству справедливости соотношения (3.1). Оно будет вестись с помощью индукции по порядку полугруппы f^S. Для основного шага индукции возможны три различные ситуации.

3.7. Лемма. Пусть S — конечная полугруппа. Тогда выполняется одно из следующих условий:

а) полугруппа S простая слева;

б) полугруппа S циклическая;

в) существуют такой собственный левый идеал $V \subset S$ и собственная подполугруппа $T \subset S$, что $S = V \cup T$.

Доказательство. Если полугруппа S простая слева, то все в порядке. Следовательно, можно предположить, что условие а) не выполняется, т. е. S не является простой слева. Так как полугруппа S конечна, она содержит максимальный левый идеал $L \neq S$. Пусть $a \in S - L$. Тогда $S = L \cup S^1 a$.

Если $S^1 a \neq S$, то справедлив случай в), где $T = S^1 a$ и $V = L$. Поэтомумы можем предположить, что $S^1 a = S$. Если $a \notin Sa$, то $S = Sa \cup \{a, a^2, \ldots\}$.

Если $S = \{a, a^2, \ldots\}$, то имеет место случай б), иначе справедлив случай в), где $V = Sa$ и $T = \{a, a^2, \ldots\}$. Следовательно, можно считать что $a \in Sa$. Тогда $a = ua$ для некоторого элемента $u \in S$. Пусть $K = \{k \in S : ka \in L\}$.

Так как $L \subseteq Sa\ (= S)$, $K \neq \phi$.

Очевидно, что множество K представляет собой левый идеал полугруппы S и $K \neq S$, так как $u \notin K$. Поскольку идеал L является максимальным, то или $K \cup L = L$, или $K \cup L = S$. Если верно

последнее соотношение, то справедлив случай в), поэтому будем считать, что $K \subseteq L$. Тогда из соотношения $xa \in L$ вытекает, что $x \in L$. Так как это относится к каждому элементу $a \in S - L$, то очевидно, что множество $S - L$ будет подполугруппой полугруппы S и вновь справедлив случай в). Другое доказательство утверждения содержится в лемме 4.1. Оно принадлежит А. Клиффорду.

3.8. Лемма. Пусть полугруппа f^S простая слева. Тогда соотношение (3.1) справедливо.

Доказательство. Из условия леммы вытекает, что $f^S = G \times A^I$, где G— некоторая группа, A — непустое множество. $\mathrm{PRIMES}\,(f^S) = \mathrm{PRIMES}\,(G)$. Далее если $P \in \mathrm{PRIMES}(f^S)$, то $P \mid f^S = G \times A^I$, поэтому или $P|G$, или $P \mid A^I$, так как P неприводима. Но A^I не содержит нетривиальных групп, поэтому $P \mid G$ и $P \in \mathrm{PRIMES}\,(G)$. Наоборот, пусть $P \in \mathrm{PRIMES}\,(\mathrm{G})$. Очевидно, что из соотношения $P \mid G$ вытекает, что $P \mid G \times A^I$, поэтому $P \in \mathrm{PRIMES}\,(f^S)$. Следовательно,

$$\mathrm{PRIMES}\,(\mathrm{G}) = \mathrm{PRIMES}\,(f^S).$$

Пусть $L = \{r_0,\ r_1\}^I$. Проверим тогда, что $L^I = h_3 U_3^I h_2^\Gamma (D_1 \times \times U_1^I)^\sigma h_1^\Gamma$,

где

1) $h_1 : L \to U_1 \times U_1, \quad h_1(r_i) = (r_i,\ r_i),\ i = 1,\ 2;$

2) $h_2 : U_3 \times U_1 \to U_3,\ h_2(1,\ x) = x$ и $h_2(r_i,\ x) = 1,\ i = 1,\ 2;$

3) $h_3 : U_3 \to L,\ h_3(r_i) = r_i,\ i = 1,2.$ Значение h_3 (1) не определено. Следовательно, $L^I \in SP\ (D_1,\ U_3^I)$. Теперь A^I будет подполугруппой в прямом произведении подходящего числа полугрупп L. Из этого следует, что $A^{If} \in SP\ (D_1,\ U_3^I)$.

Таким образом, остается только проверить справедливость соотношения (3.1) для G^I. Если оно справедливо, имеем

$$f^{SI} = G^I \times A^{II} \in SP\,[\mathrm{PRIMES}\,(G)^I \cup \{D_1,\ U_3^I\}],$$

и поскольку $f \mid f^{Sf}\ (lp)$ и $\mathrm{PRIMES}\,(G) = \mathrm{PRIMES}\,(f^S)$, то утверждение верно.

Проверим, что $G \in W\,[\mathrm{PRIMES}(G)]$. Это, конечно, так, когда G — простая группа. Предположим, что группа не является простой. Пусть G_1 — нормальный делитель группы G. Положим $H = G/G_1$ и обозначим через $N : G \to\!\!\!\to H$ канонический эпиморфизм. Пусть $\{\overline{h} \in \in N^{-1}\,(h) : h \in H\}$ есть множество представителей смежных классов по нормальному делителю G_1 при условии, что $\overline{1} = 1$. Пусть далее $(g_1,\ h) \in G_1 \times H$. Для каждого элемента $g \in G$ определим $\hat{g} \in F_R\,(G_1 \times H)$, полагая

$$(g_1, h)\,\hat{g} = [g_1\,\bar{h}g\,(\overline{hN(g)})^{-1},\ hN(g)].$$

Легко проверить, что, во-первых, $\hat{g} \in G_1$ w H и, во-вторых, отображение $g \to \hat{g}$ является изоморфизмом. Следовательно, G | G$_1$wH. Отображение, определенное здесь, представляет собой мономиальное отображение в теории групп, оно было введено Бернсайдом.

Пусть последовательность $G = G_0 \triangleright G_1 \triangleright \ldots \triangleright G_n = \{1\}$ будет композиционным рядом группы G с фактор-группами $H_i = G_{i-1}/G_i$, $i = 1, \ldots, n$. Тогда $G \mid G_1 \mathrm{w} H_1$ и $G_1 \mid G_2 \mathrm{w} H_2$ и т. д. Следовательно, $G \mid H_n$ w ... w H_1 и $G \in W(H_1, \ldots, H_n)$. Группы H_i как композиционные факторы будут простыми и поэтому $H_i \in W[\mathrm{PRIMES}(H_i)]$ для каждого $i = 1, \ldots, n$.

Следовательно,

$$G \in W(\mathrm{PRIMES}\{H_1, \ldots, H_n\}).$$

Однако $\mathrm{PRIMES}(G) = \mathrm{PRIMES}\{H_1, \ldots, H_n\}$. Далее, если $P \in \mathrm{PRIMES}(G)$, то $P \mid G \mid H_n$ w ... wH_1, откуда следует, что $P \mid H_i$ для некоторого i. Следовательно, $P \in \mathrm{PRIMES}\{H_1, \ldots, H_n\}$. (Это, однако, не означает, что P совпадает с одной из групп H). Наоборот, пусть $P \in \mathrm{PRIMES}\{H_1, \ldots, H_n\}$, т. е. $P \mid H_i$ для некоторого i. Но $H_i = G_{i-1}/G_i$, т. е. является гомоморфным образом подгруппы G_{i-1} группы G, поэтому $P \mid H_i \mid G$. Следовательно, имеется включение $P \in \mathrm{PRIMES}(G)$. Значит, $G \in W[\mathrm{PRIMES}(G)]$ и

$$G' \in SP(\mathrm{PRIMES}(G)' \cup \{D_1, U_2^!\}).$$

Лемма полностью доказана.

3.9. Лемма. Пусть f^s — циклическая полугруппа. Тогда соотношение (3.1) справедливо для f.

Доказательство. Пусть символ $C_{(n,m)}$ обозначает циклическую полугруппу индекса n и периода m. Как известно, все циклические полугруппы представляются в таком виде. Пусть $\mathbf{Z}_m$ — циклическая группа порядка m. Тогда $C_{(n\,m)} \subseteq \mathbf{Z}_m \times C_{(n,1)}$ и эта полугруппа порождается элементом (a, b), где элемент a порождает группу $\mathbf{Z}_m$, а элемент b — полугруппу $C_{(n,1)}$. Кроме того, $C_{(n,1)} \subseteq T_n = U_2$ w ...w U_2 (n раз). Это доказывается при помощи индукции по n и на основе того факта, что узловое произведение моноидов является моноидом. Пусть q_k порождает $C_{(k,1)}$ и пусть 1k — единица полугруппы T_k. Для $n = 1$ имеем $C_{(1,1)} = \{r_0\} \subseteq U_2$. Предположим теперь, что $C_{(n-1,\,1)} \subseteq T_{n-1}$. Пусть $q_n = (f, r_0) \in T_{n-1}$ w $U_2 = T_n$, где отображение $f : U_2 \to T_{n-1}$ задается соотношениями $f(r_0) = q_{n-1}$ и $f(1) = 1_{n-1}$. Из этого следует, что q_n порождает

$C_{(n,1)} \subseteq T_n$. Следовательно, получаем, что
$C_{(n,m)} \mid \mathbf{Z}_m \times T_n \in \mathcal{W}[\mathrm{PRIMES}(\mathbf{Z}_m), U_2]$, так как $T_n \in \mathcal{W}(U_2)$ и $\mathbf{Z}_m$ — циклическая группа. Но $\mathrm{PRIMES}(\mathbf{Z}_m) =$
$= \mathrm{PRIMES}(C_{(n,m)})$.
Следовательно, $f^{Si} = C^f_{(n,m)} \in SP[\mathrm{PRIMES}(f^S)^f \cup \{D_1, U^f_3\}]$, а так как $f \mid f^{Si}(lp)$, то утверждение полностью доказано.
Прежде чем приступить к рассмотрению случая в) леммы 3.7, введем следующие определения,

3.10. Определение. Пусть $f : \Sigma A \to B$— автомат и *c*— некоторый символ, не принадлежащий множествам *A* и *B*. Тогда *частичным произведением PPf* автомата *f* называется автомат, задающий отображение $PPf : \Sigma(A \cup \{c\}) \to B \cup \{c\}$, причем $PPf(\alpha) = f(\alpha_c)$, где α_c — элемент моноида $(\Sigma A)^1$ и α_c получается в результате вычеркивания в последовательности α всех элементов, стоящих перед последним *c*, а также самого *c* (т. е. любой элемент $\alpha \in \Sigma(A \cup \{c\})$ можно представить как $\alpha = \alpha_1 c \alpha_2$, где $\alpha_2 \in \Sigma A$, тогда.
$PPf(\alpha) = f(\alpha_2)$ и $\alpha_2 = \alpha_c$). Положим, кроме того, $f(1) = c$.

3.11. Определение. Пусть имеется автомат $f : \Sigma A \to B$ и *e*—элемент, не принадлежащий множествам *A* и *B*. Определим тогда автомат $ef : \Sigma(A \cup \{e\}) \to B \cup \{e\}$, полагая $ef(\alpha) = f(\alpha_e)$, где α_e— элемент моноида $(\Sigma A)^1$. Он получается в результате вычеркивания всех элементов *e* из последовательности α (т. е. любой элемент $\alpha \in \Sigma(A \cup \{e\})$ представляется в виде $\alpha = \alpha_1 e \alpha_2 e \ldots e \alpha_k$, где $\alpha_i \in \Sigma A$ для каждого $1 \leqslant i \leqslant k$; тогда
$\alpha_e = \alpha_1 \alpha_2 \ldots \alpha_k$ и $ef(\alpha) = f(\alpha_1, \ldots, \alpha_k)$.
Положим, кроме того, $f(1) = e$.

3.12. Лемма. Пусть S—полугруппа с такими собственным левым идеалом V и собственной подполугруппой *T*, что $S = V \cup T$.
Тогда
$S^f \in SP_{\_}(eV^f, PPT^f, D_1, U^f_1)$.

Доказательство. Проверим, что
$S^f = h_3(eV^f \times T'^{rf}) h_2^r 2^{\sigma}_{V' \times T'} (V'^{rf} \times PPT^f)^{\sigma} h_1^r$,
где $T' = T \cup \{c\}$, $V' = V \cup \{e\}$
и где
1) $h_1 : S \to V' \times T'$, причем
$h_1(s) = \begin{cases} (s, c), & \text{если } s \in V, \\ (e, s), & \text{если } s \in S - V \subseteq T; \end{cases}$
2) $h_2 : [(V' \times T') \cup \{*\}] \times (V' \times T') \to V' \times T'$, причем
$h_2[*, (v, t)] = (v, t)$ и

$h_2\,[(v_1,\ t_1),\ \ (v_2,\ t_2)] = \begin{cases} (t_1 v_2,\ t_2), \text{ если } t_2 = c \text{ и } t_1 \neq c, \\ (v_2,\ t_2) \text{ в противном случае;} \end{cases}$

3) $h_3 : V' \times T' \to S$, где $h_3'(v,\ t) = vt$ и элементы e и c действуют как единицы. Элемент *(e, c)* не будет встречаться.

3.13. Лемма. Если соотношению (3.1) удовлетворяет автомат *f*,то ему также удовлетворяет автомат *ef*.

Доказательство. Докажем сначала с помощью метода математической индукции, что если $U_1^f \in \mathcal{F}$, то из включения $\{ef : f \in \mathcal{F}\} \subseteq \subseteq SP\ (\mathcal{F})$ вытекает включение $\{ef : f \in SP\ (\mathcal{F})\} \subseteq SP\ (\mathcal{F})$. Пусть $f \in SP_n\ (\mathcal{F})$. Тогда или 1) $f = f_2 h^\Gamma f_1^\sigma$, или 2) $f = f_2 \times f_1$, или 3) $f = h_2 f_1 h_1^\Gamma$, где $f_1,\ f_2 \in SP_{n-1}\ (\mathcal{F})$. По предположению индукции $ef_1,\ ef_2 \in SP\ (\mathcal{F})$. Пусть в первом случае $f_1 : \Sigma A \to B;\ f_2 : \Sigma C \to D$. Тогда проверим, что

$$ef = (ef_2)\, h_2^\Gamma\, (ef_1 \times A'^{r_1})^\sigma\, h_1^\Gamma,$$

где

$A' = A \cup \{e\},\quad B' = B \cup \{e\},\quad C' = C \cup \{e\}$ и $h_1 : A' \to A' \times A'$, где $h_1(a) = (a,\ a)$ для всех $a \in A'$ и $h_2 : B' \times A' \to C'$,

где

$$h_2(b,\ a) = \begin{cases} e, & \text{если } a = e, \\ h(b) & \text{в противном случае.} \end{cases}$$ •?

Следовательно, в этом случае $ef \in SP\ (\mathcal{F})$.

Доказательства случаев 2 и 3 очевидны.

Если теперь автомат *f* удовлетворяет соотношению (3.1), то в силу изложенного ранее мы должны доказать только, что

$$\{eD_1,\ eU_3^f,\ \ eG^f : G \in \mathrm{PRIMES}\,(f^S)\} \subseteq SP\,[\mathrm{PRIMES}\,(f^S)^f \cup \{D_1,\ U_3^f\}].$$

а) Во-первых, пусть S — моноид. Тогда $eS^f \in SP\ (S^f,\ U_3^f)$. Проверим, что

$$eS^f = h_2\,(S^f \times U_3^f)\, h_1^\Gamma,$$

где отображение $h_1 : S \cup \{e\} \to S \times U_3$ определяется как $h_1\,(e) = (1,1)$ и $h_1\,(s) = (s,\ r_0)$, отображение $h_2 : S \times U_3 \to S \cup \{e\}$ определяется как

$h_2\ (s,\ 1) = e$ и $h_2\ (s,\ r_0) = s$.

б) $eD_1 \in SP\,(D_1,\ U_3^f)$. Проверим, что

$$eD_1 = h_4\,(U_3^f \times U_3'^f \times U_3'^f)\, h_3^\Gamma\, 2_A^\sigma\, (U_3^f \times U_3^f \times U_3'^f)^\sigma\, h_2^\Gamma\, 2_A^\sigma h_1^\Gamma,$$

где $A = U_3 \times U_3 \times U_3$ и

1) $h_1 : U_1 \cup \{e\} \to A$, где $h_1\,(e) = (1,\ 1,\ 1),\ h_1\,(r_i) = (1,1,r_i),\ i = 0,1$;

2) $h_2 : A \cup \{*\} \times A \to A$, где $h_2\,[*,\ (1,\ 1,\ x)] = (1,\ 1,x)$,

$$h_2\,[(1,\ 1,\ x),\ (1,\ 1,\ y)] = (1,\ x,y)$$

для всех $x, y \in U_3$;

3) $h_3 : A \cup \{*\} \times A \to A$, где $h_3[*, (1, x_1, y_1)] = (1, x_1, y_1)$,

$$h_3[(1, x_1, y_1), (1, x_2, y_2)] = \begin{cases} (1, x_2, y_2), & \text{если } y_1 = 1, \\ (x_1, x_2, y_2), & \text{если } y_1 \neq 1, \end{cases}$$

$$x_i, y_i \in U_3;$$

4) $h_4 : A \to U_3 \cup \{e\}$, где $h_4(1,1,1) = e$, $h_4(1,1,r_i) = h_4(1, r_i, 1) = 1$,

$$h_4(r_k, r_j, r_i) = h_4(1, r_j, r_i) = h_4(r_j, r_i, 1) = r_j, \ i, \ j, \ k = 0,1.$$

Так как полугруппа U_3 и группы — моноиды, то в силу пунктов а) и б) лемма доказана.

3.14. Лемма. Если автомат f удовлетворяет соотношению (3.1), то ему удовлетворяет автомат *PPf*.

Доказательство. Сначала по индукции докажем, что из включения $\{PPf : f \in \mathcal{F}\} \subseteq SP(\mathcal{F})$ следует включение $\{PPf : f \in SP(\mathcal{F})\} \subseteq SP(\mathcal{F})$. Пусть $f \in SP_n(\mathcal{F})$. Тогда или 1) $f = f_2 h^{\Gamma} f_1^{\sigma}$, или 2) $f = f_2 \times f_1$, или 3) $f = h_2 f_1 h_1^{\Gamma}$, где $f_1, f_2 \in SP_{n-1}(\mathcal{F})$. По предположению индукции $PPf_1, PPf_2 \in SP(\mathcal{F})$. В первом случае $PPf = PPf_2 h_c^{\Gamma} (PPf_1)^{\sigma}$, отображение h_c является расширением h и $h_c(c) = c$. Случаи 2 и 3 доказываются аналогично.

Если теперь f удовлетворяет соотношению (3.1), то в силу изложенного необходимо только показать, что

$$\{PPD_1, PPU_3^t, PPG^t : G \in \text{PRIMES}(f^S)\} \subseteq SP[\text{PRIMES}(f^S)^t \cup \{D_1, U_3^t\}].$$

а) Пусть G—группа. Тогда $PPG^t \in SP(G^t, U_3^t)$. Проверим, что

$$PPG^t = h_3(G^{r1} \times G^{r1t} \times \{c, *\}^{rt}) h_2^{\Gamma}[G^t \times (G \cup \{c\})^{rt}]^{\sigma} h_1^{\Gamma},$$

где

1) $h_1 : G \cup \{c\} \to G \times (G \cup \{c\})$, где $h_1(g) = (g, g)$, $h_1(c) = (1, c)$;

2) $h_2 : G \times (G \cup \{c\}) \to G \times (G \cup \{1\}) \times \{c, *\}$, где 1 есть единичный элемент в G^{r1} и $h_2(g, c) = (g, g, c)$, $h_2(g, g') = (g, 1, *)$;

3) $h_3 : G \times (G \cup \{1\}) \times \{c, *\} \to G \cup \{c\}$, где $h_3(x, y, c) = c$

для всех x, y,

$$h_3(x, y, *) = \begin{cases} x, & \text{если } y = 1, \\ y^{-1} x, & \text{если } y \neq 1. \end{cases}$$

Пусть G^{r1}— подполугруппа прямого произведения достаточно большогочисла полугрупп U_3^t. Следовательно, $G^{r1t} \in SP(U_3^t)$ и $PPG^t \in SP(G^t, U_3^t)$;

б) $PPU_3^1 \in SP\,(U_3^1)$. Проверим, что
$PPU_3^1 = h_2\,(A^{r1f} \times U_1^1)\,h_1^{\Gamma}$, где $A = \{r_0, r_1, c\}$ и

1) $h_1 : U_3 \cup \{c\} \to A \cup \{1\} \times U_1$, где

$$h_1\,(c) = (c, r_1),$$
$$h_1\,(u) = (u, r_0) \text{ для всех } u \in U_3;$$

2) $h_2 : A \cup \{1\} \times U_1 \to U_3 \cup \{c\}$, где $h_2\,(c, r_1) = c$,
$h_2\,(c, r_0) = h_2\,(1, r_0) = 1$, $h_2\,(r_i, r_0) = r_i$, $i = 1,2$;

в) $PPD_1 \in SP\,(D_1, U_1^1)$. Проверим, что
$PPD_1 = h_2\,(D_1 \times D_1 \times U_1^1)\,h_1^{\Gamma}$,

где

1) $h_1 : (U_1 \cup \{c\}) \to U_1 \times U_1 \times U_1$, где $h_1\,(c) = (r_0, r_1, r_1)$, $h_1\,(r_i) = (r_i, r_0, r_0)$,
2) $h_2 : U_3 \times U_3 \times U_1 \to U_3 \cup \{c\}$, где $h_2\,(x, y, r_1) = c$
для всех $x, y, \in U_3$, $h_2\,(r_0, r_1, r_0) = h_2\,(1,1, r_0) = 1$,
$h_2\,(r_i, r_0, r_0) = r_i$, $i = 1, 2$. Другие входы для h_2
невозможны.

Таким образом, доказав пункты а) — в), мы получили доказательство леммы,

3.15. Лемма. Соотношение (3.1) справедливо.

Доказательство. Достаточно доказать справедливость соотношения (3.1) для f^{Sf}, поскольку $f \mid f^{Sf}\ (lp)$. Следовательно, соотношение (3.1) достаточно доказать для автомата S^f, где S — конечная полугруппа. Будем вести доказательство индукцией по порядку полугруппы S. Случай |S| = 1 тривиален. Предположим, что (3.1) справедливо для всех полугрупп, порядок которых меньше *n*. Пусть | S | = *n*. Применим тогда к полугруппе S лемму 3.7. В случаях а) и б) получаем требуемое при помощи лемм 3.8 и 3.9 соответственно. Случай, представленный в пункте в), доказывается при помощи лемм 3.12—3.14 посредством метода математической индукции. Следовательно,

$$f \in SP\,[\mathrm{PRIMES}\,(f^S)^f \cup \{D_1, U_3^1\}].$$

3.16. Утверждение. Если $S \in \mathrm{IRR}$ и $S \in^2 W\,(\mathcal{S})$, то $S|S'$ для некоторой полугруппы $S' \in \mathcal{S}$.

Доказательство. Требуемый результат немедленно получается посредством индукции и определения IRR,

3.17. Доказательство теоремы. Применяя к автомату S^f соотношение (3.1), получаем соотношение (3.2). Соотношения (3.1) и (3.2) доказывают, что из включения $\mathrm{PRIMES}\,(f^S) \subseteq \mathrm{PRIMES}\,(\mathcal{S})$ вытекает, что $f \in SP\,(\mathcal{S}^f \cup \{D_1, U_3^1\})$, а из включения $\mathrm{PRIMES}\,(S) \subseteq \mathrm{PRIMES}\,(\mathcal{S})$ соответственно вытекает,

что $S \in W(\mathcal{S} \cup \{U_3\})$. Для того чтобы закончить доказательство пунктов а) и б), предположим, что $f \in SP(\mathcal{S}' \cup \{D_1, U_3'\})$. Тогда $f^S \in W(\mathcal{S} \cup \{U_3\})$. Пусть $P \in \mathrm{PRIMES}(f^S)$. Тогда $P \mid f^S$, откуда $P \in W(\mathcal{S} \cup \{U_3\})$, и, следовательно, в силу утверждения 3.16 получаем, что $P|S$ для некоторой полугруппы $S \in \mathcal{S}$ (P не может делить U_3). Следовательно, $P \in \mathrm{PRIMES}(\mathcal{S})$. Это доказывает пункты а) и б) теоремы.

Для того чтобы доказать пункт в), необходимо убедиться, что $\mathrm{IRR} \subseteq \subseteq \mathrm{PRIMES} \cup \mathrm{UNITS}$.

Пусть $S \in \mathrm{IRR}$. Тогда в силу пункта б) $S \in W[\mathrm{PRIMES}(S) \cup \cup \{U_3\}]$, поэтому или $S|G$, где G— группа, или $S \mid U_3$. Если $S|G$, то S — группа. Из доказательства леммы 3.8 мы узнаем, что только неприводимые группы являются простыми. Следовательно, $S \in \mathrm{PRIMES}$ или $S = \{1\}$, т. е. S принадлежит UNITS. Если $S \mid U_3$, то S является элементом UNITS. Это доказывает пункт в).

Для того чтобы доказать пункт г) теоремы, рассмотрим полугруппу $L = \{r_0, r_1\}'$. Тогда $\mathrm{IRR}(L) = \{U_0\}$ и так как U_0— тривиальная одноэлементная полугруппа, очевидно, что $L \notin W(U_0)$. Доказательство леммы 3.8, однако, показывает, что $L \in W(U_3)$. Тем самым основная теорема декомпозиции 3.3 полностью доказана.

3.18. Замечание. Метод исследования, примененный здесь, заключался в том, что мы брали алгебраический объект и вкладывали его в больший алгебраический объект, с которым легче работать. Формально это выражалось в понятии делимости полугрупп. Соотношение $S \mid T$ означает, что полугруппа S представляет собой гомоморфный образ подполугруппы из T. Естественно спросить: какие результаты, аналогичные полученным, будут справедливы, если мы положим $S \mid T$ тогда и только тогда, когда S — подполугруппа полугруппы T. Оказывается, что таких результатов не много.

При данном определении делимости полугруппа $F_R(X_n)$ всех отображений на множестве из n символов неприводима относительно полупрямых произведений, а следовательно, и относительно узловых произведений. Так как каждая полугруппа является подполугруппой в $F_R(X_n)$ при некотором n, мы оказываемся в ситуации, когда каждая полугруппа делит неприводимую полугруппу. Следовательно, это определение делимости не удобно.

Следующие утверждения доказывают, что полугруппа $F_R(X_n)$ неприводима при данном ранее определении делимости.

3.19.Утверждение. Пусть K—ядро полугруппы $F_R(X_n)$ и $\varphi: F_R(X_n) \to S$ — гомоморфизм. Тогда:

а) если отображение φ взаимно однозначно на *K,* то φ взаимно однозначно на $F_R\ (X_n)$;

б) если φ не взаимно однозначно на $F_R\ (X_n)$, то φ *(K)* — одна точка.

Доказательство, а) Ядро полугруппы $F_R\ (X_n)$ состоит из элементов $f_i = (i, ..., i)$ постоянных функций, переводящих все в i, i — 1, ..., n. Каждый элемент f_i является правым нулем полугруппы $F_R\ (X_n)$. Пусть $f, g \in F_R\ (X_n) — K,\ f = (i_1, ..., i_n),\ g = (j_1, ..., j_n)$ —такие элементы, что $\varphi\ (f) = \varphi\ (g)$. Тогда $f_k\ f = f_{i_k}$ и $f_k g = f_{j_k}$ для каждого k. Следовательно, $\varphi\ (f_{j_k}) = \varphi\ (f_k g) = \varphi\ (f_k f) = \varphi\ (f_{i_k})$ для всех k. Так как отображение φ взаимно однозначно на K до $i_k = j_k$ для всех k, т. е. $f = g$.

б) Пусть φ не является взаимно однозначным на $F_R\ (X_n)$. Тогда в силу пункта а) φ не будет взаимно однозначным на K. Предположим, что $\varphi\ (f_1) = \varphi\ (f_2)$. Пусть f^k — функция вида $(1,\ k,\ ...) \in \in F_R(X_n),\ k = 1, ..., n$. Тогда $f_1 f^k = f_1$ и $f_2 f^k = f_k$ для всех k. Следовательно, $\varphi(f_1) = \varphi\ (f_1 f^k) = \varphi\ (f_2 f^k) = \varphi\ (f_k)$ для всех k. Утверждение доказано.

3.20.Утверждение. Предположим, что $F_R(X_n) \subseteq S_2 \times {}_Y S_1$.Тогда или $F_R\ (X_n) \subseteq S_{1,}$ или $F_R(X_n) \subseteq S_2$.

Доказательство. Рассмотрим гомоморфизм $\pi_1 : S_2 \times {}_Y S_1 \twoheadrightarrow S_1$, где $\pi_1\ (s_2,\ s_1) = s_1$. Если отображение $n_{1,}$ ограниченное на K [ядро полугруппы $F_R(X_n)$], будет взаимно однозначным, то $F_R\ (X_n) \subseteq S_1$. Следовательно, K в $S_2 \times {}_Y S_1$ имеет такой вид: $\{(s,\ 0)$:для некоторого элемента $s \in S_2$ и нулевого элемента 0 в $\pi_1\ (F_R(X_n))\}$. Пусть отображение $\varphi : S_2 \times {}_Y S_1 \to S_2$ определяется с помощью соотношения $\varphi\ (s_2,\ s_1) = {}^0 s_2$. Тогда φ — гомоморфизм. Кроме того, φ будет взаимно однозначным на $F_R\ (X_n)$. В самом деле, пусть (s, 0) и $(t,\ 0) \in K$ и предположим, что ${}^0 s = {}^0 t$.Тогда $(s,\ 0) = (s,\ 0)^2 = (s^0 s,\ 0) = = (s^0 t,\ 0) = (s,\ 0)\ (t,\ 0) = (t,\ 0)$. Следовательно, s=*t* и φ взаимно однозначнона*K*, а следовательно, и на $F_R\ (X_n)$. Так что в этом случае $F_R\ (X_n) \subseteq S_2$.

Основная теорема о декомпозиции была впервые доказана Кроном и Роудзом.

2.5.4. Иное доказательство основной теоремы декомпозиции

В этом параграфе предлагается иное доказательство основной теоремы декомпозиции для конечных полугрупп. Оно является чисто ал-

гебраическим и не включает методов теории автоматов, рассмотренных в параграфе 2.5.3.

Возможно, методы теории автоматов не очень приятны для изучения, но они развивают интуицию и создают основу для получения дальнейших результатов. Основная теорема декомпозиции сначала была доказана при помощи автоматов и только после того, как справедливость ее стала очевидной, появилось алгебраическое доказательство. В порядке информации мы изложим новое доказательство леммы 3.6, основывающееся на теореме Риса. Повторим утверждение.

4.1. Лемма (3.7). Пусть S — конечная полугруппа. Тогда справедливо одно из следующих утверждений:

а) полугруппа S простая слева;

б) полугруппа S циклическая;

в) существуют собственный левый идеал $V \subset S$ и собственная подполугруппа $T \subset S$, такие, что $S = V \cup T$.

Доказательство. В этом доказательстве мы используем «картину» $\mathcal{Y}$ класса, полученную при помощи теорем Грина и Риса . Пусть *J* обозначает максимальный $\mathcal{Y}$ класс полугруппы *S*. Тогда *J* является либо регулярным, либо одноэлементным нулевым $\mathcal{Y}$ классом. Предположим, что *J* регулярный и содержит только один $\mathcal{L}$ класс. Тогда *J* будет подполугруппой полугруппы S. Пусть *F(J)* — идеал *S — J.*

Если $F(J) = \phi$, то *J—S*—простая слева и справедлив случай а). Если $F_*(J) \neq \phi$, положим $V = F(J)$ и $T = J$, тогда справедлив случай в).

Предположим, что *J* — регулярный класс и имеет более одного $\mathcal{L}$ класса. Пусть *L* — один из них. Если $F(J) = \phi$, положим $V = L$ и $T = J - L = S - L$. Получаем случай в). Если $F(J) \neq \phi$, положим $V = L \cup F(J)$ и $T = (J - L) \cup F(J)$. Снова получаем случай в).

Предположим *J* — одноточечный нулевой $\mathcal{Y}$ класс. Пусть $J = \{q\}$ и *Q* обозначает полугруппу, порожденную элементом *q*. Если Q = *S* — это случай б), если мы положим $V = F(J)$ и $T = Q$, то это случай в). Тем самым все возможности исчерпаны.

4.2. Определение. Пусть S — конечная полугруппа. Тогда системой для S называется упорядоченная последовательность $(S_1, \ldots, S_n)$ подполугрупп из *S*, удовлетворяющая соотношениям:

а) $S_n^l \cdot S_{n-1}^l \ldots S_1^l = S^l$ и

б) для каждой последовательности $(s_n, \ldots, s_1) \in S_n^l \times \ldots \times S_1^l$ и для каждого элемента $s_0 \in S$ существует целое *k*, $1 \leqslant k \leqslant n$, такое, что $s_{k-1} \ldots s_1 s_0 \in S_k$.

4.3. Лемма. Пусть S—конечная полугруппа. Тогда для S найдется система $(S_1, ..., S_n)$, такая, что каждая S_i, $i = 1, ..., n$, является или простой слева, или циклической.

Доказательство. Мы будем вести доказательство индукцией по $|S|$. При $S = \{1\}$ утверждение справедливо. Предположим, что лемма верна для всех полугрупп порядка, меньшего n. Пусть S — полугруппа порядка n. В случае, когда S простая слева или циклическая, имеем тривиальную систему ($S = S_1$). Предположим поэтому, что справедлив случай в) леммы 4.1.

По предположению индукции как V, так и T имеют системы, удовлетворяющие требуемым условиям.

Пусть $(V_1, ..., V_m)$ и $(T_1, ..., T_v)$— соответствующие системы для V и T. Положим теперь $S_1 = T_1, ..., S_p = T_p, S_{p+1} = V_1, ..., S_{p+m} = V_m$. Докажем, что $(S_1, ..., S_{p+m})$ —система, удовлетворяющая сформулированным условиям.

Подполугруппы $(S_1, ..., S_{p+m})$ удовлетворяют соотношению пункта а) из определения системы, так как $S^I = V^I T^I$. Для того чтобы доказать справедливость пункта б), положим $(v_m, ..., v_1, t_p, ..., t_1) \in V_m^I \times ... \times V_1^I \times T_p^I \times ... \times T_1^I$ и пусть $s_0 \in S$. Если $s_0 \in T$, то существует k_1, $1 \leqslant k_1 \leqslant p$, такое, что $t_{k_1-1} ... t_1 s_0 \in T_{k_1}$, поскольку $(T_1, ..., T_v)$ удовлетворяет условиям системы. Если $s_0 \notin T$, то $t_p ... t_1 s_0 \in V$, так как $s_0 \in V$ и V—левый идеал. Тогда существует k_2, $1 \leqslant k_2 \leqslant m$, такое, что $v_{k_2-1} ... v_1 (t_p ... t_1 s_0) \in V_{k_2}$. Из этого следует, что $(S_1, ..., S_{p+m})$ удовлетворяет пункту б).

4.4. Лемма. Пусть $(S_1, ..., S_n)$— нетривиальная система для полугруппы S, Тогда

$$(S^1, S) \mid (S_n^I, \tilde{S}_n^I) \mathrm{w} \cdots \mathrm{w} (S_1^I, \tilde{S}_1^I).$$

Если (X, S) — полугруппа отображений, то $(X, \tilde{S})$ означает, что $\tilde{S}$ есть подполугруппа из $F_R(X)$, состоящая из элементов полугруппы S и постоянных отображений множества X^1.

Доказательство. Пусть $s \in S$. Определим элемент $\hat{s} \in F_R(S_n^I \times \times ... \times S_1^I)$, полагая

$$(s_n, ..., s_1)\,\hat{s} = (s_n, ..., s_{k+1}, s_k, ..., s_1 s, I, ..., I),$$

где k — наименьшее целое число, удовлетворяющее пункту б) в определении систем. Тогда

$$\hat{s} \in (S_n^I, \tilde{S}_n^I) \mathrm{w} \cdots \mathrm{w} (S_1^I, \tilde{S}_1^I).$$

Если мы запишем $(s_n, ..., s_1)\,\hat{s} = [f_n(s_n), ..., f_1(s_1)]$, то для j=1.....n

$$f_j(s_j)=\begin{cases} s_j, & \text{если } s\in S_1, \text{ или } s_1 s\in S_2, \text{ или } \ldots, \text{ или } s_{j-2}\ldots s_1 s\in S_{j-1},\\ s_j\ldots s_1 s, & \text{если ни одно из предыдущих условий неверно и } s_{j-1}\ldots s_1 s\in S_j,\\ I & \text{в противном случае.}\end{cases}$$

Тогда f_j зависит от s и $s_{j-1}, \ldots, s_1$ и принадлежит полугруппе $\widetilde{S}_j^I$.

Отметим, что если k не является наименьшим целым числом, удовлетворяющим пункту б), то элемент $\hat{s}$ не будет принадлежать узловому произведению.

Пусть $\hat{S}$ будет подполугруппа в $(S_n^I,\widetilde{S}_n^I)\,\mathrm{w}\cdots\mathrm{w}\,(S_1^I,\widetilde{S}_1^I)$, порожденная элементами $\{\hat{s}: s\in S\}$.

Пусть отображение $\theta: S_n^I\times\ldots\times S_1^I\to S^1$ определяется соотношениями $\theta(s_n,\ldots,s_1)=s_n\ldots s_1$ и $\theta(I,\ldots,I)=1$. Тогда очевидно, что $\theta(x,\hat{s})=\theta(x)s$ для всех элементов $x\in S_n^I\times\ldots\times S_1^I$ и $s\in S$. Следовательно,

$(S^1, S)\,|\,(S_n^I,\widetilde{S}_n^I)\wr\ldots\wr(S_1^I,\widetilde{S}_1^I)$.

4.5. Лемма. Пусть $\mathscr{S}$ — семейство полугрупп. Предположим, что $(S^1,\widetilde{S})\in\overline{W}(\mathscr{S})$ для всех $S\in\mathscr{S}$. Тогда для всех $(Y,T)\in\overline{W}(\mathscr{S})$ имеет место включение $(Y,\widetilde{T})\in\overline{W}(\mathscr{S})$.

Доказательство. Будем вести доказательство индукцией по n для $\overline{W}_n(\mathscr{S})$. $\overline{W}_1(\mathscr{S})$ состоит из (S^1, S), поэтому по условию $(S^1,\widetilde{S})\in\overline{W}(\mathscr{S})$. Предположим, что лемма верна для $n-1$, пусть $(Y,T)\in\overline{W}_n(\mathscr{S})$. Тогда справедливо одно из соотношений:

1) $(Y,T)\,|\,(X,S)$ для некоторой $(X,S)\in\overline{W}_{n-1}(\mathscr{S})$ или

2) $(Y,T)\,|\,(Y_2,T_2)\wr(Y_1T_1),(Y_i,T_i)\in\overline{W}_{n-1}(\mathscr{S}),\ i=1,2$. Если справедлив случай 1, пусть (X', S')—такая пара, что $(Y,T)\overset{(\theta,\varphi)}{\leftarrow\!\!\!\rightarrow}(X',S')\subseteq\subseteq(X,S)$. Мы покажем, что $(Y,\widetilde{T})\,|\,(X,\widetilde{S})$. Пусть $V=S'\cup$ {постоянные отображения на X'}. V—подполугруппа в $\widetilde{S}$, которая оставляет неподвижным множество X'.

Следовательно, $(X',V)\subseteq(X,\widetilde{S})$. Обозначим {постоянные отображения на X'} через $C_{X'}$, пусть c_x— отображение, переводящее все в x.

Умножение в V имеет следующий вид: $s_1\cdot s_2=s_1s_2$, $c_x\cdot s=c_{xs}$, $c_{x_1}\cdot c_{x_2}=s_1\cdot c_{x_2}=c_{x_2}$. Пусть $\theta: X'\to Y$ и $\varphi: S'\to T$ — эпиморфные отображения, соответствующие делимости $(Y,T)\,|\,(X,S)$. Определим отображение $\psi: V\to\widetilde{T}$, полагая $\psi(s)=\varphi(s)$, $\psi(c_x)=c_{\theta(x)}\in C_Y$. Предположим, что

$s \in S'$ и что s — отображение, постоянное на X'. Пусть оно имеет вид c_x. Тогда $\psi(s) = \varphi(s)$ и $\psi(c_x) = c_{\theta(x)}$. Но $\theta(x_1 s) = \theta(x_1)\ \varphi(s)$ и $x_1 s = x$, поэтому $\theta(x_1 s) = \theta(x)$ для всех $x_1 \in X'$. Следовательно, $\varphi(s) = c_{\theta(x)}$, поэтому ψ определено корректно. Вновь при помощи равенства $\theta(xs) = \theta(x)\ \varphi(s)$ легко показать, что ψ есть эпиморфизм и $(X', V) \to\to (Y\tilde{T})$. Следовательно, $(Y, \tilde{T}) | (X, \tilde{S})$, так что $(Y, \tilde{T}) \in \overline{W}(\mathcal{S})$.

Если же имеет место случай 2, то в силу предыдущего результата $(Y, \tilde{T}) \mid [(Y_2 \times Y_1), (Y_2, T_2)\ \mathrm{w}\ (Y_1, T_1)^{\sim}]$. (Здесь полугруппа записывается с помощью длинного выражения, поэтому символ ~ мы пишем сбоку, т. е. E ~ вместо $\tilde{E}$.)

Теперь постоянные на множестве $Y_2 \times Y_1$ отображения принадлежат полугруппе $(Y_2, \tilde{T}_2)\ \mathrm{w}\ (Y_1, \tilde{T}_1)$, поэтому $[Y_2 \times Y_1, (Y_2, T_2)\ \mathrm{w}\ (Y_1, T_1)^{\sim}] \mid (Y_2, \tilde{T}_2)\ \mathrm{w}\ (Y_1, \tilde{T}_1) \in \overline{W}(\mathcal{S})$.

Следовательно, $(Y, \tilde{T}) \in \overline{W}(\mathcal{S})$.

4.6. Лемма. а) $(U_3, \tilde{U}_3) \in \overline{W}(U_3)$.

б) Пусть G — группа. Тогда $(G, \tilde{G}) \in \overline{W}(G \cup U_3)$.

Доказательство, а) Так как результат умножения полугруппы U_3 справа на элементы r_0 и r_1 в точности такой же, как при действии на U_3 элементов c_{r_0} и c_{r_1}, соответственно, мы видим, что $\tilde{U}_3 = \{c_{r_0}, c_{r_1}, c_1,$ тождественное отображение$\}$, т. е. это — постоянные отображения на U_3 и тождественное отображение. Пусть X — множество и $S \subseteq F_R(X)$ состоит из постоянных на X отображений и тождественного отображения. Мы покажем, что $(X, S) \in \overline{W}(U_3)$. Следовательно, в частности, $(U_3, \tilde{U}_3) \in \overline{W}(U_3)$. Переведем X с помощью взаимно однозначного отображения в декартово произведение $U_1 \times \ldots \times U_1$ с достаточно большим числом сомножителей. Пусть $\overline{X}$ обозначает образ множества X в произведении $U_1 \times \ldots \times U_1$, а $x \in X$. Если $\bar{x} = (r_{i_1}, \ldots, r_{i_n})$, $i_j \in \{0, 1\}$, то определим отображение $\varphi : S \to U_3 \times \ldots \times U_3$, полагая $\varphi(c_x) = (r_{i_1}, \ldots, r_{i_n})$ и $\varphi(Id) = (1, \ldots, 1)$. Отображение φ является мономорфизмом. Пусть $\tilde{S} = \varphi(S)$. Легко видеть, что пара $(\overline{X}, \overline{S})$ есть полугруппа отображений, изоморфная (X, S), и что $(\overline{X}, \overline{S}) \subseteq (U_3, U_3) \times \ldots \times (U_3, U_3) \in \overline{W}(U_3)$. Следовательно, $(X, S) \in \overline{W}(U_3)$.

б) Пусть G— группа. Тогда $\tilde{G}$ будет дизъюнктивным объединением G и C_G с умножением, определенным в доказательстве леммы 4.5. Если $g \in G$, определим $\hat{g} \in F_R(G \times G)$ при помощи соотношения $(g_2, g_1)\,\hat{g} = (g_2, g_1 g)$. Если $c_g \in C_G$, определим $\hat{c}_g \in F_R(G \times G)$, полагая $(g_2, g_1)\,\hat{c}_g = (g_1^{-1}, g_1 g)$. Отметим, что элементы $\hat{g}$ и $\hat{c}_g$ принадлежат произведению $(G, \hat{S},)$ w (G, G), где S = {постоянные отображения на G $\bigcup$ единичное отображение}. Обозначим символом $\hat{G}$ подполугруппу в произведении (G, S) w (G, G), порождаемую элементами $\{\hat{g}, \hat{c}_g: g, c_g \in \tilde{G}\}$. Предположим, что отображение $\theta : G \times G \to\!\!\!\!\to G$ определяется соотношением $\theta(g_2, g_1) = g_2 g_1$. Тогда очевидно, что $\theta(x\hat{s}) = \theta(x)\,s$ для всех элементов $x \in G \times G$ и всех $s \in \tilde{G}$. Следовательно, $(G, \tilde{G}) \mid (G, S) \wr (G, G)$. Но согласно пункту а) имеем $(G, S) \in \overline{W}(U_3)$. Поэтому $(G, S) \wr (G, G)$ и, следовательно, $(G, \tilde{G})$ принадлежит $\overline{W}(G \cup U_3)$.

4.7.Лемма. Пусть— $\mathcal{S}$ семейство моноидов. Тогда если $S \in \in W(\mathcal{S})$, то справедливо включение $S^I \in W(\mathcal{S} \cup \{U_3\})$.

Доказательство. Пусть $S \in W_1(\mathcal{S}) = \mathcal{S}$. S— моноид. Определим мономорфизм $\varphi : S^I \to S \times U_3$, полагая $\varphi(s) = (s, r_0)$ и $\varphi(I) = (1,1)$. Тогда, $S^I \mid S \times U_3 \mid S$ w $U_3 \in W(\mathcal{S} \cup \{U_3\})$. Предположим, утверждение справедливо для $W_{n-1}(\mathcal{S})$ и пусть $S \in \in W_n(\mathcal{S})$. Тогда или 1) $S \mid T \in W_{n-1}(\mathcal{S})$, откуда следует, что $S^I \mid T^I$, или 2) $S \mid T_2$ w T_1, $T_i \in W_{n-1}(\mathcal{S})$, $i = 1,2$. Тогда $S^I \mid (T_2$ w $T_1)^I \mid T_2^I$ w $T_1^I \in W(\mathcal{S} \cup \{U_3\})$.

4.8. Лемма. Предположим, что для полугруппы S справедливо соотношение (3.2), т. е.

$$S \in W[\mathrm{PRIMES}(S) \cup \{U_3\}].$$

Тогда

$$(S^I, \tilde{S}^I) \in \overline{W}[\mathrm{PRIMES}(S) \cup \{U_3\}].$$

Доказательство. Так как PRIMES (S) и U_3 являются моноидами, то лемма 4.7 утверждает, что $S^I \in W[\mathrm{PRIMES}(S) \cup \{U_3\}]$. Тогда в силу утверждения 2.21

$$(S^I, S^I) \in \overline{W}[\mathrm{PRIMES}(S) \cup \{U_3\}].$$

Но согласно лемме 4.6. $\overline{W}[\mathrm{PRIMES}(S) \cup \{U_3\}]$ удовлетворяет условиям леммы 4.5, поэтому

$$(S^I, \tilde{S}^I) \in \overline{W}[\mathrm{PRIMES}(S) \cup \{U_3\}].$$

Для удобства восстановим доказанные с помощью алгебраических методов результаты параграфа 2.5.3.

4.9. Лемма. а) $\mathrm{PRIMES} \cup \mathrm{UNITS} \subseteq \mathrm{IRR}$ (определения 3.1. и 3.2),

б) Если S — циклическая полугруппа, то S удовлетворяет соотношению (3.2).

в) Если *G* — группа, то *G* удовлетворяет соотношению (3.2).

Доказательство, а) См. лемму 3.6.

б) См. лемму 3.9.

в) См. доказательство леммы 3.8.

4.10. Лемма. Если S — простая слева полугруппа, то S удовлетворяет соотношению (3.2).

Доказательство. В силу предположения леммы $S \cong G \times A^l$, где G — группа и *A* — непустое множество.

Тогда PRIMES *(S)* = PRIMES (G) (см. доказательство леммы 3.8). Тогда в соответствии с пунктом в) леммы 4.9 $S \in W\,[\mathrm{PRIMES}\,(S) \cup \cup\ \{U_3,\ A^l\}]$. Остается только доказать, что $A^l \in W\ (U_3)$. Для автоматов это сделано в доказательстве леммы 3.8. Тогда лемма будет доказана.

Пусть $L = \{a,\ b\}^l$. В этом случае $L \subseteq U_3$ w $U_1 \cong F\ (U_1,\ U_3) \times \times\ {}_Y U_1$ и подполугруппа $\{[(r_0,\ 1),\ r_1],\ [(r_1,\ 1),\ r_1]\}$ изоморфна *L* [здесь элемент $(r_0,\ 1)$ следует рассматривать как функцию $f \in F\ (U_1,\ U_2)$, определяемую соотношениями $f(r_0) = r_0$ и $f(r_1) = 1$]. Поскольку A^l— подполугруппа в прямом произведении подходящего числа полугрупп *L*, имеем $A^l \in W\ (U_3)$.

4.11. Теорема. Пусть S — конечная полугруппа.

а) $S \in W\ (\mathscr{S} \cup \{U_3\})$ тогда и только тогда, когда $\mathrm{PRIMES}\ (S) \subseteq \subseteq \mathrm{PRIMES}\ (\mathscr{S})$. Напомним, в частности, соотношение (3.2):

$$S \in W\,(\mathrm{PRIMES}\,(S) \cup \{U_3\}). \quad (4.1)$$

б) $\mathrm{PRIMES} \cup \mathrm{UNITS} = \mathrm{IRR}$.

в) В общем случае $S \notin W\ [\mathrm{IRR}\ (S)]$.

Доказательство. Докажем прежде всего справедливость соотношения (4.1). Для случаев, когда полугруппа S циклическая или простая слева, это уже сделано. В общем случае *S* имеет нетривиальную систему $(S_1, \ldots, S_n)$, где каждая полугруппа простая слева или циклическая. На основании леммы 4.4 и леммы 4.8 получим

$$S \in W\,(\mathrm{PRIMES}\,\{S_1, \ldots, S_n\} \cup \{U_3\}).$$

Но очевидно, что $\mathrm{PRIMES}\ \{S_1, \ldots, S_n\} \subseteq \mathrm{PRIMES}\ (S)$, откуда и следует соотношение (4.1).

Из соотношения (4.1) вытекает, что если $\mathrm{PRIMES}\ (S) \subseteq \subseteq \mathrm{PRIMES}\ (\mathscr{S})$, то $S \in W\ (\mathscr{S} \cup \{U_3\})$. Для доказательства обрат-

ного утверждения предположим, что $S \in W(\mathcal{S} \cup \{U_3\})$ и пусть $P \in \in \mathrm{PRIMES}(S)$. Тогда $P \in W(\mathcal{S} \cup \{U_3\})$ и в силу утверждения 3.16 $P|S$ для некоторой полугруппы $S \in \mathcal{S}$, так как группа не может делить U_3. Следовательно, $P \in \mathrm{PRIMES}(\mathcal{S})$. Тем самым доказан пункт а). Для доказательства пунктов б) и в) отсылаем читателя к теореме 3.3.

4.12. Замечание. Пункт в) доказанной теоремы объясняет разницу между PRIMES и UNITS и включение полугруппы U_3 в ранее приведенные соотношения.

4.13.Следствие. Пусть S— конечная полугруппа. Тогда $S \mid S_n \mathrm{w} \ldots \mathrm{w} S_1$ для некоторой последовательности $S_1, \ldots, S_n$, где

$$S_i \in \mathrm{PRIMES}(S) \cup \mathrm{UNITS}, \quad i = 1, \ldots, n.$$

Доказательство. См. следствие 3.5.

4.14. Замечание. Если полугруппа S — объединение групп, то каждый $\mathcal{J}$- класс будет простой группой.

Последовательность $(J_1, \ldots, J_n)$ всех $\mathcal{J}$ классов полугруппы S, для которых условие $i < j$ влечет $J \leq_1 J_j$, является системой для полугруппы S. Она играет большую роль в работе со сложностью полугрупп, представляющих собой объединение групп (определение сложности дано дальше).

Согласно условию упорядоченности J_1 олжен быть максимальным $\mathcal{J}$ классом полугруппы S и J_n — ядром полугруппы S. Доказательство того, что $(J_1, \ldots, J_n)$ есть система, ведется индукцией по числу $\mathcal{J}$ классов полугруппы S. Предположим, что это верно для полугрупп с $(n — 1)$ $\mathcal{J}$ классами и пусть S имеет n $\mathcal{J}$ классов. Выберем максимальный $\mathcal{J}$ класс и обозначим его J_1. Множество S — J_1 есть идеал полугруппы S, являющейся объединением групп с $(n — 1)$ $\mathcal{J}$ классами (они представляют собой оставшиеся $\mathcal{J}$ классы полугруппы S). Продолжая так же, как в доказательстве леммы 4.3, видим, что $(J_1, \ldots, J_n)$ — система для полугруппы S.

Доказательство теоремы, приведенное в этом параграфе, принадлежит Крону и Роудзу.

2.5.5. Приложение основной теоремы декомпозиции

Сначала дадим определение и установим некоторые свойства комбинаторных полугрупп.

5.1. Определение. Полугруппу S будем называть *комбинаторной* тогда и только тогда, когда каждая подгруппа полугруппы S имеет порядок 1.

5.2. Утверждение. Гомоморфные образы, подполугруппы и конечные прямые произведения комбинаторных полугрупп будут комбинаторными.

5.3. Утверждение. а) Существует целое положительное число $p = p(S)$, такое, что $s^p = s^{p+1}$ для всех элементов $s \in S$ тогда и только тогда, когда полугруппа S комбинаторная.

б) PRIMES (S) = $\varnothing$ тогда и только тогда, когда S — комбинаторная полугруппа.

в) Класс комбинаторных полугрупп замкнут относительно операций полупрямого и узлового произведений, т. е. если полугруппы S_1 и S_2 комбинаторные, то и полугруппы $S_2 \text{ w } S_1$ и $S_2 \times {}_Y S_1$ комбинаторные. Более того, если полугруппа $S_2 \text{ w}_Y S_1$ комбинаторная, то полугруппы S_1 и S_2 также комбинаторные.

Доказательство. Оставляем читателю доказать пункты а) и б) в качестве упражнений. Для доказательства пункта в) отметим, что если $P \in$ PRIMES $(S_2 \text{w } S_1)$ или PRIMES $(S_2 \times {}_Y S_1)$, то $P \in$ PRIMES (S_1) или PRIMES (S_2). Если последнее множество пустое, то справедливо предыдущее включение. Это доказывает первое утверждение пункта в). Так как $S_i \mid S_2 \text{ w } S_1$, $i = 1,2$, имеем PRIMES $(S_i) \subseteq$ PRIMES $(S_2 \text{ w } S_1)$.

Следовательно, когда полугруппа $S_2 \text{w } S_1$ комбинаторная, полугруппы S_1 и S_2 такие же.

5.4. Принцип индукции для комбинаторных полугрупп. Пусть $\mathcal{P}$ — произвольное свойство конечных полугрупп, но такое, что

1) $\mathcal{P}$ замкнуто относительно делимости;

2) $U_3^{(n)} \equiv U_3 \text{w} \ldots \text{w } U_3$ (n раз) удовлетворяет свойству $\mathcal{P}$.

Тогда каждая комбинаторная полугруппа удовлетворяет свойству $\mathcal{P}$.

Доказательство. Пусть $\mathcal{P}$ — свойство конечных полугрупп, удовлетворяющее условиям 1 и 2, и пусть S — комбинаторная полугруппа. Тогда PRIMES $(S) = \varphi$, поэтому $S \in W(U_3)$ и $S \mid U_3^{(n)}$ для некоторого n в соответствии с основной теоремой декомпозиции. Следовательно, полугруппа S удовлетворяет свойству $\mathcal{P}$.

5.5. Обозначение, а) Пусть A — непустое множество, а символ $\Pi(A)$ обозначает множество всех бесконечных последовательностей элементов из A.

б) Пусть S — полугруппа. Определим n-ю *итерацию автомата* S^f как $(S^{f\sigma})^n : \Sigma S \to \Sigma S$, где $(S^{f\sigma})^2 = S^{f\sigma} S^{f\sigma}$ и $(S^{f\sigma})^n = S^{f\sigma} (S^{f\sigma})^{n-1}$.

в) Пусть S — полугруппа и пусть $X \in \Pi(S^1)$, $X = (x_1, x_2, \ldots)$.

Тогда $\tilde{P}_X : \Sigma S \to \Sigma S$ определяется соотношением $P_X(s_1, \ldots, s_n) = (x_1 s_1, \ldots, x_n s_n)$.

Далее мы сформулируем основное предложение этого параграфа, в котором утверждается, что если итерировать автомат комбинаторной полугруппы достаточно долго, то автомат *(n* + 1)-й итерации совпадет с автоматом *n*-й итерации. Кроме того, выход достаточно большой итерации автомата комбинаторной полугруппы соответствует умножению слева на вход и совпадает с выходом последующей итерации.

5.6. Предложение. Следующие утверждения эквивалентны:

а) S — комбинаторная полугруппа;

б) существует положительное целое число *m = m (S),* такое, что

$$(S^{f\sigma})^m = (S^{f\sigma})^{m+1}. \quad (5.1)$$

в) существует положительное целое число $q = q\,(S)$, такое,что для всех $X \in \Pi\,(S^1)$,

$$(P_X S^{f\sigma})^q = (P_X S^{f\sigma})^{q+1}, \quad (5.2) \qquad (S^{f\sigma} P_X)^q = (S^{f\sigma} P_X)^{q+1}. \quad (5.3)$$

Доказательство. Из пункта в) следует пункт б), если выбрать *X* = (1,1…..).

Покажем, что из пункта б) следует пункт а). Предположим, что *S* — полугруппа с нетривиальной подгруппой *G* (т. е. S не является комбинаторной). Пусть $g \in G,\ g \neq 1$, где 1 — единица группы *G*. Тогда $(S^{f\sigma})^n\ (g,\ 1) = (g,\ g^n)$ для всех *n* = 1, 2, ... Так как $g \neq \phi$ 1, то не существует положительного целого *m,* такого, что

$$(S^{f\sigma})^m = (S^{f\sigma})^{m+1}.$$

Покажем, что из пункта а) следует пункт в). Во-первых, заметим, что соотношения (5.2) и (5.3) эквивалентны. Предположим, что (5.2) справедливо. Тогда

$$(S^{f\sigma} P_X)^{q+1} = S^{f\sigma} (P_X S^{f\sigma})^q P_X = S^{f\sigma} (P_X S^{f\sigma})^{q+1} P_X = (S^{f\sigma} P_X)^{q+2}.$$

Аналогично из (5.3) вытекает соотношение (5.2). Пусть $\mathcal{P}$ — совокупность всех полугрупп, удовлетворяющих соотношению (5.2). Мы покажем, что $\mathcal{P}$ замкнута относительно делимости и $U_3^{(n)} \in \mathcal{P}$ при любых *n.* Тогда для доказательства предложения можно воспользоваться принципом индукции для комбинаторных полугрупп.

1) $\mathcal{P}$ *замкнута относительно делимости.* Пусть $S \mid T$ и $T \in \mathcal{P}$. Имеем $S \xleftarrow{\Phi} T' \subseteq T$.

Тогда $S^f \mid T^f$ и $S^{f\sigma} = h_2^{\Gamma} T^{f\sigma}\ h_1^{\Gamma}$, где $h_1 : S \to T'$, $h_1(s) = \bar{s}$, символ $\bar{s}$ обозначает фиксированный представитель множества

$\varphi^{-1}(s)$ и $h_2 : T' \to\!\!\!\!\to S$, $h_2(t) = \varphi(t)$. (Относительно этого обозначения см. доказательство пункта б) предложения 2.6.) Пусть $X \in \Pi(S^1)$, $X = (x_1, x_2, \ldots)$ и $q = q(T)$.

Тогда

$$(P_X S^{f\sigma})^{q+1} = (P_X h_2^\Gamma T^{f\sigma} h_1^\Gamma)^{q+1} = P_X h_2^\Gamma (T^{f\sigma} h_1^\Gamma P_X h_2^\Gamma)^q T^{f\sigma} h_1^\Gamma. \quad (5.4)$$

Мы утверждаем теперь, что

$$h_2^\Gamma (T^{f\sigma} h_1^\Gamma P_X h_2^\Gamma)^p = h_2^\Gamma (T^{f\sigma} P_Y)^p \text{ для всех } p = 1, 2, \ldots, \quad (5.5)$$

где $Y \in \Pi(T^1)$, $Y = (\bar{x}_1, \bar{x}_2, \ldots)$. Для $p = 1$ соотношение (5.5) легко проверить. Предположим, что это соотношение справедливо для p — 1. Пусть

$$(t_1, \ldots, t_n) = h_2^\Gamma (T^{f\sigma} h_1^\Gamma P_X h_2^\Gamma)^{p-1} (t'_1, \ldots, t'_n) = h_2^\Gamma (T^{f\sigma} P_Y)^{p-1} (t'_1, \ldots, t'_n).$$

Тогда для доказательства формулы (5.5) достаточно проверить, что

$$h_2^\Gamma T^{f\sigma} h_1^\Gamma P_X (t_1, \ldots, t_n) = h_2^\Gamma T^{f\sigma} P_Y (u_1, \ldots, u_n),$$

где u_i — произвольный элемент множества $\varphi^{-1}(t_i)$, $i = 1, \ldots, n$. Справедливость последнего соотношения легко устанавливается. Но так как $(T^{f\sigma} P_Y)^q = (T^{f\sigma} P_Y)^{q+1}$, из равенств (5.4) и (5.5) вытекает, что $(P_X S^{f\sigma})^{q+1} = (P_X S^{f\sigma})^{q+2}$. Следовательно, совокупность $\mathfrak{P}$ замкнута относительно делимости.

2) $U_3^{(n)} \in \mathfrak{P}$. Проверим, что $(P_X U_3^{f\sigma})^2 = (P_X U_3^{f\sigma})^3$ для всех $X \in \Pi(U_3)$. Заодно проверим, что совокупность $\mathfrak{P}$ замкнута относительно операции прямого произведения (при конечном числе сомножителей). Доказательство обоих утверждений тривиально. Более того, если S_1 и S_2 — комбинаторные полугруппы, то $q(S_1 \times S_2) = \max\{q(S_1), q(S_2)\}$.

Предположим теперь, что существует q_n, такое, что соотношение (5.2) справедливо для $U^{(n)}_3$. Мы найдем q_{n+1} при помощи q_n, так что (5.2) справедливо для $U^{(n+1)}_3 = F(U_3, U_3^{(n)}) \times_Y U_3$. Отметим, что q_n такое целое число, что (5.2) справедливо для $F(U_3, U_3^{(n)})$. Пусть $X \in \Pi(U_3^{(n+1)})$ и

$X = [(g_1, b_1), (g_2, b_2), \ldots], b_i \in U_3, g_i \in F(U_3, U_3^{(n)})$.

Пусть $M = P_X U_3^{(n+1)f\sigma}$ и $Y \in \Sigma U_3^{(n+1)}$, где $Y = [(f_1, a_1), \ldots, (f_k, a_k)]$.

Мы хотим получить ситуацию, когда можно применить предположения индукции. Это можно будет сделать, если мы сумеем получить полупрямое произведение, действующее как прямое произведение. Если все a_i и b_i равны 1, это и есть искомая ситуация. Пусть r— наибольшее целое число, такое, что все $a_1, \ldots, a_{r-1}, b_1, \ldots, b_{r-1}$ равны 1. Тогда на первые (r — 1) членов множества $Y M^p$ действует в точности так же, как $[P_X F(U_3, U_3^{(n)})^f \times U_3^f]^{\sigma p}$, где

$X' \in \Pi F\ (U_3,\ U_3^{(n)})$ с $X' = (g_1, g_2, \ldots)$. Поэтому после q_n итерации автомата M первые $(r-1)$ членов из Y не изменяются. Запишем $M^{q_n}(Y) = [(h_1, 1), \ldots, (h_{r-1}, 1), (h_r', c_r), \ldots, (h_k', c_k)]$. Заметим, что каждый $c_i \in \{r_0, r_1\}$ после второй итерации и, конечно, $q_n \geqslant 2$. Следовательно, $xc_i = c_i$ для всех $x \in U_3$.

Теперь после $q_n + 1$ итераций r член остается неизменным. Для того чтобы убедиться в этом, начнем с $M^{q_n}(Y)$ и проделаем итерацию p раз. Новый r член имеет вид

$$\{g_r[^{b_r}(h_1 \ldots h_{r-1}\ g_r)]^{p-1} \cdot {}^{b_r}(h_1 \ldots h_r'),\ c_r\}.$$

Теперь для всех $f \in F\ (U_3,\ U_3^{(n)})$ имеем $f^{q_n} = f^{q_n+1}$. {Рассмотрим $[F(U_3, U_3^{(n)})^{\sigma}]^{q_n}(f, f)$.} Следовательно, при $p = q_n + 1$ член не меняется. Запишем

$$M^{2q_n+1}(Y) = [(h_1, 1), \ldots, (h_r, c_r), \ldots, (h_k, c_k)].$$

Теперь покажем, что в результате дальнейших $q_n + 1$ итераций все члены остаются неизменными, т. е. $q_{n+1} \leqslant 3q_n + 2$. Тем самым будет доказано утверждение. Пусть c_1 обозначает единицу полугруппы $F\ (U_3,\ U_3^{(n)})$. Определим автомат $f : \Sigma U_3^{(n+1)} \to U_3^{(n+1)}$, полагая

$$f[(k_1, d_1), \ldots, (k_n, d_n)] = \begin{cases} ({}^{d_{n-1}}k_n, d_n), \text{ если } n \geqslant 2,\ d_{n-1} \neq 1, \\ (c_1, d_n) \text{ в противном случае.} \end{cases}$$

Определим также $X_1 \in \Pi F\ (U_3,\ U_3^{(n)})$, полагая

$$X_1 = [j_1, \ldots, j_r, {}^{c_r}g_{r+1}{}^{c_r b_{r+1}}(h_1 \ldots h_r), \ldots, {}^{c_{k-1}}g_k^{c_{k-1}b_k}(h_1 \ldots h_r), j_{k+1}, \ldots],$$

где каждое $j_i = c_1$. Определим $X_2 \in \Pi(U_3^{(n+1)})$, полагая

$$X_2 = [(h_1, 1), \ldots, (h_r, 1), ({}^{b_{r+1}}g_{r+1}(h_1 \ldots h_r), 1), \ldots, (g_k^{b_k}(h_1 \ldots h_r), 1), (j_{k+1}, 1), \ldots],$$

где, как и раньше, каждое $j_i = c_1$. Отметим, что X_1 и X_2 являются функциями X и Y. Пусть $F = [F(U_3, U_3^{(n)})]^f$. Проверим с помощью непосредственного вычисления, что

$$M^{2q_n+2+p}(Y) = P_{X_2} U_3^{(n+1)\sigma}[P_{X_1} F \times U_3^f]^{\sigma p} f^{\sigma} M^{2q_n+1}(Y)$$

для $p = 1,2,\ldots$ По индукции $[P_{X_1} F \times U_3^f]^{\sigma q_n} = [P_{X_1} F \times U_3^f]^{\sigma(q_n+1)}$ для всех $X_1 \in \Pi F(U_3, U_3^{(n)})$. Следовательно, $M^{3q_n+2}(Y) = M^{3q_n+3}(Y)$ для всех $Y \in \Sigma U_3^{(n+1)}$.

5.7. Замечание. Способ наглядного представления $(S^{1f\sigma})^p$ дается «прямоугольником Паскаля» таблицы умножения моноида S^1. Пусть $X = (x_1, x_2, \ldots)$ и $Y = (s_1, s_2, \ldots) \in \Pi(S^1)$. Рассмотрим рис. 1.

	$s_1 = a_{01}$	$s_2 = a_{02}$	$s_3 = a_{03}$	...
$x_1 = a_{10}$	$x_1\ s_1 = a_{11}$	$x_1\ s_1\ s_2 = a_{12}$	$x_1\ s_1\ s_2\ s_3 = a_{13}$	...
$x_2 = a_{20}$	$x_2\ x_1\ s_1 = a_{21}$	$x_2 x_1 s_1 x_1 s_1 s_2 = a_{22}$	$a_{22} \cdot a_{13} = a_{23}$	...
$x_3 = a_{30}$	$x_3\ x_2\ x_1\ s_1 = a_{31}$	$a_{31} \cdot a_{22} = a_{32}$	$a_{32} \cdot a_{23} = a_{33}$	...
⋮	⋮	⋮	⋮	⋮

Рис. 1. Прямоугольник Паскаля для S^1

Пусть $a_{0i} = s_i$, $a_{j0} = x_j$. Вычислим элементы таблицы по формуле $a_{mn} = a_{m(n-1)} \cdot a_{(m-1)n}$. Если $x_1 = x_2 = ... = 1$, то $(a_{01}, a_{02}, ...)$ будет выходом для $(S^1|\sigma)^n\ (s_1, s_2, ...)$.

Следовательно, пункт б) предложения 5.6 утверждает, что полугруппа S будет комбинаторной тогда и только тогда, когда существует $m = m(S^1)$, такое, что для всех $Y \in \Pi(S^1)$ m, $m+1$, ... строки таблицы одинаковы.

Пусть $f_i = S^1|_i L_{x_i}$, где $L_{x_i}(s_1, ..., s_n) = (x_i, s_1, ..., s_n)$. Тогда $a_{jk} = f_j f_{j-1}^{\sigma} ... f_1^{\sigma}(s_1, ..., s_k)$.

Здесь имеется следующая «двойственность». Транспозиция в таблице (см. рис. 1) относительно диагонали, т. е. преобразование, переводящее элемент a_{jk} в a_{kj}, может быть выполнена, если заменить X на Y, а S^1 — на $r(S^1)$. Результат, двойственный пункту б) предложения 5.6, формулируется следующим образом.

5.8. Предложение. Полугруппа S будет комбинаторной тогда и только тогда, когда существует целое число $n = n(S)$, такое, что если $b_{jk} = f_j f_{j-1}^{\sigma} \cdots f_1(1, ..., 1)$ (k раз), то $b_{jn} = b_{j(n+1)} = ...$ для всех j=1,2,... и для всех $(x_1, x_2, ...) \in \Pi(S^1)$.

Доказательство. Пусть $\overline{a_{kj}} = f'_k f'^{\sigma}_{k-1} ... f'^{\sigma}_1 (x_1, ..., x_j)$, где $f'_i = r(S^1)\ |L_{Si}$ представляет (k, j) элемент прямоугольника Паскаля для $r(S^1)$. Тогда поскольку полугруппа $r(S^1)$ комбинаторная, если $s_1 = s_2 = ... = 1$, то существует целое n, такое, что $\overline{a_{nj}} = \overline{a_{(n+1)j}} = ...$ для всех $j = 1, 2, ...$ и для всех $(x_1, x_2, ...)$.

Но согласно предыдущему замечанию $\overline{a_{kj}} = a_{jk}$. Записывая $b_{jk} = a_{jk}$, когда $s_1 = s_2 = ... = 1$, получаем требуемое утверждение.

2.6. Сложность и групповая сложность конечных автоматов

2.6.1. Определения групповой сложности

В предыдущих главах мы подошли к изучению конечных автоматов (или конечных последовательностных машин) тремя различными путями, рассматривая:
1) автоматы как пятерки (X, Y, Q, δ, β), включающие входные и выходные сигналы, а также состояния;
2) вместо автоматов абстрактные конечные полугруппы S;
3) автоматы как полугруппы преобразований (Q, S), где множество преобразований S пространства состояний Q определяет переходы состояний из Q, индуцированные входными последовательностями.
Для каждого случая была получена соответствующая форма основной теоремы декомпозиции.
1. Говорят, что автомат M — *групповой* или *перестановочный* автомат, если каждый входной сигнал его индуцирует перестановку элементов в его множестве состояний; в этом случае полугруппа M^s автомата M будет группой. Мы называем автомат M *основным комбинаторным* или *тождественно-возвратным* автоматом, если каждый входной сигнал автомата M индуцирует возвратное (т. е. постоянное) отображение или тождественное отображение его множества состояний. Мы называем автомат *комбинаторным* или *свободным от групповых компонент,* если автомат M моделируется каскадным соединением тождественно-возвратных автоматов. Эквивалентным будет следующее утверждение: полугруппа M^s является комбинаторной, т. е. она не содержит подгрупп, порядок которых больше 1.
Тогда теорема о декомпозиции утверждает, что приведенный конечный автомат M имеет GC декомпозицию X, т. е. автомат M моделируется каскадным соединением X автоматов $M_1, ..., M_n$, причем каждая компонента M_j будет конечным приведенным групповым или конечным приведенным комбинаторным автоматом, и в общем случае входной сигнал, поступающий в автомат M_j, в момент времени t определяется входным сигналом, поступающим в систему в момент времени t, а также выходными сигналами, выдаваемыми автоматами $M_1 .... M_{j-1}$, в момент времени t , кроме того, в соединении нет обратных связей.
2. Пусть задана конечная полугруппа S, тогда существуют полугруппы

$S_1, \ldots, S_n \in \mathrm{PRIMES}(S) \cup \{U_3\}$,

такие, что

$$(S^1, S) \mid (S_n, S_n) \wr \cdots \wr (S_1, S_1). \quad (1.1)$$

Если пары (X_1, S_1) и (X_2, S_2) представляют собой группы преобразований (т. е. S_i — группа и единица группы S_i действует как тождественное преобразование на множество X_i), то $(X_2, S_2) \wr (X_1, S_1)$ будет группой преобразований. Кроме того, если (X_1, S_1) и (X_2, S_2) — комбинаторные полугруппы преобразований (т. е. S_1 и S_2 — комбинаторные полугруппы преобразований), то $(X_2, S_2) \wr (X_1, S_1)$ будет комбинаторной полугруппой. Так как элементы из PRIMES — группы, то (G, G) является группой преобразований. Если G—группа и U_3 — комбинаторная полугруппа, то мы можем собрать вместе соседние однотипные элементы в формуле (1.1) и получить (S^1, S), делящую узловое произведение полугрупп преобразований так, чтобы в нем группы преобразований чередовались с комбинаторными полугруппами преобразований.

3. Пусть (X, S) — полугруппа преобразований. Тогда поскольку S есть подполугруппа из $F_R(X)$—полугруппы всех функций на множестве X, то легко показать, что

$$(X, S) \mid (X, F_R(X)) \mid (F_R(X), F_R(X)).$$

Поэтому в соответствии со случаем 2 существуют полугруппы $S_1, \ldots, S_m \in \mathrm{PRIMES} \cup \{U_3\}$, такие, что

$$(X, S) \mid (S_m, S_m) \wr \cdots \wr (S_1, S_1). \quad (1.2)$$

[Однако в этом случае мы не показали, что простые группы обязательно принадлежат PRIMES (S). Позднее будет проиллюстрировано, что декомпозиция для (X, S) может быть выбрана так, что $S_i \in \in \mathrm{PRIMES}(S) \cup \{U_3\}$].

Тогда, как и в случае 2, можно собрать соседние члены для получения (X, S), делящего узловое произведение, в котором группы преобразований чередуются с комбинаторными полугруппами преобразований.

Возникает вопрос: каково наименьшее число групп, требуемое для разложения объекта предыдущего вида в групповые и комбинаторные объекты? Оказывается, это число не зависит от того, какое из трех предыдущих описаний автомата выбрано. Оно зависит только от полугруппы S . Назовем это число *групповой сложностью* полугруппы S (обозначим как $\ddagger_G(S)$) и посвятим настоящую главу исследованию основных свойств сложности.

1.1. Определение. Пусть *X* есть GC-дскомпозиция конечного приведенного автомата *M* в каскадное соединение групповых и комбинаторных автоматов $M_1, \ldots, M_n$, упорядоченных некоторым образом. Пусть $\#_G(X)$ обозначает число групповых автоматов среди $M_1,\ldots,M_n$. Тогда *групповой сложностью* автомата *M* [определение (а)] является $\#_G^a(M) = \min\{\#_G(X) : X$ есть GC-декомпозиция $M\}$. Для полугруппы S с автоматом S_M мы положим

$$\#_G^a(S) = \#_G^a(S^M).$$

1.2. Определение. Пусть S — конечная полугруппа. Тогда *групповой сложностью* полугруппы S [определение (б)] $\#_G^b(S)$ будет наименьшее неотрицательное целое число *n,* такое, что

$$(S^1, S) \mid (Y_n, C_n) \wr (X_n, G_n) \wr (Y_{n-1}, C_{n-1}) \wr \cdots$$

$$\cdots \wr (Y_1, C_1) \wr (X_1, G_1) \wr (Y_0, C_0),$$

где (X_i, G_i), $i = 1, \ldots, n$ — нетривиальные группы преобразований и $C_1, \ldots, C_{n-1}$ — нетривиальные комбинаторные полугруппы (C_0 и C_n могут быть тривиальными).

1.3. Теорема. $\#_G^a(M) = \#_G^b(M^S)$, поэтому, в частности, определения а) и б) для $\#_G(S)$ эквивалентны, и два автомата с одной и той же полугруппой имеют равные групповые сложности.

Доказательство. Теорема следует из материала гл. 2.3 (теорема 2.6), в которой показано, что S^M можно моделировать каскадным соединением $M_1^{S^M}$ и $M_2^{S^M}$, если полугруппа S делит узловое произведение

$$M_1^S \text{ и } M_2^S.$$

1.4. Определение. Пусть *(X, S)* — полугруппа преобразований. Тогда *групповой сложностью* [определение в)] $\#_G^c(X, S)$ для *(X, S)* будет наименьшее неотрицательное целое число *n,* такое, что

$$(X, S) \mid (Y_n, C_n) \wr (X_n, G_n) \wr (Y_{n-1}, C_{n-1}) \wr \cdots$$

$$\cdots \wr (Y_1, C_1) \wr (X_1, G_1) \wr (Y_0, C_0), \qquad \textbf{(1.3)}$$

где (X_i, G_i), $i = 1, \ldots, n$ — нетривиальные группы преобразований и $C_1, \ldots, C_{n-1}$ — нетривиальные комбинаторные полугруппы (C_0 и C_n могут быть тривиальными).

(Можно убедиться, что соотношения такого вида тесно связаны с конечной физикой. Действие компонент можно рассматривать как аналог «собственных векторов», так что эвристически, $\#_G^c(S)$ является наименьшим числом «групповых или перестановочных собственных

векторов» при приведении S к треугольному виду (из перестановок или постоянных компонент).

Заметим, что если $(X, S) | (Y, T)$, то $\#_G^c(X, S) \leqslant \#_G^c(Y, T)$.

1.5. Теорема. а) Если пара *(X, S)* — полугруппа преобразований, то

$$(X, S) \in \overline{W}[\mathrm{PRIMES}(S) \cup \{U_3\}]$$

б) $\#_G^c(X, S) = \#_G^b(S)$ для всех множеств *X*, таких, что пара *(X, S)* — полугруппа преобразований (т. е. таких, что S действует точно на *X)*.

Доказательство. Рассмотрим сначала пару *(X,* {0}), где {0} действует тождественно на *X*. Пусть S — нетривиальная полугруппа с единичным элементом 1. Представим *X* как подмножество прямого произведения подходящего числа копий *S*. Тогда легко показать, что

$$(X, \{0\}) | [(S \times \ldots \times S), \{1\}] | (S, S) \times \ldots \times (S, S).$$

Так как замыкание узловых произведений замкнуто относительно делимости и прямого произведения, полагая S = U_3, получим

$$(X, \{0\}) \in \overline{W}[\mathrm{PRIMES}(\{0\}) \cup \{U_3\}] = \overline{W}(U_3).$$

Следовательно, $\#_G^c(X, \{0\}) = 0$.

Пусть теперь имеется произвольная полугруппа преобразований *(X, S)*. Так как в общем случае для любой полугруппы отображений *(X, S)*

$$(S^1, S) | (X, S) \times \ldots \times (X, S) \quad (|X| \text{ раз}),$$

получаем, что $\#_G^c(S^1, S) \leqslant \#_G^c(X, S)$ (см. пункт в) утверждения 2.14 в гл. 2.5 и пункт в) утверждения 2.2 в данной главе).

Пусть элемент $x \in X$, рассмотрим в *X* циклическое подмножество $xS \cup \{x\}$, порожденное элементом *{x}*. Поскольку в S может небыть элемента, оставляющего *x* на месте, мы добавляем *{x}*.

Множество *X* можно представить как объединение (возможно, пересекающихся) циклических подмножеств, порожденных, например, элементами $x_1, \ldots, x_n$. Пусть $I_n = \{1, \ldots, n\}$. Тогда легко показать, что

$$(X, S) | (S^I \times I_n, S^I \times \{0\}) = (S^I, S^I) \times (I_n, \{0\}),$$

где элемент {0} действует как единица на множество I_n и

$$(S^I \times I_n, S^I \times \{0\}) \supseteq (S^I \times I_n, S \times \{0\}) \xrightarrow{(\theta, \varphi)} (X, S),$$

где $\theta(s, i) = x_i s$ и $\theta(I, i) = x_i$ и $\varphi(s, 0) = s$.

Следовательно, поскольку $(S^I, S^I) | (S^1, S) \times (\{0\}^I, \{0\}^I)$, получаем

$$(X, S) \in \overline{W}[\mathrm{PRIMES}(S) \cup \{U_3\}],$$

и в силу пункта в) утверждения 2.2

$$\#_G^c(S^I, S^I) = \#_G^c(S^1, S).$$

Следовательно, $\#_G^c(X, S) = \#_G^c(S^1, S) = \#_G^b(S)$.

Подводя итог, можно сказать, что доказана эквивалентность определений сложности а), б) и в). Поэтому дальше мы пишем $\#_G(S)$

Для $\#_G^a(S) = \#_G^b(S) = \#_G^c(X, S)$.

Мы надеемся, что читатель отчетливо представляет себе естественность и важность сложности $\#_G(S)$ в каскадной декомпозиции конечных автоматов и в структурной теории конечных полугрупп.

Введение сложности представляет собой один из этапов явного описания компонент, необходимых для композиции данного автомата или полугруппы с учетом порядка, в котором компоненты располагаются.

Изложим теперь некоторые простые факты и приведем примеры.

1.6. Утверждение. а) Если $T|S$, то $\#_G(T) \leqslant \#_G(S)$.

б) Если $T \mid (X_2, S_2)\mathrm{w}(X_1, S_1)$ (или в эквивалентной формулировке T^M моделируется каскадным соединением автоматов M_1 и M_2, таких, что $M_1^S = S_1$ и $M_2^S = S_2$), то $\#_G(T) \leqslant \#_G(S_2) + \#_G(S_1)$.

в) Групповая сложность каскадного соединения данного автомата с комбинаторным не превосходит групповой сложности данного автомата.

1.7. Примеры (некоторые примеры приводятся без доказательств),

а) Если S — простая или 0-простая полугруппа, то $\#_G(S) \leqslant 1$.

Из теории полугрупп известно, что если S — простая или 0-простая полугруппа, то полугруппа S^0 изоморфна регулярной рисовской полугруппе матричного типа $\mathcal{M}^0(G; A, B; C)$, в которой $(g, a, b)(g', a', b') = (g'', a, b')$,где $g'' = gC(b, a')g$. Если теперь посмотреть на каскадное соединение на рис. 1, в котором самое большее — один групповой автомат и которое моделирует $\mathcal{M}^0(G; A, B; C)$, то сразу придем к требуемому результату.

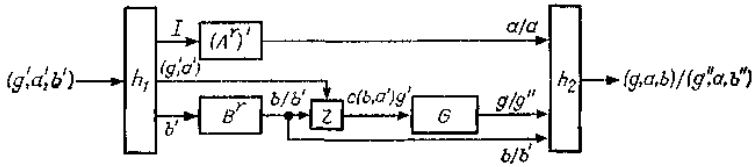


Рис. 1

б) $\#_G[F_R(X_n)] = n - 1$.

Кроме того, $\#_G[F_L(X_n)] = n - 1$. Это равенство получено Деннисом Р. Алленом. Отметим, что справедливо соотношение $F_L(X_n) = r[F_R(X_n)]$. Однако это еще не доказывает второго равенства, так как при выполнении операции r сложность может измениться. В действительности Зальштейн построил полугруппы, сложность которых при выполнении операции r может измениться на любое число.

в) Пусть $G \neq \{1\}$ — группа и n — некоторое положительное целое число. Тогда существует полугруппа S, такая, что 1) S есть объединение групп, каждая из которых изоморфна группе G и
2) $\#_G(S) = n$.
г) Пусть S — абелева полугруппа. Тогда $\#_G(S) \leqslant 1$.
д) Пусть S — инверсная полугруппа. Тогда $\#_G(S) \leqslant 1$.

2.6.2 Определение сложности

В примерах 1.7, приведенных в предыдущем параграфе, мы оставили без доказательства утверждение, что $\#_G[F_R(X_n)] = n - 1$. Отметим, что из этого соотношения следует существование полугрупп с любой, наперед заданной групповой сложностью. В настоящем параграфе мы проверим справедливость этого утверждения, а также, введя более удобное определение сложности, выясним и изучим более тонкие свойства этой величины.
Новое определение сложности отличается от определения групповой сложности тем, что мы учитываем также комбинаторные компоненты и указываем, какая первая координата — групповая или комбинаторная.
2.1. Определение. Пусть *(X, S)* — полугруппа преобразований. Тогда *величиной сложности* $\#(X, S)$ *(*это не та же самая величина, что групповая сложность $\#_G(S)$) полугруппы *(X, S)* является наименьшее целое положительное число *n,* такое, что $(X, S) \mid (X_n, S_n) \wr \ldots \wr (X_1, S_1)$, где или
а) $(X_1, S_1), (X_3, S_3), (X_5, S_5), \ldots$ будут группами преобразований и S_2, S_4, S_6, ... будут комбинаторными полугруппами, или б) S_1, S_3, S_5, ... — комбинаторные полугруппы и (X_2, S_2), (X_4, S_4), (X_6, S_6), ... — группы преобразований.
Сложность C (X, S) полугруппы преобразований *(X, S)* равна:
(*n,* **G**) тогда и только тогда, когда а) справедливо при $n = \#(X, S)$, но б) не справедливо никогда при $n = \#(X, S)$;
(n, **C**) тогда и только тогда, когда б) справедливо при $n = \#(X, S)$, но а) никогда не справедливо при $n = \#(X, S)$;
$(n, \mathbf{C} \vee \mathbf{G})$ тогда и только тогда, когда а) и б) могут иметь место при $n = \#(X, S)$.
2.2. Утверждение. а) Множество всех сложностей представляет собой структуру с упорядочением $\leq$, где $(n, \alpha) \leqslant (m, \beta)$ тогда и только , тогда, когда или $(n, \alpha) = (m, \beta)$, или $n < m$, или $n = m$ и $\alpha = \mathbf{C} \vee \mathbf{G}$. Минимальным элементом структуры является элемент $(1, \mathbf{C} \vee \mathbf{G})$.

б) Если $(X, S) \mid (Y, T)$, то $C(X, S) \leqslant C(Y, T)$.
в) $C[(X_1, S_1) \times \ldots \times (X_n, S_n)] = \mathrm{LUB}\{C(X_i, S_i) : i = 1, \ldots, n\}$.

Доказательство. Рекомендуем читателю самостоятельно с максимальной тщательностью провести доказательство пунктов а) и б). Перейдем к пункту в). Так как $(X_i, S_i) \mid (X_1, S_1) \times \ldots \times (X_n, S_n)$, мы имеем $C(X_i, S_i) \leqslant C[(X_1, S_1) \times \ldots \times (X_n, S_n)]$, т. е. доказано неравенство в одну сторону. Теперь достаточно доказать неравенство в другую сторону. Но если мы докажем, что

$$[(X_2, S_2) \wr (X_1, S_1) \times [(Y_2, T_2) \wr (Y_1, T_1)]] \mid [(X_2, S_2) \times (Y_2, T_2)] \wr [(X_1, S_1) \times (Y_1, T_1)],$$

то, рассматривая групповые компоненты для каждой (X_i, S_i) и опираясь на полученный ранее результат, получим требуемое неравенство. Проверим это соотношение:

$$[(X_2, S_2) \times (Y_2, T_2)] \wr [(X_1, S_1) \times (Y_1, T_1)] \cong [(X_2 \times Y_2 \times X_1 \times Y_1),\ F(X_1 \times Y_1,\ S_2 \times T_2) \times_Y (S_1 \times T_1)].$$

Предположим, что p_1 и p_2 — отображения проекций на $S_2 \times T_2$. Пусть F есть множество $f \in F(X_1 \times Y_1,\ S_2 \times T_2)$, таких, что $\{p_1 f(x_0, y) : y \in Y_1\}$ — единственный элемент для каждого $x_0 \in X_1$ и $\{p_2 f(x, y_0) : x \in X\}$ — единственный элемент для каждого $y_0 \in Y_1$.

Множество F представляет собой подполугруппу в $F(X_1 \times Y_1, S_2 \times T_2)$, кроме того, $Y(S_1 \times T_1) F \subseteq F$, поэтому $F \times {}_Y(S_1 \times T_1)$ — подполугруппа в

$$F(X_1 \times Y_1,\ S_2 \times T_2) \times_Y (S_1 \times T_1).$$

Если $f \in F$, то существуют элементы $f_1 \in F(X_1, S_2)$ и $f_2 \in F(Y_1, T_2)$, такие, что $f_1(x_1) = p_1 f(x_1, y)$ для всех элементов $y \in Y_1$ и $f_2(y_1) = p_2 f(x, y_1)$ для всех элементов $x \in X_1$. Тогда

$$[(X_2 \times Y_2 \times X_1 \times Y_1),\ F \times_Y (S_1 \times T_1)] \xrightarrow{(\theta, \varphi)} [(X_2 \times X_1 \times Y_2 \times Y_1),\ (F(X_1, S_2) \times_Y S_1) \times (F(Y_1, T_2) \times_Y T_1)] \cong \cong [(X_2, S_2) \wr (X_1, S_1)] \times [(Y_2, T_2) \wr (Y_1, T_1)],$$

где

$$\theta(x_2, y_2, x_1, y_1) = (x_2, x_1, y_2, y_1) \text{ и } \varphi[f, (s, t)] = [(f_1, s), (f_2, t)].$$

Проверим, что отображение φ — эпиморфизм и что

$$\theta((x_2, y_2, x_1, y_1) \cdot [f, (s, t)]) = \theta(x_2, y_2, x_1, y_1) \cdot \varphi[f, (s, t)].$$

Тогда требуемое соотношение доказано.

2.3.Теорема. Пусть (X, S) — полугруппа преобразований. Тогда

$C(X, S) = C(S^1, S)$, исключая случай, когда S — группа и единица из S действует не как тождественная функция на *X*. В этом случае имеем

$$C(X, G) = \begin{cases} (1, \mathbf{C}), & \text{если } G = \{1\}; \\ (2, \mathbf{C} \vee \mathbf{G}), & \text{если } G \text{ — нетривиальная группа.} \end{cases}$$

Доказательство. Доказательство этой теоремы совершенно аналогично доказательству теоремы 1.5. Сперва доказывается, что $C(X, \{0\}) = (1, \mathbf{C} \vee \mathbf{G})$, если {0} действует как единица. Затем, так как $C(S', S') = C(S^1, S)$, исключая случай, когда S — группа, легко получить общий результат. Специальные случаи также доказываются без труда.

2.4. Обозначение. Мы будем писать *C (S),* имея в виду *C* (S^1, S), и всегда $C(S^1, S) = C(X, S)$, кроме упомянутых особых случаев. Однако, когда для нахождения сложности производится комбинаторно-групповая декомпозиция, из определения вытекает, что эти случаи исключаются, поэтому они не должны нас беспокоить. Следовательно, в интересах единства обозначений и терминологии можно их вообще не рассматривать. Таким образом, $C(S) = (1, \mathbf{C} \vee \mathbf{G})$ тогда и только тогда, когда $S = \{0\}$. $C(S) = (1, \mathbf{G})$ тогда и только тогда, когда S′— нетривиальная группа, и $C(S) = (1, \mathbf{C})$ тогда и только тогда, когда S — нетривиальная комбинаторная полугруппа.

2.5. Определение. Мы определяем сложение $\oplus$ для сложностей с помощью следующих таблиц:

m четное

$\oplus$	$(m, \mathbf{G})$	$(m, \mathbf{C})$	$(m, \mathbf{C} \vee \mathbf{G})$
$(n, \mathbf{G})$	$(n+m, \mathbf{G})$	$(n+m-1, \mathbf{C})$	$(n+m-1, \mathbf{C})$
$(n, \mathbf{C})$	$(n+m-1, \mathbf{G})$	$(n+m, \mathbf{C})$	$(n+m-1, \mathbf{G})$
$(n, \mathbf{C} \vee \mathbf{G})$	$(n+m-1, \mathbf{G})$	$(n+m-1, \mathbf{C})$	$(n+m-1, \mathbf{C} \vee \mathbf{G})$

m нечетное

$\oplus$	$(m, \mathbf{G})$	$(m, \mathbf{C})$	$(m, \mathbf{C} \vee \mathbf{G})$
$(n, \mathbf{G})$	$(n+m-1, \mathbf{G})$	$(n+m, \mathbf{C})$	$(n+m-1, \mathbf{G})$
$(n, \mathbf{C})$	$(n+m, \mathbf{G})$	$(n+m-1, \mathbf{C})$	$(n+m-1, \mathbf{C})$
$(n, \mathbf{C} \vee \mathbf{G})$	$(n+m-1, \mathbf{G})$	$(n+m-1, \mathbf{C})$	$(n+m-1, \mathbf{C} \vee \mathbf{G})$

Тогда, например,

а) $(n, \mathbf{C}) \oplus (1, \mathbf{G}) = (n+1, \mathbf{G})$;
б) $(n, \mathbf{G}) \oplus (1, \mathbf{G}) = (n, \mathbf{C} \vee \mathbf{G}) \oplus (1, \mathbf{G}) = (n, \mathbf{G})$,
в) $(n, \mathbf{G}) \oplus (1, \mathbf{C}) = (n+1, \mathbf{C})$ и
г) $(n, \mathbf{C}) \oplus (1, \mathbf{C}) = (n, \mathbf{C} \vee \mathbf{G}) \oplus (1, \mathbf{C}) = (n, \mathbf{C})$.

Отметим, что операция $\oplus$ не является коммутативной.

2.6. Утверждение. Если $S \mid (X_2, S_2)\mathrm{w}\,(X_1, S_1)$, то $C(S) \leqslant C(S_2) \oplus \oplus C(S_1)$.

Следующие три леммы приводят к основной теореме настоящего параграфа.

2.7. Лемма. Если $S \mid (X_2, T)\mathrm{w}\,(X_1, G)$, где (X_1, G) есть группа преобразований, то $C[IG(S)] \leqslant C(T)$. (Напомним, что символ $IG(S)$ обозначает полугруппу, порожденную множеством идемпотентов $E(S)$ полугруппы S.)

Доказательство. $(X_2, T)\mathrm{w}(X_1, G) \cong F(X_1, T) \times_Y G$ и если элемент e есть единица группы G, то $Y(e)$ является единичным автоморфизмом для $F(X_1, T) \equiv F$. Если $(f, g) \in E(F \times_Y G)$, то $(f, g) = (f, g)^2 = = [fY(g)f, g^2]$. Поэтому $g = e$ и $f^2 = f$. Следовательно, $E(F \times_Y G) = E(F) \times E(G) = E(F) \times \{e\}$ и $IG(F \times_Y G) \cong IG(F)$. Тогда, поскольку из соотношения $S_1 \mid S_2$ вытекает $IG(S_1) \mid IG(S_2)$, мы имеем $IG(S) \mid IG(F) \mid F = F(X_1, T)$. Но так как $F(X_1, T) \cong T \times \ldots \times T$ ($|X_1|$ раз), $C[F(X_1, T)] = C(T)$. Следовательно, $C[IG(S)] \leqslant C(T)$. Лемма доказана.

2.8. Лемма. Пусть S — моноид, а C — такая комбинаторная полугруппа, что $U_1 \subseteq C$. (Напомним, что $U_1 = \{r_0, r_1\}Y$.) Тогда $S \mid IG[(X_2, S)\mathrm{w}\,(X_1, C)]$ и $C(S) \leqslant C\{IG[(X_2, S)\mathrm{w}\,(X_1, C)]\}$.

Доказательство. Из условия леммы следует, что существует элемент $x \in X_1$, такой, что $xr_0 \neq xr_1$. Положим $x_0 = xr_0$, $x_1 = xr_1$. Тогда $x_i r_j = x_j$, $i, j = 0, 1$. Введем множество $Y = \{x_0, x_1\}$, тогда (Y, U_1) будет полугруппой преобразований и $(Y, U_1) \mid (X_1, C)$, следовательно,

$$(X_2, S)\mathrm{w}\,(Y, U_1) \mid (X_2, S)\mathrm{w}\,(X_1, C)$$

и

$$IG[(X_2, S)\mathrm{w}\,(Y, U_1)] \mid IG[(X_2, S)\mathrm{w}\,(X_1, C)].$$

$(X_2, S)\mathrm{w}\,(Y, U_1) \cong F(Y, S) \times_Y U_1$. Обозначим символом F_S подполугруппу в $F(Y, S)$, состоящую из всех постоянных отображений, т. е. $F_S = \{f_s \in F(Y, S) : f_s(x) = s$ для всех $x \in Y\}$. $F_S \times_Y \{r_0\}$ является подполугруппой в $F(Y, S) \times_Y U_1$ и $F_S \times_Y \{r_0\} \cong S$ при отображении, переводящем $(f_s, r_0) \to s$.

Предположим теперь, что $f \in F(Y, S)$ будет функцией, определяемой соотношениями $f(x_0) = 1$, $f(x_1) = s$. Тогда $(f_s, r_0) = (f_1, r_1)(f, r_0)$ и оба элемента (f_1, r_1) и (f, r_0) представляют собой идемпотенты. Следовательно,

$$F_S \times_Y \{r_0\} \subseteq IG[(X_2, S) \mathrm{w} (Y, U_1)].$$

2.9. Лемма. Предположим, что

$$(X_2, S_1)\mathrm{w}(X_1, G) \mid (Y_2, S_2)\mathrm{w}(Y_1, C),$$

где S_1 — моноид, S_2 — полугруппа, (X_1, G) — нетривиальная группа преобразований и C — комбинаторная полугруппа. Тогда $C(S_1) \leqslant \leqslant C(S_2)$.

Доказательство. $(X_2, S_1)\mathrm{w}(X_1, G) \cong F(X_1, S_1) \times_Y G$.

Пусть $H = \{(f_1, g) : g \in G\}$,

где $f_1(x) = 1$ — единица моноида S_1 для всех элементов $x \in X_1$. H является подгруппой в $F(X_1, S_1) \times_Y G$.

Предположим, что e — единица группы G и $h \neq e \in G$. Тогда существует элемент $x_0 \in X_1$, такой, что $x_0 h \neq x_0 e = x_0$. Введем множество $T = \{(f, e) : f(x_0) = 1\}$. T будет подполугруппой в $F(X_1, S_1) \times_Y G$.

Пусть (f, g) — общий элемент произведения $F(X_1, S_1) \times_Y G$. Рассмотрим элементы $f', f'' \in F(X_1, S_1)$, определяемые соотношениями

$$f'(x) = \begin{cases} f(x_0), & \text{если } x = x_0 h; \\ 1 & \text{в противном случае;} \end{cases}$$

$$f''(x) = \begin{cases} 1, & \text{если } x = x_0; \\ f(x) & \text{в противном случае.} \end{cases}$$

Очевидно, что

(f', e) и $(f'', e) \in T$, а $(f, g) = (f_1, h)(f', e)(f_1, h^{-1})(f'', e)(f_1, g)$.

(Как обычно, символ $\langle X \rangle$ обозначает полугруппу, порожденную множеством X). Следовательно, мы показали, что

$\langle H \cup T \rangle = (X_2, S_1)\mathrm{w}(X_1, G)$.

Пусть $P_1 = F(X_1, S_1) \times_Y \{e\}$ есть подполугруппа в (X_2, S_1) w w (X_1, G). Очевидно, что $T \subseteq P_1$. Определим эпиморфизм $\theta_1 : P_1 \twoheadrightarrow S_1$, полагая $\theta_1(f, e) = f(x_0)$. Отметим, что

$\theta_1(T) = \{1\}$.

Если элемент $t \in T$, то множество tP_1t будет подполугруппой полугруппы $t[(X_2, S_1)\mathrm{w}(X_1, G)]t$ и $\theta_1(tP_1t) = S_1$. Следовательно,

$S_1 \mid t[(X_2, S_1)\mathrm{w}(X_1, G)]t$ для всех элементов $t \in T$.

Рассмотрим гомоморфизм $(Y_2,\ S_2)\mathrm{w}\,(Y_1,\ C) \supseteq M \xrightarrow{\varphi} (X_2,\ S_1)\mathrm{w}\ \mathrm{w}\ (X_1,\ G)$. Тогда существует подгруппа H' в M, такая, что $\varphi(H') = H$.

Введем множество $R = \varphi^{-1}(T)$ и положим $T' = 1'R1'$, где $1'$— единичный элемент в H'. Тогда $\varphi(T') = T$. Пусть $M' = \langle H' \cup T' \rangle$— полугруппа, порожденная множествами H' и T'. Отметим, что $1'$ является единичным элементом в M' и $\varphi(M') = (X_2,\ S_1)\ \mathrm{w}\ (X_1,\ G)$. Следовательно, $S_1 \mid t'M't'$ для всех элементов $t' \in T'$.

Предположим, что $p_1 : (Y_2, S_2)\mathrm{w}\ \ (Y_1,\ \ C) \twoheadrightarrow C$ гомоморфизм проекции. Пусть t' — идемпотент в $K(T')$. Тогда $t'T't'$ есть подгруппа в $K(T')$. Гомоморфизм p_1 переводит подгруппы узлового произведения $(Y_2,\ S_2)\mathrm{w}\ (Y_1,\ C)$ в одноэлементные множества, так как C — комбинаторная полугруппа. Следовательно, $p_1(H') = \{\hat{1}\}$— единичный элемент в $p_1(M')$. Пусть $p_1(t') = \hat{t}$. Тогда

$$\begin{aligned} p_1(t'M't') &= \hat{t}\,p_1 \langle H' \cup T' \rangle\ \hat{t} \\ &= \hat{t} \langle \{1\} \cup p_1(T') \rangle\ \hat{t} \\ &= \hat{t}\,p_1(T')\,\hat{t} \\ &= p_1(t'T't') \\ &= \{p_1(t')\} = \{\hat{t}\}. \end{aligned}$$

Следовательно, первая координата полугруппы $t'M't'$ как подполугруппа в $(Y_2,\ S_2)\mathrm{w}\ (Y_1,\ C) \cong F(Y_1,\ S_2) \times_Y C$ состоит из единственного идемпотентного элемента t. Определим гомоморфизм $\psi : t'M't' \twoheadrightarrow F(Y_1,\ S_2)$, полагая $(f,\ \hat{t}) \to {}^{\hat{t}}f$. (Напомним, что ${}^{\hat{t}}f \equiv Y(\hat{t})\ f$.)

Пусть $t' = (\bar{f},\ \bar{t})$. Элемент t' является единицей для $t'\ M'\ t'$. Предположим, что $\psi(f, \hat{t}) = \psi(f',\ \hat{t})$, т. е. ${}^{\hat{t}}f = {}^{\hat{t}}f'$. Тогда

$$(f, \hat{t}) = (\bar{f},\ \hat{t})(f,\ \hat{t}) = (\bar{f}\,{}^{\hat{t}}f,\ \ t) = (\bar{f}\,{}^{\hat{t}}f',\ \hat{t}) = (\bar{f},\ \hat{t})(f',\ \hat{t}) = (f',\ \hat{t}).$$

Поэтому отображение ψ взаимно однозначное и $t'M't' \mid F(Y_1,\ S_2)$. Следовательно, $S_1 \mid F(Y_1,\ S_2)$ и

$$C(S_1) \leqslant C[F(Y_1,\ S_2)] = C(S_2).$$

Лемма доказана.

2.10. Теорема. Пусть S — моноид.

а) Пусть C — комбинаторная полугруппа, содержащая полугруппу U_1. Тогда

$$C[(X,\ S)\,\mathrm{w}\,(Y,\ C)] = C(S) \oplus (1, \mathbf{C}). \quad (2.1)$$

б) Пусть (Y, G) — нетривиальная группа преобразований. Тогда

$$C\,[(X,\ S)\,\mathrm{w}\,(Y,\ G)] = C\,(S) \oplus (1,\ \mathbf{G}). \quad \textbf{(2.2)}$$

Доказательство, а) Если справедливо равенство $C\ (S) = (n,\ \mathbf{C})$, то соотношение (2.1) выполняется. Предположим, что сложность $C\ (S) = (n,\ \mathbf{G})$ или $(n + 1,\ \mathbf{C} \vee \mathbf{G})$. Тогда

$$(n,\ \mathbf{G}) \leqslant C\,[(X,\ S)\,\mathrm{w}\,(Y,\ C)] \leqslant (n + 1,\ \mathbf{C}).$$

Предположим, что $C\,[(X,\ S)\mathrm{w}\,(Y,\ C)] \neq (n + 1,\ \mathbf{C})$. Тогда

$$C\,[(X,\ S)\,\mathrm{w}\,(Y,\ C)] = (n,\ \mathbf{G}) \text{ или } (n + 1,\ \mathbf{C} \vee \mathbf{G}).$$

В обоих случаях существует полугруппа T, такая, что

$$(X,\ S)\,\mathrm{w}\,(Y,\ C)\,|\,(X_1,\ T)\,\mathrm{w}\,(Y_1,\ G),$$

где $C\ (T) \leqslant (n,\ \mathbf{C})$. В силу лемм 2.7 и 2.8 имеем

$$C\ (S) \leqslant C\,(IG\,[(X,\ S)\,\mathrm{w}\,(Y,\ C)]) \leqslant C\ (T),$$

поэтому получаем противоречие $C\ (S) \leqslant (n,\ \mathbf{C})$. Следовательно, $C\,[(X,\ S)\mathrm{w}\,(Y,\ C)] = (n + 1,\ \mathbf{C})$ и, так как приведенные рассуждения справедливы при всех $n \geq 1$,

$$C\,[(X,\ S)\,\mathrm{w}\,(Y,\ C)] = C\,(S) \oplus (1,\ \mathbf{C}).$$

б) Если справедливо равенство $C\ (S) = (n,\ \mathbf{G})$, то соотношение (2.2) выполняется. Предположим, чго $C\ (S) = (n,\ \mathbf{C})$ или $(n + 1,\ \mathbf{C} \vee \mathbf{G})$.

Тогда

$$(n,\ \mathbf{C}) \leqslant C\,[(X,\ S)\,\mathrm{w}\,(Y,\ G)] \leqslant (n + 1,\ \mathbf{G}).$$

. Предположим, что $C\,[(X,\ S)\,\mathrm{w}\,(Y,\ G)] \neq (n + 1,\ \mathbf{G})$. Тогда

$$C\,[(X,\ S)\,\mathrm{w}\,(Y,\ G)] = (n,\ \mathbf{C}) \text{ или } (n + 1,\ \mathbf{C} \vee \mathbf{G}).$$

В обоих случаях существует полугруппа T, такая, что $(X,\ S)\mathrm{w}\ (Y,\ G)\,|\,(X_1,\ T)\mathrm{w}\ (Y_1,\ C)$, где $C\ (T) \leqslant (n,\ \mathbf{G})$. Но согласно лемме 2.9 $C\ (S) \leqslant C\ (T) \leqslant (n,\ \mathbf{G})$, а это противоречие.

Следовательно, $C\,[(X,\ S)\mathrm{w}\ (Y,\ G)] = (n + 1,\ \mathbf{C})$, и так как эти рассуждения справедливы для всех $n \geq 1$, соотношение (2.2) доказано.

2.11. Замечание. Мы приводим контрпримеры, подтверждающие, что условия, сформулированные в пунктах а) и б) теоремы 2.10, необходимы для выполнения равенств (2.1) и (2.2) соответственно.

а) $C\ (A^1\ \mathrm{w}\ G) = (2,\ \mathbf{C} \vee \mathbf{G})$ не совпадает с $(2,\ \mathbf{G})$. Отметим, что условие теоремы не выполнено, так как A^1 не является моноидом.

б) $C\ (G\mathrm{w}A^{(1)}) = (2,\ \mathbf{C} \vee \mathbf{G})$ не совпадает с $(2,\ \mathbf{C})$. Отметим, что здесь условие теоремы также нарушено, поскольку $U_1 \not\subseteq A^{(1)}$.

2.12. Следствие. Существуют полугруппы любой сложности. Другими словами, если имеется произвольное каскадное соединение групповых автоматов, то в общем случае невозможно перерасположить их так, чтобы групповые автоматы были собраны вместе.

Доказательство. Тривиальная полугруппа имеет сложность $(1, \mathbf{C} \vee \mathbf{G})$; $C(U_3) = (1, \mathbf{C})$; $C(G) = (1, \mathbf{G})$, G — нетривиальная группа, $C(U_3 \mathrm{w} G) = (2, \mathbf{G})$, $C(G \mathrm{w} U_3) = (2, \mathbf{C})$ и т. д. Если $C(S) = (n, \mathbf{G})$ и $C(T) = (n, \mathbf{C})$, то $C(S \times T) = (n + 1, \mathbf{C} \vee \mathbf{G})$.

2.6.3. Сложность и идеалы

3.1. Теорема (обобщенная теорема о вложении). Пусть S — полугруппа с подполугруппой V и подмножеством T, такими, что $VT = S$. Пусть $t : S \to T$ и $v : S \to V$ — отображения, выбранные так, что $v(s)t(s) = s$. Пусть отображение $f_t : \Sigma S \to T$ определяется по индукции с помощью соотношений

$$f_t(s_1) = t(s_1),$$

$$f_t(s_1, \ldots, s_n) = t[f_t(s_1, \ldots, s_{n-1}) s_n].$$

Тогда автомат S^M можно моделировать каскадным соединением автоматов V^M и f_t^M. В частности,

$$\#_G(S) \leqslant \#_G(V) + \#_G(f_t^S).$$

Доказательство. Рассмотрим каскадное соединение, представленное на рис. 2, где

$$Z : T \times S \to T : (x, s') \to v(xs')$$

и

$$h : V \times T \to S : (y, x) \to yx.$$

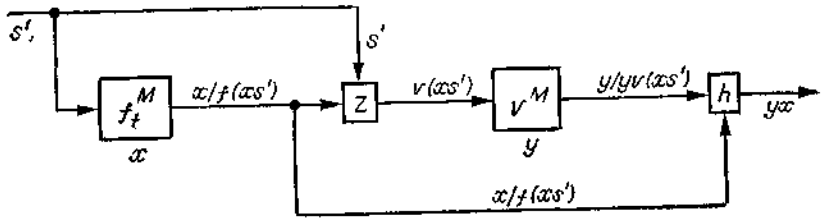


Рис. 2

Тогда это соединение «ведет себя» как автомат S^M, так как, если его текущее состояние (x, y) кодируется как $s = yx$ и его вход есть s', следующее состояние соединения будет $[t(xs'), yv(xs')]$ и оно кодируется как $yv(xs')\, t(xs') = yxs' = ss'$. Приведенное далее определение мотивируется изложенным вслед за ним следствием из теоремы 3.1.

3.2. Определение. Пусть T — подполугруппа. Определим тогда автомат REP (T), у которого входной алфавит есть T, множество состояний есть T^1, функция переходов определяется как

$$(x,\ s) \to s, \text{ если } x = 0,$$

$$(x,\ s) \to xs, \text{ если } x \neq 0,$$

и выходной сигнал всегда равен его состоянию.

3.3. Следствие. Пусть S — полугруппа с подполугруппой T и левым идеалом V, такими, что $T \cup V = S$. Тогда $T \cap V$ является идеалом в T. Мы можем ввести фактор-полугруппу Риса $T_V = T/(T \cap V)$. Тогда автомат S^M можно моделировать каскадным соединением автоматов $\mathrm{REP}\ (T_V)$ (с состоянием 0, декодируемым как 1) и $(V^1)^M$.

Доказательство. Отметим, что $S^1 = V^1 T^1_V$. Определим отображения $t : S \to T^1_V$ и $v : S \to V^1$, полагая

$$t(s) = s, \qquad v(s) = 1, \text{ если } s \in T - V,$$

$$t(s) = 1, \qquad v(s) = s, \text{ если } s \notin T - V.$$

Тогда $v(s)t(s) = s$ для всех $s \in S$. Если автомат f_t находится в состоянии x и на вход поступает элемент s, то он переходит в состояние xs, если $xs \in T - V$, и в состояние 1, если $xs \in V \cap T$. Таким образом, это есть автомат $\mathrm{REP}\ (T_V)$ с состоянием 0, декодируемым как 1.

3.4. Лемма. а) Если S — полугруппа с нулевым умножением (т. е. произведение двух любых элементов есть 0) и $T \mid S$, то

$$\#_G\ [\mathrm{REP}\ (T)] \leqslant 1$$

б) Если S есть 0-простая полугруппа и $T \mid S$, то $\#_G\ [\mathrm{REP}\ (T)] \leqslant 1$.

Доказательство, а) Если S — полугруппа с нулевым умножением, то входу t_n, занумерованному нечетным числом, соответствует выход t_n, а выходу, занумерованному четным числом, соответствует выход 0. Следовательно, автомат REP *(S)* можно построить как параллельное соединение только одного группового автомата, а именно $\mathbf{Z}_2 = (\{0, 1\}, +)$, и автомата полугруппы S^r с кодирующей функцией для входов $h_1 : S \to S^r \times \mathbf{Z}_2 : s \to (s, 1)$ и кодирующей функцией для выходов

$$h_1 : S^r \times \mathbf{Z}_2 \to S^1 : \begin{cases} (s, 0) \to 0, \\ (s, 1) \to s. \end{cases}$$

б) Пусть $S \cong \mathcal{M}^0\ (G;\ A,\ B;\ C)$. Неравенство $\#_G\ (S) \leqslant 1$ следует из рис. 3, так как он показывает, что самое большее — один групповой автомат используется в REP (S), а именно автомат $\mathrm{REP}\ (G^0)$.

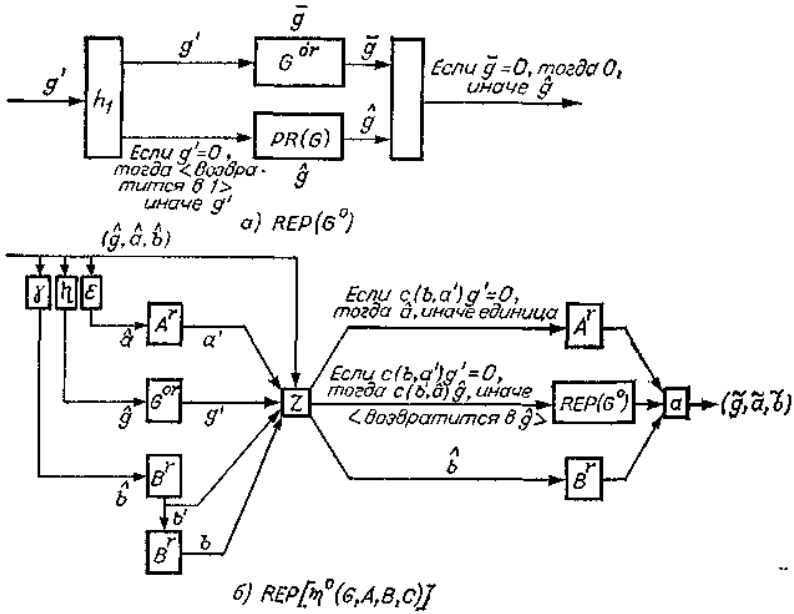


Рис. 3

Отметим, наконец, что если $T \mid S$ и нули полугрупп S и T совпадают (если они существуют), то $\mathrm{REP}\,(T) \mid \mathrm{REP}\,(S)$.

3.5. Определение. Пусть I_1 и I_2 — идеалы полугруппы S. Назовем идеал I_1 *почти максимальным* в I_2, если $I_1 \subseteq I_2$, и фактор-полугруппа Риса I_2/I_1 делит полугруппу с нулевым умножением или 0-простую полугруппу.

3.6. Лемма. Пусть I_1 и I_2 — идеалы некоторой полугруппы и I_1 почти максимален в I_2. Тогда

$$\#_G(I_1) \leqslant \#_G(I_2) \leqslant \#_G(I_1) + 1.$$

Доказательство. В силу следствия 3.3 получаем, что

$$\#_G(I_2) \leqslant \#_G(I_1) + \#_G[\mathrm{REP}\,(I_2/I_1)].$$

Требуемый результат вытекает тогда из леммы 3.4.

Читатель может проверить, что если идеал I_1 *максимальный* в I_2 (т. е. единственный идеал в I_2, кроме I_1 есть сам I_2), то I_1 *почти максимален* в I_2. Тогда можно вывести следующую теорему.

3.7. Теорема. а) Пусть I_1 и I_2 — идеалы полугруппы S и I_1 максимален в I_2. Тогда

$$\#_G(I_1) \leqslant \#_G(I_2) \leqslant \#_G(I_1) + 1.$$

б) Пусть V — идеал в полугруппе S и $\#_G(S) = n > k = \#_G(V)$. Тогда в S существуют идеалы V_n, ..., V_k, такие, что

$$V = V_k \subset V_{k+1} \subset \ldots \subset V_n = S \text{ и } \#_G(V_j) = j,\ k \leqslant j \leqslant n.$$

в) Пусть T — подидеал полугруппы S, т. е. существуют подполугруппы $S_0, \ldots, S_k$, такие, что $T = S_k \subseteq S_{k-1} \subseteq \ldots \subseteq S_0 = S$ и S_j является идеалом в S_{j-1} при $i = 1, \ldots, k$. Предположим, что

$$\#_G(S) = n \leqslant k = \#_G(T).$$

Тогда существуют подидеалы V_n, ..., V_k, такие, что

$$T = V_k \subset V_{k+1} \subset \ldots \subset V_n = S,$$

и V_i является подидеалом в V_{i+1} при $i = k, \ldots, n-1$ и $\#_G(V_i) = i$, $i = k, \ldots, n$.

Тот факт, что каждый идеал I полугруппы S содержится в цепочке идеалов

$$I \subset I_1 \subset I_2 \subset \ldots \subset S,$$

для которой сложность $\#_G$ на каждом шаге увеличивается на 1, называется *непрерывностью сложности по отношению к идеалам.*

Предыдущий пункт в) нельзя усилить, потребовав, чтобы V_i являлся идеалом в V_{i+1}. Следующий контрпример принадлежит Биллу Раундзу. Пусть $N = \{0, n_0, n_1\}$— полугруппа с нулевым умножением и $\mathbf{Z}_2 = \{z_0, z_1\}$ — группа из двух элементов.

Множество $S = N \cup \mathbf{Z}_2$ становится полугруппой с законом умножения:

$$\left.\begin{array}{l} zn = n_0 = nz \\ 0z = z0 = 0 \end{array}\right\} \text{ для } 0 \neq n \in N, z \in \mathbf{Z}_2.$$

Тогда $T = \{0, n_1\}$ есть идеал в N и N есть идеал в S, но T не является идеалом полугруппы S и $\#_G(S) = 1$, тогда как $\#_G(N) = \#_G(T) = 0$.

Теперь мы можем распространить это со сложности 1 на сложность n при помощи теоремы 2.10.

Выберем R с $\#_G(R) = n - 1$. Тогда

$S' = (R \text{w} U_3) \text{w} S$ имеет $\#_G(S') = n$;

$N' = (R \text{w} U_3) \text{w} N$ имеет $\#_G(N') = n - 1$;

$T' = (R \text{w} U_3) \text{w} T$ имеет $\#_G(T') = n - 1$.

T' есть идеал в N'; N' является идеалом в S', но T' не является идеалом в S'.

Отметим, что где-нибудь в этом построении следует использовать полугруппы с нулевым умножением, так как известно, что если I_2/I_1 не является полугруппой с нулевым умножением ни для каких идеалов

I_2 и 1_1 в S, то полугруппа S регулярна и каждый подидеал в S будет идеалом полугруппы S.

3.8. Теорема. Пусть M — приведенный конечный автомат с полугруппой S и множеством состояний Q.

а) Пусть $r_k = \{f \in S : |f(Q)| \leqslant k\}$.

Тогда r_k есть идеал полугруппы S и $\#_G(r_k) \leqslant \#_G(r_{k+1}) \leqslant \#_G(r_k) + 1$ для $1 \leqslant k \leqslant |Q|$.

б) Пусть $\text{spec}\ (M) = \{k > 1 : r_k - r_{k-1} \neq \varnothing\}$. Тогда $\#_G(M) \leqslant |\text{spec}\ (M)|$. (читается: спектр M)

в) Когда $S = F_R(Q)$, тогда $\#_G(M) = |\text{spec}\ (M)| = |Q| - 1$.

Доказательство. Для доказательства пунктов а) и б) нам надо показать только, что если k и $k + j$ принадлежат spec (M), но $r_{k+i} \notin \notin \text{spec}\ (M)$ при i=l,...,j— 1, то r_k является почти максимальным в r_{k+j}. Затем можно применить лемму 3.6. Оставляем это для читателя как упражнение.

Половина пункта в) вытекает из пункта б), а именно из пункта б) следует, что $\#_G [F_R^*(Q)] \leqslant |Q| - 1$.

Доказательство обратного результата, т. е. неравенства $\#_G [F_R(Q)] \geqslant |Q| - 1$, значительно сложнее.

Пункт б) можно доказать также, применяя метод Зейгера: построение последовательности вложенных покрытий $\{C_j : j \in \text{spec}\ (M)\}$, т. е. C_j есть покрытие, состоящее из j элементных подмножеств в Q.

3. Методы декомпозиции абстракных автоматов

3.1. Параллельная декомпозиция автоматов с разделением входов

В этом параграфе докажем теорему о необходимом и достаточном условии представления автомата *параллельной одновременной работой двух более простых автоматов с раздельными входами* и сформулируем алгоритм такого разложения автомата. Другими словами, опишем критерий принадлежности произвольного автомата $A \in \mathfrak{A}_\times$ подполугруппе $\mathbf{A}_\times$ автоматов, представимых параллельной одновременной работой двух или более автоматов с раздельными входами из $\mathfrak{A}_\times$. В данном случае $\mathfrak{A}_\times$—бесконечная полутруппа абстрактных автоматов Мили.

Пусть $A = (X, V, N, v_1 \in V, F(x \in X/n \in N))$ и $B =$

$= (Y, W, S, w_1 \in W, P(y \in Y/s \in S))$ — произвольные абстрактные автоматы Мили. Если

$$R_A = \| r_{\alpha\beta}(x/n) \|,$$

где $\alpha, \beta \in I = \{1, 2, \ldots, k\}$—матрица соединений автомата *A,* а

$$R_B = \| r_{\gamma\delta}(y/s) \|,$$

где $\gamma, \delta \in J = \{1, 2, \ldots, l\}$— матрица соединений автомата *B,* то как известно, матрица соединений R_C автомата С, равного произведению автоматов *A* и *B,* будет иметь вид

$$R_C = R_A \times R_B = \begin{Vmatrix} r_{11}(x/n) \cdot R_B & r_{12}(x/n) \cdot R_B & \ldots & r_{1k}(x/n) \cdot R_B \\ r_{21}(x/n) \cdot R_B & r_{22}(x/n) \cdot R_B & \ldots & r_{2k}(x/n) \cdot R_B \\ \ldots & \ldots & \ldots & \ldots \\ r_{k1}(x/n) \cdot R_B & r_{k2}(x/n) \cdot R_B & \ldots & r_{kk}(x/n) \cdot R_B \end{Vmatrix} =$$

$$= \begin{Vmatrix} R_{11}(z/m) & R_{12}(z/m) & \ldots & R_{1k}(z/m) \\ R_{21}(z/m) & R_{22}(z/m) & \ldots & R_{2k}(z/m) \\ \ldots & \ldots & \ldots & \ldots \\ R_{k1}(z/m) & R_{k2}(z/m) & \ldots & R_{kk}(z/m) \end{Vmatrix} = \| R_{\alpha\beta}(z/m) \|, \tag{3.1}$$

где $R_{\alpha\beta}(z/m)$, $\alpha, \beta \in I$ — клетки порядка *l*.

Из (3.1) следует, что R_C может содержать следующие типы клеток:

1) если $r_{\alpha\beta}(x/n) = 0$, то $R_{\alpha\beta}(z/m) = R_0$, где R_0— нулевая матрица порядка *l*;

2) если $r_{\alpha\beta}(x/n) \neq 0$, то существуют такие элементы $r_{\alpha\beta}(x_i/n_i) = x_i/n_i$ и $r_{\alpha\beta}(x_j/n_j) = x_j/n_j$, что матрицы

$$R_{\alpha\beta}(z_i/m_i) = \begin{Vmatrix} (x_i/n_i)\, r_{11}(y/s) & (x_i/n_i)\, r_{12}(y/s) & \ldots & (x_i/n_i)\, r_{1l}(y/s) \\ (x_i/n_i)\, r_{21}(y/s) & (x_i/n_i)\, r_{22}(y/s) & \ldots & (x_i/n_i)\, r_{2l}(y/s) \\ \ldots & \ldots & \ldots & \ldots \\ (x_i/n_i)\, r_{l1}(y/s) & (x_i/n_i)\, r_{l2}(y/s) & \ldots & (x_i/n_i)\, r_{ll}(y/s) \end{Vmatrix}$$

и

$$R_{\alpha\beta}(z_j/m_j) = \begin{Vmatrix} (x_j/n_j)\, r_{11}(y/s) & (x_j/n_j)\, r_{12}(y/s) & \ldots & (x_j/n_j)\, r_{1l}(y/s) \\ (x_j/n_j)\, r_{21}(y/s) & (x_j/n_j)\, r_{22}(y/s) & \ldots & (x_j/n_j)\, r_{2l}(y/s) \\ \ldots & \ldots & \ldots & \ldots \\ (x_j/n_j)\, r_{l1}(y/s) & (x_j/n_j)\, r_{l2}(y/s) & \ldots & (x_j/n_j)\, r_{ll}(y/s) \end{Vmatrix}$$

по буквам входного (выходного) алфавитов либо равны в случае $x_i = x_j (n_i = n_j)$, либо между буквами z_i и z_j (m_i и m_j), стоящими на соответствующих местах клеток, можно установить взаимно однозначное соответствие.

Матрицу соединений вида (3.1) назовем *правильной клеточной матрицей соединений* (ПКМС).

Сформулируем теперь следующее предложение.

Теорема 3.1. *Автомат A с $n = k \cdot l$ состояниями представим произведением автоматов A_1 и A_2 соответственно с k и l состояниями (параллельной одновременной работой двух автоматов с раздельными входами), если и только если существует подстановка $t \in T$ алфавита состояний, которая преобразует матрицу соединений R автомата A к виду правильной клеточной матрицы соединений.*

Необходимость следует из определения правильной клеточной матрицы соединений автомата.

Приведем доказательство достаточности. Пусть матрица соединений R автомата A является ПКМС или существует подстановка $t \in T$, которая приводит ее к виду ПКМС. По матрице R строим матрицы соединений R_1 и R_2 автоматов A_1 и A_2 следующим образом. Матрица R_1 порядка k (алфавит состояний V с начальным состоянием v_1) имеет столько переходов, сколько содержится ненулевых клеток порядка l в матрице R. Переходам (выходам), соответствующим различным клеткам матрицы R, приписываем различные буквы $x \in X (l \in L)$, а одинаковым клеткам — одинаковые буквы $x \in X (l \in L)$ входного (выходного) алфавита. Матрица R_2 порядка l (алфавит состояний W с начальным состоянием w_1) совпадает с любой ненулевой клеткой порядка l матрицы R, если заменить различные буквы $z \in Z (m \in M)$ входного (выходного) алфавита различными буквами $y \in Y (s \in S)$, а одинаковые — одинаковыми буквами $y \in Y (s \in S)$. Легко видеть, что $A_1 \times A_2 = A$. Теорема доказана.

Сформулируем теперь алгоритм представления произвольного автомата $A \in \mathfrak{A}_\times$ произведением двух автоматов с раздельными входами.

1°. *Подсчитываем число n состояний автомата A. Если $n = k \cdot l$, то переходим к* 2°. *Если n — простое число, то к* 7°.

2°. *Матрицу соединений R_A автомата A разбиваем на k^2 клеток порядка l каждая. Если R_A — правильная клеточная матрица соединений, то переходим к* 6°. *В противном случае к* 3°.

3°. *По матрице соединений R_A записываем матрицу смежности R автомата A. Переходим к* 4°.

4°. *Применяя метод разложения графов в произведение двух графов, ищем подстановку, переводящую матрицу смежности R в правильную клеточную матрицу R'. Если $t \in T$ — искомая подстановка, то*

переходим к 5°. *Если такая подстановка не существует, то переходим к* 7°.

5°. *Применяем полученную подстановку к матрице соединений* R_A *автомата А. Если получаем правильную клеточную матрицу соединений* R'_A, *то переходим к* 6°. *В противном случае к* 7°.

6°. *По матрице* R'_A *строим матрицы соединений* R_{A_1} *и* R_{A_2} *автоматов* A_1 *и* A_2 *так, как это показано в доказательстве достаточности теоремы* 3.1.

7°. *Автомат А не разложим по операции умножения.*

Пример 3.1. Пусть

$$A = (Z, Q, M, q_1 \in Q, K(z \in Z/m \in M))$$

— абстрактный автомат, графоид которого показан на рис. 3.1.

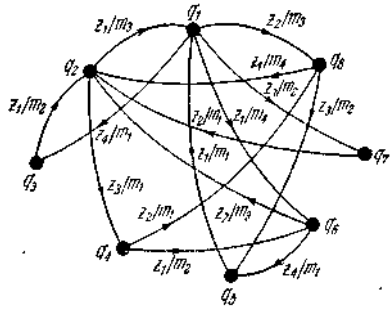


Рис. 3.1.

Определить, разложим ли автомат A по операции умножения, и если да, то представить его произведением двух автоматов.

Матрицасоединений R_A автомата A имеет вид

$$R_A = \left\|\begin{array}{cccc:cccc} 0 & 0 & z_4/m_1 & 0 & 0 & z_1/m_4 & z_3/m_2 & z_2/m_3 \\ z_1/m_3 & 0 & 0 & z_3/m_1 & 0 & 0 & 0 & 0 \\ 0 & z_1/m_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & z_1/m_2 & 0 & z_2/m_1 \\ \hdashline z_1/m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & z_2/m_3 & 0 & 0 & z_4/m_1 & 0 & 0 & 0 \\ 0 & z_2/m_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & z_1/m_4 & 0 & 0 & z_3/m_2 & 0 & 0 & 0 \end{array}\right\|.$$

Запишем по ней матрицу смежности

$$R=\left\|\begin{array}{cccc:cccc}0&0&1&0&0&1&1&1\\1&0&0&1&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&0&0&0&1&0&1\\\hdashline 1&0&0&0&0&0&0&0\\0&1&0&0&1&0&0&0\\0&1&0&0&0&0&0&0\\0&1&0&0&1&0&0&0\end{array}\right\|.$$

Поскольку R не является правильной клеточной матрицей, то определим пару (π, ρ) разбиений. Выпишем полустепени исхода s и захода p каждой вершины графа автомата и образуем множества возможных разложений этих чисел на два сомножителя. Получим

q_1 | $s_1 = 4 = 1 \cdot 4,\ 4 \cdot 1,\ \underline{2 \cdot 2},$ $\quad p_1 = 2 = 1 \cdot 2,\ \underline{2 \cdot 1},$

q_2 | $s_2 = 2 = 1 \cdot 2,\ \underline{2 \cdot 1},$ $\quad p_2 = 4 = 1 \cdot 4,\ 4 \cdot 1,\ \underline{2 \cdot 2},$

q_3 | $s_3 = 1 = \underline{1 \cdot 1},$ $\quad p_3 = 1 = \underline{1 \cdot 1},$

q_4 | $s_4 = 2 = \underline{1 \cdot 2},\ 2 \cdot 1,$ $\quad p_4 = 1 = \underline{1 \cdot 1},$

q_5 | $s_5 = 1 = \underline{1 \cdot 1},$ $\quad p_5 = 2 = \underline{1 \cdot 2},\ 2 \cdot 1,$

q_6 | $s_6 = 2 = 1 \cdot 2,\ \underline{2 \cdot 1},$ $\quad p_6 = 2 = 1 \cdot 2,\ \underline{2 \cdot 1},$

q_7 | $s_7 = 1 = \underline{1 \cdot 1},$ $\quad p_7 = 1 = \underline{1 \cdot 1},$

q_8 | $s_8 = 2 = 1 \cdot 2,\ \underline{2 \cdot 1},$ $\quad p_8 = 2 = 1 \cdot 2,\ \underline{2 \cdot 1}.$

Подчеркнутые разложения определяют следующие разбиения:

$$\pi = \{\pi_1, \pi_2\}, \qquad \rho = \{\rho_1, \rho_2, \rho_3, \rho_4\},$$

$$\pi_1 = \{q_1, q_2, q_6, q_8\}, \quad \pi_2 = \{q_3, q_4, q_5, q_7\},$$

$$\rho_1 = \{q_1, q_4\}, \quad \rho_2 = \{q_6, q_7\}, \quad \rho_3 = \{q_3, q_8\}, \quad \rho_4 = \{q_2, q_5\}.$$

Найденная пара (π, ρ) разбиений эквивалентна подстановке

$$t = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \\ q_1 & q_6 & q_8 & q_2 & q_4 & q_7 & q_3 & q_5 \end{pmatrix}$$

алфавита состояний. Применяя подстановку

$$t^{-1} = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \\ q_1 & q_4 & q_7 & q_5 & q_8 & q_2 & q_6 & q_3 \end{pmatrix},$$

обратную t, к матрице соединений R_A, получим

$$R'_A = \left\|\begin{array}{cccc:cccc} 0 & z_1/m_4 & z_2/m_3 & 0 & 0 & z_3/m_2 & z_4/m_1 & 0 \\ 0 & 0 & 0 & z_2/m_3 & 0 & 0 & 0 & z_4/m_1 \\ 0 & 0 & 0 & z_1/m_4 & 0 & 0 & 0 & z_3/m_2 \\ z_1/m_3 & 0 & 0 & 0 & z_3/m_1 & 0 & 0 & 0 \\ \hdashline 0 & z_1/m_2 & z_2/m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & z_2/m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & z_1/m_2 & 0 & 0 & 0 & 0 \\ z_1/m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right\|,$$

которая является правильной клеточной матрицей соединений. По матрице R_A строим матрицы соединений

$$R_{A_1} = \left\|\begin{array}{cc} x_1/n_2 & x_2/n_1 \\ x_1/n_1 & 0 \end{array}\right\| \quad \text{и} \quad R_{A_2} = \left\|\begin{array}{cccc} 0 & y_1/s_2 & y_2/s_1 & 0 \\ 0 & 0 & 0 & y_2/s_1 \\ 0 & 0 & 0 & y_1/s_2 \\ y_1/s_1 & 0 & 0 & 0 \end{array}\right\|,$$

определяющие автоматы

$$A_1 = (X, V, N, v_1 \in V, F(x \in X/n \in N))$$

и

$$A_2 = (Y, W, S, w_1 \in W, P(y \in Y/s \in S)),$$

показанные на рис. 3.2 и 3.3.

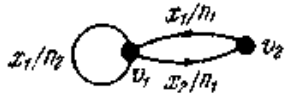


Рис. 3.2.

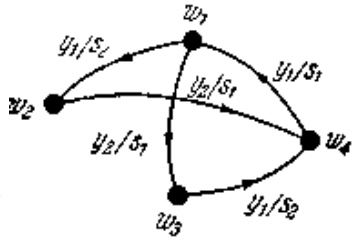


Рис. 3.3.

Приведем оценку числа автоматов, разложимых по операции умножения ×.

Пусть A — автомат с $n = k \cdot l$ состояниями, $m = p \cdot r$ входами и $t = v \cdot s$ выходами. Обозначим через $\psi_\times$ число автоматов,

представимых произведением двух автоматов A_1 и A_2 соответственно с k и l состояниями, p и r входами и v и s выходами, среди множества Ф автоматов с n состояниями, m входами и t выходами. Имеет место следующая теорема.

Теорема 3.2.

$$\frac{(k+1)^{kpv}\cdot(l+1)^{lrs}}{k!l!}\,n! < \psi_{\times} < (k+1)^{kpv}\cdot(l+1)^{lrs}\cdot n!$$

Доказательство. Определим вначале число различных автоматов с k состояниями, p входами и v выходами. Представим матрицу соединений R_1 порядка k автомата A_1 объединением автономных матриц по буквам входного алфавита

$$R_1 = \bigcup_{i \in I} R_{1i},$$

где $I = \{1, 2, \ldots, p\}$. Так как каждая автономная матрица R_{1i} в любой строке не может содержать более одной входной буквы (в противном случае происходит нарушение условия автоматности), то множество M_{1i} автоматов с k состояниями и одним входом (без учета выходных букв), включая частичные автоматы, содержит не более $(C_k^1 + 1)^k = (k+1)^k$ элементов. Очевидно, что множество M_1 различных матриц вида R_1 равно декартову произведению

$$M_1 = \underbrace{M_{1i} \times M_{1i} \times \ldots \times M_{1i}}_{pv \text{ раз}}.$$

Поэтому число ψ_{M_1} элементов множества M_1 и тем самым число автоматов с k состояниями, p входами и v выходами составляет $\psi_{M_1} = (k+1)^{kpv}$.

Аналогично находим, что число ψ_{M_2} автоматов с l состояниями, r входами и s выходами составляет

$$\psi_{M_2} = (l+1)^{lrs}.$$

Из формулы (3.1) следует, что множество M автоматов с $n = k \cdot l$ состояниями, $m = p \cdot r$ входами и $t = v \cdot s$ выходами, обладающих правильной клеточной матрицей соединений и, следовательно, представимых произведением двух автоматов, состоит из

$$\psi_M = (k+1)^{kpv} \cdot (l+1)^{lrs}$$

элементов.

Если предположить, что каждая матрица соединений из множества M является представителем класса изоморфных автоматов, то получим

$$\psi_{\times} < \psi_M \cdot n!.$$

С другой стороны, если из множества M заведомо исключить изоморфные автоматы, то получим

$$\psi_{\times} > \psi_M \cdot \frac{n!}{k!l!}.$$

Этим теорема 3.2 доказана.

Таким образом, число $\psi_{\times}$ автоматов, представимых параллельной одновременной работой двух автоматов с раздельными входами,

невелико по сравнению с общим числом $\Phi = (n+1)^{nmt}$ автоматов с n состояниями, m входами и t выходами.

3.2. Параллельная декомпозиция автоматов с общим входом

Рассмотрим задачу разложения абстрактного автомата Мили по операции умножения $\otimes$, которая соответствует представлению сложного автомата *параллельной одновременной работой более простых автоматов с одним и тем же, что и исходный автомат, входным алфавитом.* Докажем теорему, из которой следует критерий принадлежности произвольного автомата $A \in \mathfrak{A}(X)_{\otimes}$ подполугруппе $\mathbf{A}(X)_{\otimes}$ автоматов, представимых произведением двух или более автоматов из $\mathfrak{A}(X)_{\otimes}$ с общим входным алфавитом X, и сформулируем алгоритм разложения автомата $A \in \mathbf{A}(X)_{\otimes}$ по операции умножения $\otimes$. Кроме того, покажем, что любой автомат A, разложимый по операции умножения $\times$, который представим произведением двух автоматов с раздельными входами, разложим также по операции умножения $\otimes$, т. е. представим произведением двух автоматов с тем же входным алфавитом, что и автомат A.

Теорема 3.3. *Автомат A с $n = k \cdot l$ состояниями и входным алфавитом X представим произведением автоматов A_1 и A_2 соответственно с k и l состояниями и входным алфавитом X (параллельной одновременной работой двух автоматов с общим входом), если и только если существует подстановка $t \in T$ алфавита состояний, которая преобразует матрицы соединений всех автономных* (имеются в виду автономные автоматы с выходами *) автоматов A_x, $x \in X$, к виду правильных клеточных матриц соединений.*

Доказательство. Покажем необходимость существования правильных клеточных матриц для всех автономных автоматов A_x, $x \in X$, автомата A, представимого произведением двух автоматов. Действительно, если A представим произведением автоматов A_1 и A_2, то его матрица соединений R будет иметь вид

$$R = R_1 \otimes R_2 = \bigcup_{x \in X} (R_{1x} \times R_{2x}) = \bigcup_{x \in X} R_x,$$

из которого вытекает, что матрица соединений R_x каждого автономного автомата A_x является прямым произведением матриц соединений R_{1x} и R_{2x} автономных автоматов A_{1x} и A_{2x} и, следовательно, на

основании (3.1) представляет собой правильную клеточную матрицу соединений.

Покажем достаточность. Пусть все матрицы R_x автономных автоматов A_x, $x \in X$, имеют вид правильных клеточных матриц соединений или существует подстановка $t \in T$ алфавита состояний, которая приводит их к этому виду. Построим по матрице R матрицы соединений R_1 и R_2 автоматов A_1 и A_2. Для этого вначале по матрице R_x автономного автомата A_x строим матрицы R_{1x} и R_{2x} автономных автоматов A_{1x} и A_{2x} следующим образом. Матрица $R_{1x} = \|r_{ij}(x/y)\|$, где $i, j \in I = \{1, 2, \ldots, k\}$, содержит столько переходов по букве $x \in X$, сколько ненулевых клеток R_{ij} порядка l в матрице R_x, причем различным выходным буквам матрицы R_x приписываем различные выходные буквы $y \in Y$, а одинаковым — одинаковые буквы $y \in Y$ матрицы R_{1x}. Если клетка R_{ij} матрицы R_x равна нулевой клетке R_0, то элемент $r_{ij}(x/y) = 0$.

Матрица R_{2x} совпадает с любой ненулевой клеткой порядка l матрицы R_x с учетом того, что выходные буквы матрицы R_{2x} определяются так же, как и для R_{1x}. Подобным образом находим матрицы R_{1x} и R_{2x} для всех $x \in X$.

Матрицы соединений R_1 и R_2 автоматов A_1 и A_2 находим объединением полученных матриц

$$R_1 = \bigcup_{x \in X} R_{1x}, \qquad R_2 = \bigcup_{x \in X} R_{2x}.$$

Легко проверить, что $R_1 \otimes R_2 = R$ и $A_1 \otimes A_2 = A$. Этим теорема доказана.

Сформулируем теперь алгоритм представления произвольного автомата $A \in \mathfrak{A}(X)_{\otimes}$ произведением двух автоматов с общим входом.

1°. *Подсчитываем число n состояний автомата A. Если* $n = k \cdot l$, *то переходим к* 3°. *Если n — простое число, то к* 2°.

2°. *Добавляем одно изолированное состояние, чтобы выполнялось* $n = k \cdot l$. *Переходим к* 3°.

3°. *Матрицу соединений R автомата A представляем объединением матриц* R_x *автономных автоматов* A_x *по буквам входного алфавита* $x \in X$. *Переходим к* 4°.

4°. *Матрицы соединений* R_x *разбиваем на* k^2 *клеток порядка l каждая. Если каждая* R_x, $x \in X$, *— правильная клеточная матрица соединений, то переходим к* 9°. *В противном случае к* 5°.

5°. *По одной из матриц соединений* R_{x_i} *записываем матрицу смежности* R_i *графа автономного автомата* A_{x_i}, *заменяя каждую пару букв* x_i/m_i *в матрице* R_{x_i} *единицей. Переходим к* 6°.

6°. *Применяя метод разложения графов в произведение двух графов , ищем подстановку, переводящую матрицу смежности* R_i *в правильную клеточную матрицу* R'_i. *Если* $t \in T$ *—искомая подстановка, то переходим к* 7°. *Если такая подстановка не существует, то переходим к* 10°.

7°. *Применяем полученную подстановку ко всем остальным матрицам соединений* R_x. *Если каждая из матриц преобразуется к виду правильной клеточной матрицы соединений, то переходим к* 9°. *Если хотя бы одна из матриц* R_x *не преобразуется к искомому виду, то переходим к* 8°.

8°. *По матрице смежности* R_i *ищем новую подстановку* $t' \notin t \cdot \bar{T}$, *переводящую матрицу* R_i *в правильную клеточную матрицу. Если* t' *— искомая подстановка, то переходим к* 7°. *Если такой подстановки не существует, то переходим к* 10°.

9°. *По правильным клеточным матрицам соединений* R_x, $x \in X$, *строим матрицы соединений* R_1 *и* R_2 *автоматов* A_1 *и* A_2 *так, как это показано в доказательстве теоремы* 3.3.

10°. *Автомат A не разложим по операции умножения* $\otimes$.

Пример 3.2. Пусть дан автомат $A = (X, Q, M, q_1 \in Q, K(x \in X, m \in M))$, графоид которого показан на рис. 3.4.

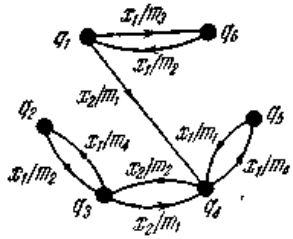


Рис 3.4.

Определить, разложим ли автомат *A* по операции умножения, и если да, то представить его произведением двух автоматов с общим входом. По графоиду запишем матрицы соединений R_{x_1} и *R* автономных автоматов A_{x_1} и A_{x_2}, которые имеют вид

$$R_{x_1} = \begin{Vmatrix} 0 & 0 & 0 & 0 & 0 & x_1/m_3 \\ 0 & 0 & x_1/m_2 & 0 & 0 & 0 \\ 0 & x_1/m_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_1/m_4 & 0 \\ 0 & 0 & 0 & x_1/m_1 & 0 & 0 \\ x_1/m_2 & 0 & 0 & 0 & 0 & 0 \end{Vmatrix},$$

$$R_{x_2} = \begin{Vmatrix} 0 & 0 & 0 & x_2/m_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2/m_1 & 0 & 0 \\ 0 & 0 & x_2/m_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{Vmatrix}$$

и не являются правильными клеточными матрицами соединений. Запишем матрицу смежности R_2 графа автономного автомата A_{x_2}

$$R_2 = \left\| \begin{array}{ccc:ccc} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \hdashline 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right\|$$

и найдем подстановку $t \in T$, приводящую R_2 к виду правильной клеточной матрицы. Выпишем полустепени исхода s и захода p каждой вершины графа автомата и образуем множества возможных разложений этих чисел на два сомножителя. Получим

$q_1 \mid s_1 = 1 = \underline{1 \cdot 1}, \quad p_1 = 0 = 0 \cdot 1, \ \underline{1 \cdot 0},$

$q_2 \mid s_2 = 0 = \underline{0 \cdot 1}, \quad p_2 = 0 = \underline{0 \cdot 1},$

$q_3 \mid s_3 = 1 = \underline{1 \cdot 1}, \quad p_3 = 1 = \underline{1 \cdot 1},$

$q_4 \mid s_4 = 1 = \underline{1 \cdot 1}, \quad p_4 = 2 = \underline{1 \cdot 2}, \ 2 \cdot 1,$

$q_5 \mid s_5 = 0 = \underline{0 \cdot 1}, \quad p_5 = 0 = \underline{0 \cdot 0},$

$q_6 \mid s_6 = 0 = \underline{1 \cdot 1}, \quad p_6 = 0 = 0 \cdot 1, \ \underline{0 \cdot 2}.$

Подчеркнутые разложения определяют следующую пару (π, ρ) разбиений множества Q:

$$\pi = \{\pi_1,\ \pi_2\}, \qquad \rho = \{\rho_1,\ \rho_2,\ \rho_3\},$$
$$\pi_1 = \{q_1,\ q_3,\ q_4\},\ \ \pi_2 = \{q_2,\ q_5,\ q_6\},$$
$$\rho_1 = \{q_1,\ q_5\}, \qquad \rho_2 = \{q_2,\ q_3\},\ \ \rho_3 = \{q_4,\ q_6\}.$$

Найденная пара (π, ρ) разбиений эквивалентна подстановке

$$t = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ q_1 & q_3 & q_4 & q_5 & q_2 & q_6 \end{pmatrix}$$

алфавита состояний. Применяя подстановку

$$t^{-1} = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ q_1 & q_5 & q_2 & q_3 & q_4 & q_6 \end{pmatrix}$$

к матрицам R_{x_1} и R_{x_2}, получим матрицы

$$R'_{x_1} = \begin{Vmatrix} 0 & 0 & 0 & 0 & 0 & x_1/m_3 \\ 0 & 0 & 0 & 0 & x_1/m_4 & 0 \\ 0 & 0 & 0 & x_1/m_4 & 0 & 0 \\ 0 & 0 & x_1/m_1 & 0 & 0 & 0 \\ 0 & x_1/m_2 & 0 & 0 & 0 & 0 \\ x_1/m_2 & 0 & 0 & 0 & 0 & 0 \end{Vmatrix},$$

$$R'_{x_2} = \begin{Vmatrix} 0 & 0 & x_2/m_1 & 0 & 0 & 0 \\ 0 & 0 & x_2/m_1 & 0 & 0 & 0 \\ 0 & x_2/m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{Vmatrix},$$

которые являются правильными клеточными матрицами соединений. По матрицам R'_{x_1} и R'_{x_2} строим матрицы соединений

$$R_{A_1} = \begin{Vmatrix} x_2/y_1 & x_1/y_2 \\ x_1/y_1 & 0 \end{Vmatrix}$$

и

$$R_{A_2} = \begin{Vmatrix} 0 & 0 & x_1/s_1 \vee x_2/s_1 \\ 0 & x_1/s_2 & x_2/s_1 \\ x_1/s_2 & x_2/s_2 & 0 \end{Vmatrix},$$

определяющие автоматы $A_1 = (X,\ V, Y, v_1 \in V, F(x \in X/y \in Y))$ и $A_2 = (X, W, S, w_1 \in W, P(x \in X/s \in S))$ с общим входным алфавитом X, показанные соответственно на рис. 3.5, *а, б.*

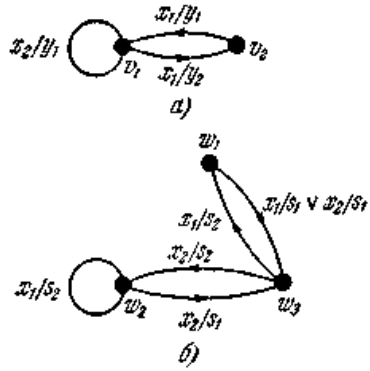


Рис. 3.5

Приведем оценку числа автоматов, представимых параллельной одновременной работой автоматов с общим входом.

Пусть A — автомат с $n = k \cdot l$ состояниями, m входами и $t = v \cdot s$ выходами. Если через $\psi_{\otimes}'$ обозначить число автоматов, разложимых по операции умножения $\otimes$, то справедлива следующая теорема.

Теорема 3.4.

$$\frac{(k+1)^{kmv} \cdot (l+1)^{lms}}{k!l!} \cdot n! < \psi_{\otimes} < (k+1)^{kmv} \cdot (l+1)^{lms} \cdot n!.$$

Доказательство этой теоремы аналогично доказательству теоремы 3.2.

Заметим, что, хотя по сравнению с общим числом автоматов с n состояниями, m входами и t выходами,число автоматов, представимых параллельной одновременной работой с общим входом, невелико, однако, как следует из теорем 3.2 и 3.4, оно в

$$(k+1)^{kmv\left(1-\frac{1}{r}\right)} \cdot (l+1)^{lms\left(1-\frac{t}{p}\right)}$$

раз больше числа автоматов, представимых параллельной одновременной работой с раздельными входами.

Покажем теперь, что любой абстрактный автомат, разложимый по операции умножения $\times$, разложим также и по операции умножения $\otimes$, иначе говоря, *параллельная декомпозиция автоматов с разделением входов является частным случаем параллельной декомпозиции автоматов с общим входом.*

Теорема 3.5. *Подполугруппа* $\mathrm{A}_{\times}$ *автоматов, разложимых по операции умножения* $\times$, *вкладывается в подполугруппы* $\mathrm{A}(Z)_{\otimes}$ *автоматов, разложимых по операции умножения* $\otimes$.

Доказательство. Пусть $A = (Z, Q, M, q_1 \in Q, K(z \in Z/m \in M))$ — автомат, который принадлежит подполугруппе $A_\times$. Тогда справедливо $A = A_1 \times A_2$,где $A_1 = (X, V, L, v_1 \in V, F(x \in X/l \in L))$ и $A_2 = (Y, W, S, w_1 \in W, P(y \in Y/s \in S))$.

Представляяавтоматы A_1 и A_2 объединением автономных автоматов по буквам входных алфавитов X и Y и учитывая дистрибутивность объединения относительно умножения автоматов, можно записать

$$A = A_1 \times A_2 = \bigcup_{x \in X} A_{1x} \times \bigcup_{y \in Y} A_{2y} = \bigcup_{(x,\ y) \in X \times Y} (A_{1x} \times A_{2y}) = \bigcup_{z \in Z} (A_{1x} \times A_{2y}). \quad \textbf{(3.2)}$$

В каждом автономном автомате A_{1x} выражения (3.2) заменим букву $x \in X$ на такую букву $z \in Z$, для которой выполняется $pr_1 z = x$. Это всегда можно сделать, так как $Z = X \times Y$, и поэтому $z = (x, y)$. Аналогично, в каждом автономном автомате A_{2y} заменим букву $y \in Y$ на такую букву $z \in Z$, что $pr_2 z = y$.Тогда получим

$$A = \bigcup_{z \in Z} (A_{1z} \times A_{2z}) = A_1' \otimes A_2',$$

где $A_1' = \bigcup_{z \in Z} A_{1z}$ и $A_2' = \bigcup_{z \in Z} A_{2z}$. Таким образом, от $A = A_1 \times A_2$ перешли к $A = A_1' \otimes A_2'$. Поэтому $A \in \mathbf{A}(Z)_\otimes$.

Этим теорема доказана.

3.3. Параллельная поочередная декомпозиция автоматов

Рассмотрим теперь *параллельную неодновременную (поочередную) декомпозицию автоматов с раздельными входами*.

Пусть $A \in \mathfrak{A}_+$ — произвольный абстрактный автомат Мили. Требуется найти критерий принадлежности автомата A подполугруппе $\mathbf{A}_+$ автоматов, представимых суммой двух или более автоматов, и сформулировать алгоритм разложения автомата $A \in \mathbf{A}_+$ в сумму двух автоматов.

Пусть A и B — произвольные абстрактные автоматы Мили, а $R_A = \|r_{\alpha\beta}(x/n)\|$ и $R_B = \|r_{\gamma\delta}(y/s)\|$ — их матрицы соединений. Матрица соединений R_C автомата С, равного сумме автоматов A и B, будет, как известно, иметь вид

$$R_C = (R_A \times E_B) \cup (E_A \times R_B) =$$

$$= \left\| \begin{array}{llll} r_{11}(x/n)\cdot E_B \cup R_B & r_{12}(x/n)\cdot E_B & \dots & r_{1k}(x/n)\cdot E_B \\ r_{21}(x/n)\cdot E_B & r_{22}(x/n)\cdot E_B \cup R_B & \dots & r_{2k}(x/n)\cdot E_B \\ \dots & \dots & \dots & \dots \\ r_{k1}(x/n)\cdot E_B & r_{k2}(x/n)\cdot E_B & \dots & r_{kk}(x/n)\cdot E_B \cup R_B \end{array} \right\|. \quad \textbf{(3.3)}$$

Из (3.3) следует, что матрица R_C может содержать следующие типы клеток $R_{\alpha\beta}$:

1) если $\alpha \neq \beta$, то

$$R_{\alpha\beta} = r_{\alpha\beta}(x_i/n_i)\cdot E_B = \left\| \begin{array}{llll} x_i/n_i & 0 & \dots & 0 \\ 0 & x_i/n_i & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & x_i/n_i \end{array} \right\|$$

при $r_{\alpha\beta}(x_i/n_i) = x_i/n_i$ и

$$R_{\alpha\beta} = E_0 \text{ при } r_{\alpha\beta}(x_i/n_i) = 0,$$

где E_0 —нулевая матрица порядка l;

2) если $\alpha = \beta$, то

$$R_{\alpha\beta} = r_{\alpha\beta}(x_j/n_j)\cdot E_B \cup R_B =$$

$$= \left\| \begin{array}{llll} x_j/n_j \vee r_{11}(y/s) & r_{12}(y/s) & \dots & r_{1l}(y/s) \\ r_{21}(y/s) & x_j/n_j \vee r_{22}(y/s) & \dots & r_{2l}(y/s) \\ \dots & \dots & \dots & \dots \\ r_{l1}(y/s) & r_{l2}(y/s) & \dots & x_j/n_j \vee r_{ll}(y/s) \end{array} \right\|$$

при $r_{\alpha\beta}(x_j/n_j) = x_j/n_j$ и

$$R_{\alpha\beta} = R_B \text{ при } r_{\alpha\beta}(x_j/n_j) = 0.$$

Матрицу соединений вида (3.3) будем называть *регулярной клеточной матрицей соединений.*

Очевидно, что, заменяя каждую букву входного алфавита в регулярной клеточной матрице соединений единицей, а дизъюнкцию букв — суммой единиц, получим матрицу смежности R_C автомата С, которая является регулярной клеточной матрицей.

Сформулируем теперь теорему, из которой следует критерий принадлежности автомата $A \in \mathfrak{A}_+$ подполугруппе $\mathbf{A}_+$, причем каждый автомат из разложим $\mathbf{A}_+$ в сумму, по крайней мере, двух автоматов из $\mathfrak{A}_+$.

Теорема 3.6. *Автомат A с $n = k \cdot l$ состояниями представим суммой двух автоматов* A_1 *и* A_2 *соответственно с k и l состояниями (параллельной поочередной работой двух автоматов), если и только если существует подстановка $t \in T$ алфавита*

состояний, которая преобразует матрицу соединений R автомата A к виду регулярной клеточной матрицы соединений.

Доказательство нетрудно провести по аналогии с доказательством теоремы 3.1.

Следует отметить, что замена дизъюнкции букв входного алфавита суммой единиц при переходе от матрицы соединений к матрице смежности исключает неоднозначность разложения автомата в сумму двух автоматов. Поэтому, если найденная подстановка $t \in T$ алфавита Q преобразует матрицу соединений R автомата A в регулярную клеточную матрицу соединений, то из (3.3) вытекает, что множество входных (выходных) букв алфавита $Z(M)$ разбивается на два непересекающихся подмножества Z_1 и Z_2 (M_1 и M_2), каждому из которых можно поставить во взаимно однозначное соответствие входные (выходные) алфавиты $X(N)$ и $Y(S)$ автоматов A_1 и A_2.

Для автоматов, разложимых по операции суммирования, как и для графов, справедливо следующее утверждение: число мест, отличных от нуля, в каждой строке и в каждом столбце матрицы соединений не превышает $k + l — 1$.

Сформулируем алгоритм представления автомата $A \in \mathfrak{A}_+$ параллельной поочередной работой двух автоматов с раздельными входами.

1°. *Подсчитываем число n состояний автомата A. Если $n = k \cdot l$, то переходим к* 2°. *Если n — простое число, то к* 8°.

2°. *Матрицу соединений R_A автомата A разбиваем на k^2 клеток порядка l каждая. Если R_A — регулярная клеточная матрица соединений, то переходим к* 7°. *В противном случае к* 3°.

3°. *Проверяем критерий о числе мест, отличных от нуля, в матрице соединений R_A. Если критерий выполняется, то переходим к* 4°, *в противном случае к* 8°.

4°. *По матрице соединений R_A строим матрицу смежности R автомата A. Переходим к* 5°.

5°. *Применяя алгоритм разложения графа по операции суммирования , ищем подстановку, переводящую матрицу смежности R в регулярную клеточную матрицу. Если $t \in T$ — искомая подстановка, то переходим к* 6°. *Если указанной подстановки не существует, то переходим к* 8°.

6°. *Применяем полученную подстановку $t \in T$ к матрице соединений R_A автомата A. Если приходим к регулярной клеточной матрице соединений, то переходим к 7°. В противном случае к* 8°.

7°. *По регулярной клеточной матрице соединений строим матрицы соединений R_{A_1} и R_{A_2} автоматов A_1 и A_2 подобно тому, как это сделано в определении регулярной клеточной матрицы соединений.*

8°. *Автомат A не разложим по операции суммирования.*

Пример 3.3. Пусть дан автомат $A = (Z, Q, M, q_1 \in Q, K(z \in Z/m \in M))$, графоид которого показан на рис. 3.6.

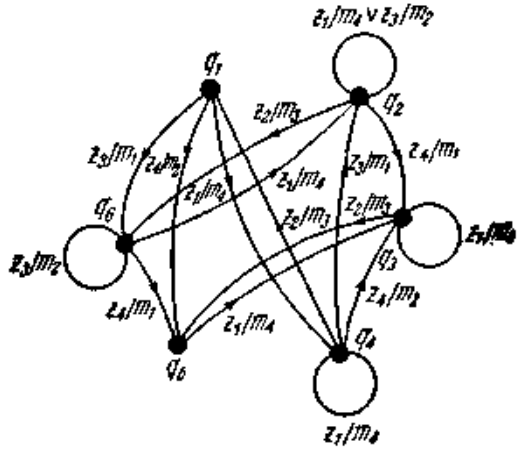


Рис. 3.6

Матрица соединений R_A автомата A имеет вид

$$R_A = \left\| \begin{array}{ccc:ccc} 0 & 0 & 0 & z_1/m_4 & z_4/m_2 & z_3/m_1 \\ 0 & z_1/m_4 \vee z_3/m_2 & z_4/m_1 & 0 & 0 & z_2/m_3 \\ 0 & 0 & z_1/m_4 & 0 & z_2/m_3 & 0 \\ \hdashline z_2/m_3 & z_3/m_1 & z_4/m_2 & z_1/m_4 & 0 & 0 \\ 0 & 0 & z_1/m_4 & 0 & 0 & 0 \\ 0 & z_1/m_4 & 0 & 0 & z_4/m_1 & z_3/m_2 \end{array} \right\|,$$

и не является регулярной клеточной матрицей соединений. Запишем матрицу смежности R, соответствующую R_A:

$$R = \left\| \begin{array}{ccc:ccc} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ \hdashline 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{array} \right\|.$$

Используя алгоритм разложения графа по операции суммирования, находим пару (π, ρ) разбиений множества Q:

$$
\begin{array}{l|l}
q_1 & s_1 = 3 = 0+3,\ 3+0,\ \underline{1+2},\ 2+1,\\
q_2 & s_2 = 4 = 0+4,\ 4+0,\ 1+3,\ 3+1,\ \underline{2+2},\\
q_3 & s_3 = 2 = 0+2,\ \underline{2+0},\ 1+1,\\
q_4 & s_4 = 4 = 0+4,\ 4+0,\ 1+3,\ 3+1,\ \underline{2+2},\\
q_5 & s_5 = 1 = 0+1,\ \underline{1+0},\\
q_6 & s_6 = 3 = 0+3,\ 3+0,\ \underline{1+2},\ 2+1,
\end{array}
$$

$$
\begin{array}{l}
p_1 = 1 = 0+1,\ \underline{1+0},\\
p_2 = 4 = 0+4,\ 4+0,\ 1+3,\ 3+1,\ \underline{2+2},\\
p_3 = 4 = 0+4,\ 4+0,\ 1+3,\ 3+1,\ \underline{2+2},\\
p_4 = 2 = 0+2,\ \underline{2+0},\ 1+1,\\
p_5 = 3 = 0+3,\ 3+0,\ \underline{1+2},\ 2+1,\\
p_6 = 3 = 0+3,\ 3+0,\ \underline{1+2},\ 2+1.
\end{array}
$$

Пара разбиений множества Q имеет вид

$$\pi_1 = \{q_1,\ q_5,\ q_6\},\quad \pi_2 = \{q_2,\ q_3,\ q_4\},$$

$$\rho_1 = \{q_1,\ q_4\},\quad \rho_2 = \{q_2,\ q_6\},\quad \rho_3 = \{q_3,\ q_5\}.$$

Полученная пара $(\pi,\ \rho)$ разбиений эквивалентна подстановке

$$t = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ q_1 & q_6 & q_5 & q_4 & q_2 & q_3 \end{pmatrix}.$$

Применяя подстановку

$$t^{-1} = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ q_1 & q_5 & q_6 & q_4 & q_3 & q_2 \end{pmatrix}$$

к матрице соединений R_A, приходим к регулярной клеточной матрице соединений

$$R'_A = \left\| \begin{array}{ccc:ccc} 0 & z_3/m_1 & z_4/m_2 & z_1/m_4 & 0 & 0 \\ 0 & z_3/m_2 & z_4/m_1 & 0 & z_1/m_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & z_1/m_4 \\ \hdashline z_2/m_3 & 0 & 0 & z_1/m_4 & z_3/m_1 & z_4/m_2 \\ 0 & z_2/m_3 & 0 & 0 & z_1/m_4 \vee z_3/m_2 & z_4/m_1 \\ 0 & 0 & z_2/m_3 & 0 & 0 & z_1/m_4 \end{array} \right\|.$$

По матрице R'_A строим матрицы соединений

$$R_{A_1} = \left\| \begin{array}{cc} 0 & z_1/m_4 \\ z_2/m_3 & z_1/m_4 \end{array} \right\| \quad \text{и} \quad R_{A_2} = \left\| \begin{array}{ccc} 0 & z_3/m_1 & z_4/m_2 \\ 0 & z_3/m_2 & z_4/m_1 \\ 0 & 0 & 0 \end{array} \right\|.$$

Обозначая состояния через *V* и *W*, входные алфавиты через $X = \{x_1, x_2\}$ и $Y = \{y_1, y_2\}$, выходные алфавиты через $N = \{n_1, n_2\}$ и $S = \{s_1, s_2\}$, получим автоматы A_1 и A_2, которые показаны на рис. 3.7, *а, б.*

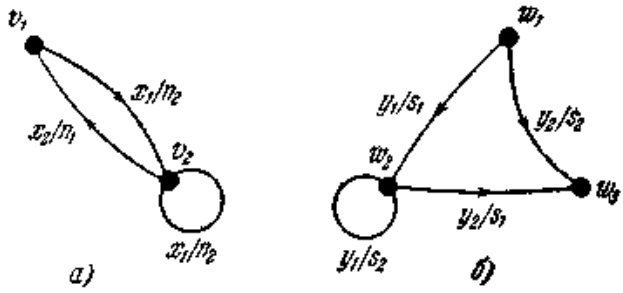


Рис. 3 7.

Очевидно, что

$$A_1 + A_2 \sim A.$$

Приведем оценку числа автоматов, представимых параллельной поочередной работой двух автоматов с раздельными входами. Пусть *A* — автомат с $n = k \cdot l$ состояниями, $m = s + t$ входами и $p = r + v$ выходами. Обозначим через Ψ_+ число автоматов, представимых суммой двух автоматов A_1 и A_2 соответственно с *k* и *l* состояниями, s и *t* входами и *r* и *v* выходами среди общего числа $(n+1)^{nmp}$ автоматов с *n* состояниями, *m* входами и *p* выходами. Тогда справедлива следующая теорема

Теорема 3.7.

$$\frac{(k+1)^{ksr} \cdot (l+1)^{ltv}}{k!l!} \cdot n! < \Psi_+ < (k+1)^{ksr} \cdot (l+1)^{ltv} \cdot n!$$

Доказательство легко провести, основываясь на теореме 3.2.

Покажем теперь, что любой абстрактный автомат, разложимый по операции суммирования, разложим также и по операции умножения $\otimes$, т. е. *параллельная поочередная декомпозиция автоматов с раздельными входами является частным случаем параллельной декомпозиции автоматов с общим входом.*

Теорема 3.8. *Подполугруппа* A_+ *автоматов, разложимых по операции суммирования, вкладывается в подполугруппы* $A(Z)_{\otimes}$ *автоматов, разложимых по операции умножения* $\otimes$.

Доказательство. Пусть

$$A = (Z, Q, M, q_1 \in Q, \quad K(z \in Z / m \in M))$$

— автомат, принадлежащий подполугруппе A_+. Тогда имеет место $A = A_1 + A_2$, где $A_1 = (X, V, N, v_1 \in V, F(x \in X/n \in N))$ и $A_2 = (Y, W, S, w_1 \in W, P(y \in Y/s \in S))$.

Пусть также R, R_1, R_2—матрицы соединений автоматов A, A_1, A_2. Запишем сумму двух автоматов в матричной форме

$$R = (R_1 \times E_2) \cup (E_1 \times R_2),$$

где E_1 и E_2—единичные матрицы смежности порядка R_1 и R_2. Матрицы соединений R_1 и R_2 перепишем в виде R_{1X} и R_{2Y} с учетом входных алфавитов автоматов A_1 и A_2.

Так как диагональные элементы матриц смежности E_1 и E_2 равны единице, то вместо каждого единичного элемента матрицы E_1 можно записать дизъюнкцию всех букв входного и выходного алфавитов Y/S, а вместо каждого единичного элемента матрицы E_2 можно записать дизъюнкцию всех букв входного и выходного алфавитов X/N. В результате получим матрицы соединений E_{1Y} и E_{2X}, которые соответственно задают автоматы с тождественными переходами по всем буквам входного алфавита Y и X. Тогда запишем

$$R = (R_{1X} \times E_{2X}) \cup (E_{1Y} \times R_{2Y}).$$

Поскольку автоматы, задаваемые матрицами соединений в круглых скобках этого выражения, имеют одинаковые входные алфавиты, операцию умножения $\times$ можно заменить операцией умножения $\otimes$. Поэтому

$$R = (R_{1X} \otimes E_{2X}) \cup (E_{1Y} \otimes R_{2Y}) = \bigcup_{x \in X} (R_{1x} \times E_{2x}) \cup \bigcup_{y \in Y} (E_{1y} \times R_{2y}).$$

Учитывая, что $X \cup Y = Z$ (так же как $N \cup S = M$), и обозначая $R_{1x} \cup E_{1y} = R'_{1z}$ и $E_{2x} \cup R_{2y} = R'_{2z}$, окончательно приходим к выражению

$$R = \bigcup_{z \in Z} ((R_{1x} \cup E_{1y}) \times (E_{2x} \cup R_{2y})) = \bigcup_{z \in Z} (R'_{1z} \times R'_{2z}) = R'_1 \otimes R'_2.$$

Таким образом, от $A = A_1 + A_2$ перешли к $A = A'_2 \otimes A'_2$. Поэтому $A \in A(Z)_{\otimes}$.

Автоматы A'_2 и A'_2, задаваемые матрицами соединений R'_1 и R'_2, определяются следующим образом:

$$R'_1 = R_{1X} \cup E_{1Y},$$

где R_{1X} — матрица соединений автомата A_1, а $E_{1Y} = \|r_{\alpha\beta}(y/s)\|$, причем

$$r_{\alpha\beta}(y/s) = \begin{cases} Y/S, & \text{если } \alpha = \beta; \\ 0, & \text{если } \alpha \neq \beta, \end{cases}$$

— матрица соединений такого же порядка, что и R_1, которая соответствует автомату с тождественными переходами по буквам входного алфавита $y \in Y$ автомата A_2.

Аналогично

$$R_2' = R_{2Y} \cup E_{2X},$$

где R_{2Y}— матрица соединений автомата A_2, а $E_{2X} = \|r_{\gamma\delta}(x/n)\|$, причем

$$r_{\gamma\delta}(x/n) = \begin{cases} X/N, & \text{если } \gamma = \delta, \\ 0, & \text{если } \gamma \neq \delta, \end{cases}$$

— матрица соединений такого же порядка, что и R_2, которая соответствует автомату с тождественными переходами по буквам входного алфавита $x \in X$автомата А. Этим теорема доказана.

3.4. Последовательная декомпозиция автоматов

В этом параграфе доказывается теорема, из которой следует критерий принадлежности автомата $A \in \mathfrak{B}_*$ подполугруппе $\mathbf{B}_*$ автоматов, разложимых по операции суперпозиции. Приводится оценка числа автоматов, представимых последовательной работой двух автоматов. Кроме того, будет показано, что различные виды параллельной совместной работы автоматов, описанные выше, можно свести, в конечном счете, к последовательной работе автоматов. Для простоты при доказательстве теоремы разложения автоматов по операции суперпозиции рассматриваются абстрактные автоматы без выходов. Поэтому при последовательном соединении двух автоматов считаем, что входной алфавит второго автомата совпадает с алфавитом состояний первого автомата.

Пусть $A = (X, Y, y_1 \in Y, F(x \in X))$ и $B = (Y, V, v_1 \in V, P(y \in Y))$ — некоторые автоматы из $\mathfrak{B}_*$, а

$$R_A = \|r_{\alpha\beta}(x)\|,$$

где

$$r_{\alpha\beta}(x) = \begin{cases} x, & \text{если } y_\beta \in Fy_\alpha \text{ по букве } x \in X, \\ 0, & \text{если } y_\beta \notin Fy_\alpha, \ \alpha, \beta \in I = \{1, 2, \ldots, k\}, \end{cases}$$

и

$$R_B = \|r_{\gamma\delta}(y)\|,$$

где

$$r_{\gamma\delta}(y) = \begin{cases} y, & \text{если } v_\delta \in Pv_\gamma \text{ по букве } y \in Y, \\ 0, & \text{если } v_\delta \notin Pv_\gamma, \ \gamma, \delta \in J = \{1, 2, \ldots, l\}, \end{cases}$$

— соответственно их матрицы соединений.

Матрицу соединений R_C автомата $C = A * B$ можно записать в виде

$$R_C = R_A * R_B = \begin{Vmatrix} r_{11}(x) \cdot R_{y_1} & r_{12}(x) \cdot R_{y_2} \ldots & r_{1k}(x) \cdot R_{y_k} \\ r_{21}(x) \cdot R_{y_1} & r_{22}(x) \cdot R_{y_2} \ldots & r_{2k}(x) \cdot R_{y_k} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ r_{k1}(x) \cdot R_{y_1} & r_{k2}(x) \cdot R_{y_2} \ldots & r_{kk}(x) \cdot R_{y_k} \end{Vmatrix} =$$

$$= \begin{Vmatrix} R_{11}(x) & R_{12}(x) \ldots & R_{1k}(x) \\ R_{21}(x) & R_{22}(x) \ldots & R_{2k}(x) \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ R_{k1}(x) & R_{k2}(x) \ldots & R_{kk}(x) \end{Vmatrix}, \qquad \textbf{(3.4)}$$

где R_{y_i}, $i \in I$,— матрица смежности автомата B по букве y_i.

Из (3.4) следует, что матрица R_C содержит k^2 клеток $R_{\alpha\beta}(x)$, $\alpha, \beta \in I$, таких, что любые две ненулевые клетки $R_{\nu\beta}(x)$ и $R_{\mu\beta}(x)$, $\nu, \mu \in I$, с одинаковым индексом β либо равны между собой, если $r_{\nu\beta}(x) = r_{\mu\beta}(x)$, либо между ними можно установить взаимно однозначное соответствие. Таким образом, матрица R_C может содержать не более k различных клеток порядка l с точностью до установления взаимно однозначного соответствия. Поэтому матрицу вида (3.4) будем называть *k-правильной клеточной матрицей соединений.*

Теорема 3.9. *Автомат A представим последовательной работой двух автоматов* A_1 *и* A_2*, если и только если существует подстановка* $t \in T$, *переводящая матрицу соединений R автомата A в k-правильную клеточную матрицу соединений.*

Доказательство необходимости вытекает из определения *k*-правильной клеточной матрицы соединений автомата.

Достаточность. Предположим, что матрица соединений R автомата A является *k*-правильной клеточной матрицей. Тогда матрицы соединений R_1 и R_2 автоматов A_1 и A_2 построим следующим образом. Матрица R_1 порядка k имеет столько переходов, сколько содержится ненулевых клеток $R_{\alpha\beta}(x)$ в матрице R, причем переходу, соответствующему какой-либо клетке $R_{\alpha\beta}(x)$, приписываем ту букву $x \in X$, по которой выполняются переходы в этой клетке. Алфавит состояний автомата A_1 обозначим через $Y = \{y_i\}$, где $i \in I = \{1, 2, \ldots, k\}$, а $y_1 \in Y$— начальное состояние A_1, причем алфавит Y является входным алфавитом автомата A_2. Объединяя все

ненулевые клетки $R_{\alpha\beta}(x)$ по соответствующим буквам входного алфавита $u \in Y$, получим матрицу соединений R_2 автомата A_2, алфавит состояний которого обозначим через $V = \{v_j\}$, где $j \in J = \{1, 2, \ldots, l\}$, причем $v_1 \in V$ — начальное состояние A_2. Нетрудно проверить, что $A_1 * A_2 = A$. Поэтому теорема доказана.

Алгоритм разложения автоматов по операции суперпозиции аналогичен алгоритму, приведенному для графов.

Приведем теперь оценку числа автоматов без выходов, которые могут быть представлены суперпозицией двух автоматов.

Пусть A — автомат с $n = k \cdot l$ состояниями и m входами.

Обозначим через ψ_* число автоматов, представимых суперпозицией двух автоматов из общего числа автоматов с n состояниями и m входами. Тогда имеет место следующая теорема.

Теорема 3.10.

$$\frac{2^{k^2 m} \cdot (l+1)^n}{k! l!} \cdot n! < \psi_* < 2^{k^2 m} \cdot (l+1)^n \cdot n!$$

Из теоремы 3.10 следует, что число ψ_* автоматов, представимых последовательной работой двух автоматов, невелико по сравнению с множеством Φ автоматов с n состояниями и m входами, которое составляет $\Phi = (n+1)^{nm}$.

Покажем теперь, что *различные виды параллельной работы автоматов можно свести, в конечном итоге, к последовательной работе автоматов.* Точнее говоря, докажем ряд теорем, из которых вытекает, что любой автомат, представимый параллельной одновременной работой двух автоматов с раздельными или общим входами, а также любой автомат, представимый параллельной поочередной работой двух автоматов, либо совместной работой двух автоматов, представим последовательной работой двух автоматов, отличных, в общем случае, от автоматов сомножителей. Этот результат играет важную роль при решении задачи декомпозиции сложного автомата на заданные стандартные автоматы, которая рассматривается в последнем параграфе этой главы. При доказательстве теорем будем иметь дело с автоматами Мили.

Теорема 3.11. *Подполугруппа* $A_\times$ *автоматов, разложимых по операции умножения* $\times$, *вкладывается в подполугруппу* B^* *автоматов, разложимых по операции суперпозиции.*

Доказательство. Пусть $A = (Z, Q, M, q_1 \in Q, K(z \in Z / m \in M))$ — некоторый автомат Мили и $A \in A_\times$. Тогда имеет место $A = A_1 \times A_2$, где $A_1 = (X, V, L, v_1 \in V, F(x \in X / l \in L))$ и $A_2 = (Y, W, U, w_1 \in W, P(y \in Y / u \in U))$.

Представим автомат A_i объединением автономных автоматов по буквам выходного алфавита $l \in L$:

$$A_1 = \bigcup_{l \in L} A_{1l},$$

а автомат A_2 объединением автономных автоматов по буквам входного алфавита $y \in Y$:

$$A_2 = \bigcup_{y \in Y} A_{2y}.$$

Тогда

$$A = \bigcup_{l \in L} A_{1l} \times \bigcup_{y \in Y} A_{2y}.$$

Учитывая, что $A_{1l} \times E_l = A_{1l}$ и $A_{2y} \times E_y = A_{2y}$ (с точностью до изоморфизма автоматов), где E_l и E_y — единичные автоматы полугруппы $\mathfrak{A}_\times$ по буквам $l \in L$ и $y \in Y$, а также дистрибутивность умножения относительно объединения автоматов и, наконец, коммутативность умножения автоматов, можно записать

$$A = \left(\bigcup_{l \in L} (A_{1l} \times E_l)\right) \times \left(\bigcup_{y \in Y} (A_{2y} \times E_y)\right) =$$

$$= \bigcup_{\substack{l \in L \\ y \in Y}} (A_{1l} \times E_l \times A_{2y} \times E_y) = \bigcup_{\substack{l \in L \\ y \in Y}} (A_{1l} \times E_y \times E_l \times A_{2y}).$$

Обозначим $L \times Y = S$. Тогда, учитывая, что $A_{1l} \times \times E_y = A_{1s}$ и $E_l \times A_{2y} = A_{2s}$, в итоге получим выражение

$$A = \bigcup_{s \in S} (A_{1s} \times A_{2s}),$$

которое определяет суперпозицию двух автоматов. Поэтому можно записать $A = A'_1 * A'_2$, откуда следует, что $A \in \mathfrak{B}_*$. Этим теорема 3.11 доказана.

Легко видеть, что автоматы A'_1 и A'_2 определяются по формулам

$$A'_1 = \bigcup_{s \in S} A_{1s} = \bigcup_{\substack{l \in L \\ y \in Y}} (A_{1l} \times E_y) = \bigcup_{l \in L} A_{1l} \times \bigcup_{y \in Y} E_y = A_1 \times E_Y,$$

где E_Y — единичный автомат, который выполняет роль единицы в полугруппе $\mathfrak{B}_*$ по алфавиту Y, а

$$A'_2 = \bigcup_{s \in S} A_{2s} = \bigcup_{\substack{l \in L \\ y \in Y}} (E_l \times A_{2y}) = \bigcup_{l \in L} E_l \times \bigcup_{y \in Y} A_{2y} = E_L \times A_2,$$

где E_L — единичный автомат, который выполняет роль единицы в полугруппе $\mathfrak{B}_*$ по алфавиту L.

Теорема 3 12. *Подполугруппы* $\mathbf{A}(X)_{\otimes}$ *автоматов, разложимых по операции умножения* $\otimes$, *вкладываются в подполугруппу* B* *автоматов, разложимых по операции суперпозиции.*

Доказательство. Пусть $A \in \mathbf{A}(X)$, и следовательно, имеет место $A = A_1 \otimes A_2$, где $A = (X, Q, M, q_1 \in$

$\in Q, K(x \in X/m \in M))$, $A_1 = (X, V, L, v_1 \in V, F(x \in X /$
$/l \in L)$ и $A_2 = (X, W, U, w_1 \in W, P(x \in X/u \in U))$

Представляя автоматы A_1 и A_2 соответственно объединением автономных автоматов по буквам выходного и входного алфавитов, запишем

$$A = \left(\bigcup_{l \in L} A_{1l}\right) \otimes \left(\bigcup_{x \in X} A_{2x}\right) =$$

$$= \left(\bigcup_{l \in L} (A_{1l} \times E_l)\right) \otimes \left(\bigcup_{x \in X} (A_{2x} \times E_x)\right).$$

Переходя от умножения $\otimes$ ® к умножению $\times$, получим

$$A = \bigcup_{x \in X} \left(\left(\bigcup_{l \in L} (A_{1lx} \times E_{lx})\right) \times \left(\bigcup_{x \in X} (A_{2x} \times E_x)\right)\right).$$

Так как E_l не зависит от букв $x \in X$, то $E_{lx} = E_l$. Учитывая дистрибутивность умножения относительно объединения и коммутативность умножения автоматов, запишем

$$A = \bigcup_{x \in X} \left(\bigcup_{\substack{l \in L \\ x \in X}} (A_{1lx} \times E_x \times E_l \times A_{2x})\right).$$

Вводя обозначения $L \times X = S$, $A_{1lx} \times E_x = A_{1sx}$, $E_l \times$ $\times A_{2x} = A_{2s}$ и учитывая коммутативность объединения автоматов, приходим к выражению

$$A = \bigcup_{s \in S} \left(\bigcup_{x \in X} (A_{1sx} \times A_{2s})\right) = \bigcup_{s \in S} \left(\left(\bigcup_{x \in X} A_{1sx}\right) \times A_{2s}\right) =$$

$$= \bigcup_{s \in S} (A_{1s} \times A_{2s}),$$

которое соответствует суперпозиции двух автоматов. Поэтому $A = A'_1 * A'_2$ и $A \in \mathbf{B}_*$. Этим теорема 3.12 доказана.

Здесь автомат A'_1 определяется выражением

$$A'_1 = \bigcup_{x \in X} \left(\bigcup_{\substack{l \in L \\ x \in X}} (A_{1lx} \times E_x)\right) = \bigcup_{\substack{l \in L \\ x \in X}} \left(\bigcup_{x \in X} (A_{1lx} \times E_x)\right) =$$

$$= \bigcup_{\substack{l \in L \\ x \in X}} (A_{1l} \otimes E_x) = \left(\bigcup_{l \in L} A_{1l}\right) \otimes E_X = A_1 \otimes E_X,$$

где E_X — автомат, выполняющий роль единицы полугруппы $\mathfrak{B}_*$ по алфавиту X, а

$$A_2' = \bigcup_{x \in X} \left(\bigcup_{\substack{l \in L \\ x \in X}} (E_l \times A_{2x}) \right) = \bigcup_{\substack{l \in L \\ x \in X}} \left(\bigcup_{x \in X} (E_l \times A_{2x}) \right) =$$

$$= \bigcup_{\substack{l \in L \\ x \in X}} (E_l \times A_2) = E_L \times A_2,$$

где E_L — автомат, выполняющий роль единицы полугруппы $\mathfrak{B}_*$ по алфавиту L.

Теорема 3.13. *Подполугруппа* A_+ *автоматов, разложимых по операции суммирования, вкладывается в подполугруппу* B^* *автоматов, разложимых по операции суперпозиции.*

Доказательство. Пусть $A \in A_+$. В этом случае $A = A_1 + A_2$, где $A = (Z, Q, M, q_1 \in Q, K(z \in Z/m \in M))$, $A_1 = (X, V, L, v_1 \in V, F(x \in X/l \in L))$ и $A_2 = (Y, W, U, w_1 \in W, P(y \in Y/u \in U))$.

Представляя матрицы соединений R_1 и R_2 автоматов A_1 и A_2 объединением матриц соединений автономных автоматов, запишем

$$R = \left(\bigcup_{l \in L} R_{1l} \right) + \left(\bigcup_{y \in Y} R_{2y} \right) =$$

$$= \left(\left(\bigcup_{l \in L} R_{1l} \right) \times E_2 \right) \cup \left(E_1 \times \left(\bigcup_{y \in Y} R_{2y} \right) \right) =$$

$$= \left(\bigcup_{l \in L} (R_{1l} \times E_2) \right) \cup \left(\bigcup_{y \in Y} (E_1 \times R_{2y}) \right).$$

Вместо тождественно истинных переходов в единичных матрицах E_2 и E_1 вводим буквы соответствующих алфавитов L и Y. Учитывая, что объединение с единичными матрицами дистрибутивно относительно умножения, и обозначая $L \cup Y = S$, получим

$$R = \left(\bigcup_{l \in L} (R_{1l} \times E_{2l}) \right) \cup \left(\bigcup_{y \in Y} (E_{1y} \times R_{2y}) \right) =$$

$$= \bigcup_{s \in S} ((R_{1l} \cup E_{1y}) \times (E_{2l} \cup R_{2y})) = \bigcup_{s \in S} (R_{1s} \times R_{2s}) = R_1' * R_2'.$$

Таким образом, матрицу соединений R автомата A представили суперпозицией матриц R'_1 и R'_2 автоматов A'_1 и A'_2. Поэтому $A = A_1' * A_2'$ и $A \in B_*$. Этим теорема 3.13 доказана.

Матрицы соединений R'_1 и R'_2 автоматов A'_1 и A'_2 определяются следующим образом:

$$R_1' = \bigcup_{s \in L \cup Y} (R_{1l} \cup E_{1y}) = R_1 \cup E_{1Y} =$$

$$= (R_1 \times \{1\}) \cup (E_1 \times R_{E_Y}) = R_1 + R_{E_Y},$$

где R_{E_Y}—матрица соединений автомата E_Y, который

играет роль единицы полугруппы $\mathfrak{B}_*$ по алфавиту Y.

$$R_2' = \bigcup_{s \in L \cup Y} (E_{1l} \cup R_{2y}) = E_{2L} \cup R_2 =$$
$$= (R_{E_L} \times E_2) \cup (\{1\} \times R_2) = R_{E_L} + R_2,$$

где R_{E_L} — матрица соединений автомата E_L, который играет роль единицы полугруппы $\mathfrak{B}_*$ по алфавиту L.

Теорема 3.14. *Подполугруппа A_0 автоматов, разложимых по операции композиции, вкладывается в подполугруппу* B* *автоматов, разложимых по операции суперпозиции.*

Доказательство. Пусть $A \in A_0$. Тогда $A = A_1 \circ A_2$, где

$A = (Z, Q, M, q_1 \in Q, K(z \in Z/m \in M))$.
$A_1 = (X, V, L, v_1 \in V, F(x \in X, u \in U/l \in L))$ и $A_2 = (Y, W, U, w_1 \in W, P(y \in Y, l \in L/u \in U))$.

Представим автоматы A_1 и A_2 объединением автономных автоматов соответственно по буквам выходного $l \in L$ и входных $y \in Y$, $l \in L$ алфавитов. Запишем

$$A = \bigcup_{l \in L} A_{1l} \circ \bigcup_{\substack{y \in Y \\ l \in L}} A_{2yl} = \bigcup_{\substack{u \in U \\ l \in L}} \left(\left(\bigcup_{l \in L} A_{1l} \right)_{u/l} \times \left(\bigcup_{\substack{y \in Y \\ l \in L}} A_{2yl} \right)_{l/u} \right) =$$
$$= \bigcup_{\substack{u \in U \\ l \in L}} \left(\left(\bigcup_{l \in L} A_{1\,(u/l)} \right) \times \left(\bigcup_{\substack{y \in Y \\ l \in L}} A_{2\,(yl/u)} \right) \right) =$$
$$= \bigcup_{\substack{u \in U \\ l \in L}} \left(\left(\bigcup_{l \in L} A_{1\,(u/l)} \times E_u \times E_l \right) \times \right.$$
$$\left. \times \left(\bigcup_{\substack{y \in Y \\ l \in L}} A_{2\,(yl/u)} \times E_y \times E_l \times E_u \right) \right).$$

Учитывая дистрибутивность умножения относительно объединения автоматов и коммутативность умножения, получим

$$A = \bigcup_{\substack{u \in U \\ l \in L}} \left(\bigcup_{\substack{y \in Y \\ l \in L}} (A_{1\,(u/l)} \times E_u \times E_l \times A_{2\,(yl/u)} \times E_y \times E_l \times E_u) \right) =$$
$$= \bigcup_{\substack{u \in U \\ l \in L \\ y \in Y}} ((A_{1\,(u/l)} \times E_y \times E_u \times E_l) \times (A_{2\,(yl/u)} \times E_l \times E_u)) =$$
$$= \bigcup_{\substack{u \in U \\ l \in L \\ y \in Y}} (A_{1\,(ylu/ylu)} \times A_{2\,(ylu/ylu)}).$$

Обозначая $Y \times L \times U = S$, окончательно получаем

$$A = \bigcup_{s \in S} (A_{1s} \times A_{2s}) = A_1' * A_2'.$$

Таким образом, $A \in B_*$ и теорема 3.14 доказана.

Здесь

$$A_1' = \bigcup_{\substack{u \in U \\ l \in L \\ y \in Y}} (A_{1\,(u/l)} \times E_y \times E_u \times E_l) =$$

$$= \left(\bigcup_{\substack{u \in U \\ l \in L}} A_{1\,(u/l)} \right) \circ \left(\bigcup_{y \in Y} E_y \right) \circ \left(\bigcup_{u \in U} E_u \right) \circ \left(\bigcup_{l \in L} E_l \right) =$$

$$= A_1 \circ E_Y \circ E_U \circ E_L$$

и

$$A_2' = \bigcup_{\substack{u \in U \\ l \in L \\ y \in Y}} (A_{2\,(yl/u)} \times E_l \times E_u) =$$

$$= \left(\bigcup_{\substack{u \in U \\ l \in L \\ y \in Y}} A_{2\,(yl/u)} \right) \circ \left(\bigcup_{l \in L} E_l \right) \circ \left(\bigcup_{u \in U} E_u \right) = A_2 \circ E_L \circ E_U,$$

причем E_Y, E_L и E_U — соответственно единицы по алфавитам Y, L и U полугруппы $\mathfrak{B}_*$.

3.5. Общая декомпозиция абстрактных автоматов

Как было показано в предыдущих параграфах, число автоматов, обладающих специальными видами матриц соединений такими, например, как правильная и регулярная клеточные матрицы, невелико по сравнению с общим числом автоматов. Поэтому автоматы, представимые параллельной или последовательной работой двух или более автоматов, составляют небольшое число всега множества автоматов. В связи с этим необходимо научиться представлять *произвольные* автоматы совместной работой более простых автоматов. Такое представление сводится, по сути дела, к *разложению произвольного автомата по операции композиции автоматов.*

Основная идея излагаемого ниже метода состоит в том, что матрицы соединений каждого из автономных автоматов по входам исходного автомата приводятся к виду правильной клеточной матрицы с запрещенными переходами, представляются прямым произведением матриц сомножителей с отмеченными переходами, в которые затем вводятся необходимые зависимости от состояний соседних автоматов, устраняющие запрещенные переходы. В итоге получаем разложение произвольного автомата в совместную работу более простых автоматов

с тем же входным алфавитом, что и исходный автомат. *Представление автомата совместной работой элементарных автоматов со связями между ними назовем общей декомпозицией абстрактных автоматов.*
В качестве элементарных абстрактных автоматов могут быть выбраны автоматы с любым числом состояний (например, двумя, тремя, пятью и т. д.).
Заметим, что при разложении автомата по операции композиции ставится, как правило, задача оптимальной декомпозиции, т. е. представление произвольного автомата совместной работой элементарных абстрактных автоматов с минимальным числом связей между ними. Решение задачи оптимальной декомпозиции приводит, в конечном счете, к минимальной комбинационной части автомата на уровне функциональных схем.
Введем теперь понятие *правильной клеточной матрицы соединений с запрещенными переходами* и рассмотрим методику общей декомпозиции произвольного абстрактного автомата.
Пусть $R = \|r_{\mu\nu}(z/m)\|$, $\mu, \nu \in N = \{1, 2, \ldots, n\}$,— матрица соединений произвольного автомата Мили $A = (Z, Q, M, q_1 \in Q, K(z \in Z/m \in M))$, причем R не является ПКМС или каким-либо другим ранее определенным типом матрицы соединений.
Представим матрицу соединений R объединением матриц автономных автоматов по буквам входного алфавита $z \in Z$:

$$R = \bigcup_{z \in Z} R_z$$

и дальнейшие преобразования проводим над матрицей R_z. Если число состояний $n = k \cdot l$, то разобьем матрицу R_z на k^2 клеток порядка l. Если n — простое число, то, добавляя одно изолированное состояние, получим $n = k \cdot l$. Все ненулевые клетки порядка l объединим до универсальной клетки R_{zl} порядка l, элементы которой являются дизъюнктивным объединением пар z/m букв нходного и выходного алфавита, стоящих на соответствующих местах каждой клетки.
Каждую ненулевую клетку матрицы R_z дополним через дизъюнкцию по буквам входного и выходного алфавита до R_{zl}. В этом случае получим матрицу соединений R'_z с запрещенными переходами.
Под *запрещенными переходами,* которые будем отмечать на матрице соединений квадратиками, имеем в виду те переходы, которых не было в исходной матрице соединений R_z.
Так как R'_z является ПКМС, то на основании теоремы 3.1 представим ее произведением матриц R'_{1z} и R'_{2z} с *отмеченными* переходами следующим образом:

$$R'_z = \| r_{\mu\nu}(z/m) \| = \| r_{\alpha\beta}(z/m) \| \times \| r_{\gamma\delta}(z/m) \| = R'_{1z} \times R'_{2z},$$

где

$$\mu = (\alpha,\ \gamma),\ \nu = (\beta,\ \delta),\ \alpha,\ \beta \in I = \{1,\ 2,\ \ldots,\ k\},$$

$$\gamma,\ \delta \in J = \{1,\ 2,\ \ldots,\ l\}.$$

В матрице $R'_{1z} = \| r_{\alpha\beta}(z/m) \|$ содержится только одна входная буква $z \in Z$ (именно та, по которой выделен автономный автомат A_z), так как все клетки порядка l матрицы R'_z, отличные от нулевой, равны между собой. Аналогично для выходной буквы $m \in M$.

Поскольку для устранения запрещенных переходов в матрице R'_z будут использоваться выходные сигналы, совпадающие с состояниями памяти автоматов A_{1z} и A_{2z}, то выходную букву $m \in M$ в матрице R'_{1z} можно исключить и считать, что матрица

$$R'_{1z} = \| r_{\alpha\beta}(z) \|$$

задает автомат Мура, выходные сигналы которого совпадают с состояниями его памяти.

Для фиксирования отмеченных переходов в матрицах R'_{1z} и R'_{2z} введем k^2 различных *меток*. Если переход $r_{\mu\nu}(z/m)$ в матрице R'_z помечен квадратиком, т. е. является запрещенным по буквам $z \in Z$ и $m \in M$, то соответствующие переходы $r_{\alpha\beta}(z)$ в матрице $A_{1z}R'_{1z}$ и $r_{\gamma\delta}(z/m)$ в матрице R'_{2z} необходимо отметить одной и той же меткой. С содержательной точки зрения это означает, что в автоматах A_{1z} и A_{2z} не могут одновременно совершаться переходы, помеченные одной и той же меткой, тогда как переходы, помеченные различными метками, допустимы.

Заметим, что запрещенные переходы в матрице R'_z могут быть двух типов. Одна часть запрещенных переходов, возникающих в матрице R_z при приведении ее к правильному виду, определяется наличием связей между автоматами A_{1z} и A_{2z}, т. е. противоречивостью их работы, а другая часть запрещенных переходов возникает из-за наличия противоречий выходной (комбинационной) части автомата A_{2z}, поскольку одной и той же входной букве $z \in Z$ могут соответствовать различные выходные буквы $m_1 \in M$ и $m_2 \in M$.

Перейдем теперь от матриц R'_{1z} и R'_{2z} с отмеченными переходами к матрицам соединений R_{1z} и R_{2z}, в которых переходы зависят не только от входного алфавита Z, но и от состояний соответствующих автоматов. Эту процедуру назовем *введением связей* в матрицы соединений R'_{1z} и R'_{2z}. Обозначим состояния автомата A_{1z} через $V = \{v_1,\ v_2,\ \ldots,\ v_k\}$, а состояния автомата A_{2z} через $W = \{w_1, w_2, \ldots, w_l\}$. Пусть $r_{\alpha\beta}(z)$ — отмеченный переход в матрице R'_{1z}. Выбираем в матрице R'_{2z} все те переходы $r_{\gamma\delta}(z/m)$, которые не

помечены меткой перехода $r_{\alpha\beta}(z)$. Так как переход $r_{\alpha\beta}(z)$ возможен одновременно только с выбранными переходами $r_{\gamma\delta}(z/m)$, то вместо перехода $r_{\alpha\beta}(z)$ в матрицу A_{1z} вводим дизъюнкцию конъюнктивных членов соответствующих состояний $w \in W$ и входных букв $z \in Z$ указанных переходов $r_{\gamma\delta}(z/m)$ матрицы R'_{2z}.

Далее, если $r_{\gamma\delta}(z/m)$— отмеченный переход в матрице R'_{2z}, то вместо входной буквы $z \in Z$ указанного перехода матрицы R'_{2z} вводим дизъюнкцию конъюнктивных членов соответствующих состояний $v \in V$ и входной буквы $z \in Z$ переходов $r_{\alpha\beta}(z)$ матрицы R'_{1z}, не помеченных меткой перехода $r_{\gamma\delta}(z/m)$. После выполнения описанной процедуры получаем матрицы соединений $R_{1z} = \|r_{\alpha\beta}(wz)\|$ и $R_{2z} = \|r_{\gamma\delta}(vz/m)\|$ автономных автоматов A_{1z} и A_{2z}.

Аналогично поступаем с каждым автономным автоматом $A_z(z \in Z)$. В результате объединения автономных матриц соединений R_{1z} и R_{2z} получаем матрицы соединений

$$R_1 = \bigcup_{z \in Z} R_{1z}, \qquad R_2 = \bigcup_{z \in Z} R_{2z},$$

определяющие автоматы $A_1 = (Z,\ V,\ v_1 \in V,\ F(w \in W,\ z \in Z/v \in V))$ и $A_2 = (Z,\ W,\ M,\ w_1 \in W,\ P(v \in V,\ z \in Z/m \in M))$, первый из которых A_1 является автоматом Мура, а второй A_2— автоматом Мили. Легко показать, что $R_1 \circ R_2 = R$, т. е. $A_1 \circ A_2 = A$.

Учитывая приведенные выше построения, можно утверждать, что справедливо следующее предложение.

Любой автомат Мили с числом состояний n > 2 можно представить совместной работой (композицией) двух или большего числа простых автоматов, один из которых является автоматом Мили, а остальные —автоматами Мура.

Покажем методику общей декомпозиции автомата на примере.

Пример 3.4. Пусть дан автомат Мили *A,* показанный на рис. 3.8. Представить его композицией двух автоматов A_1 и A_2.

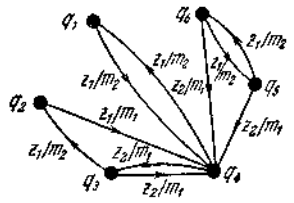


Рис. 3 8.

Матрица соединений R автомата A имеет вид

$$R = \begin{Vmatrix} 0 & 0 & 0 & z_1/m_2 & 0 & 0 \\ 0 & 0 & 0 & z_1/m_1 & 0 & 0 \\ 0 & z_1/m_2 & 0 & z_2/m_1 & 0 & 0 \\ z_1/m_2 & 0 & z_2/m_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & z_2/m_1 & 0 & z_1/m_2 \\ 0 & 0 & 0 & z_2/m_1 & z_1/m_2 & 0 \end{Vmatrix}.$$

Приводя матрицы соединений R_{Z1} и R_{Z2} к виду правильных клеточных матриц соединений с запрещенными переходами, получим матрицы

$$R'_{z_1} = \left\| \begin{array}{ccc:ccc} \boxed{z_1/m_2} & 0 & 0 & z_1/m_2 & 0 & 0 \\ \boxed{z_1/m_2} & 0 & \boxed{z_1/m_2} & z_1/m_2 & 0 & \boxed{z_1/m_2} \\ 0 & z_1/m_2 & 0 & 0 & \boxed{z_1/m_2} & 0 \\ \hdashline z_1/m_2 & 0 & 0 & \boxed{z_1/m_2} & 0 & 0 \\ \boxed{z_1/m_2} & 0 & \boxed{z_1/m_2} & \boxed{z_1/m_2} & 0 & z_1/m_2 \\ 0 & \boxed{z_1/m_2} & 0 & 0 & z_1/m_2 & 0 \end{array} \right\|,$$

$$R'_{z_2} = \left\| \begin{array}{ccc:ccc} 0 & 0 & 0 & 0 & 0 & \boxed{z_2/m_1} \\ 0 & 0 & 0 & \boxed{z_2/m_1} & 0 & 0 \\ 0 & 0 & 0 & z_2/m_1 & 0 & 0 \\ \hdashline 0 & 0 & z_2/m_1 & 0 & 0 & \boxed{z_2/m_1} \\ \boxed{z_2/m_1} & 0 & 0 & z_2/m_1 & 0 & 0 \\ \boxed{z_2/m_1} & 0 & 0 & z_2/m_1 & 0 & 0 \end{array} \right\|,$$

которые содержат 15 запретов.

Представим матрицы R'_{z_1} и R'_{z_2} прямым произведением двух матриц и одинаковыми метками из набора (—, ~, ○, ●), отметим те переходы в матрицах сомножителях, совместная работа которых образует запрещенные переходы в матрицах R'_{z_1} и R'_{z_2}. Тогда получим

$$R'_{z_1} = R'_{1z_1} \times R'_{2z_1} = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \left\| \begin{matrix} \underline{z}_1 & \underset{\sim}{z}_1 \\ z_1 & z_1 \end{matrix} \right\| \times \begin{matrix} w_1 \\ w_2 \\ w_3 \end{matrix} \left\| \begin{matrix} \underline{z_1/m_2} & 0 & 0 \\ \underset{\sim}{z_1/m_1} & 0 & \underline{z_1/m_2} \\ 0 & \underset{\sim}{z_1/m_2} & 0 \end{matrix} \right\|,$$

$$R'_{z_2} = R'_{1z_2} \times R'_{2z_2} = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \left\| \begin{matrix} 0 & \underset{\sim}{z}_2 \\ z_2 & z_2 \end{matrix} \right\| \times \begin{matrix} w_1 \\ w_2 \\ w_3 \end{matrix} \left\| \begin{matrix} 0 & 0 & \underset{\sim}{z_2/m_1} \\ \underset{\sim}{z_2/m_1} & 0 & 0 \\ z_2/m_1 & 0 & 0 \end{matrix} \right\|.$$

Введем связи в матрицы сомножителей R'_{1z_1} и R'_{2z_1}. Для этого вместо элемента z_1 в матрице R'_{1z_1} следует записать

$$\underline{z}_1 = w_3 z_1,$$
$$\underset{\sim}{z}_1 = w_1 z_1 \vee w_2 z_1 = (w_1 \vee w_2) z_1,$$
$$z_1 = w_1 z_1,$$
$$z_1 = w_2 z_1 \vee w_3 z_1 = (w_2 \vee w_3) z_1,$$

а в матрице R'_{2z_1} соответственно

$$\underline{z}_1 = v_1 z_1 \vee v_2 z_1 = (v_1 \vee v_2) z_1 = z_1,$$
$$z_1 = v_1 z_1,$$
$$\underset{\sim}{z}_1 = v_2 z_1,$$
$$\underset{\sim}{z}_1 = v_1 z_1 \vee v_2 z_1 = (v_1 \vee v_2) z_1 = z_1,$$

так как $v_1 \vee v_2 = 1$, т. е. образует тождественно истинный набор состояний.

После введения связей в матрицы R'_{1z_1}, R'_{2z_1} и объединения соответствующих матриц автономных автоматов в итоге получаем матрицы

$$R_1 = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \left\| \begin{matrix} w_3 z_1 & (w_1 \vee w_2) z_1 \vee w_3 z_2 \\ w_1 (z_1 \vee z_2) & (w_2 \vee w_3)(z_1 \vee z_2) \end{matrix} \right\|$$

и

$$R_2 = \begin{matrix} w_1 \\ w_2 \\ w_3 \end{matrix} \left\| \begin{matrix} z_1/m_2 & 0 & v_2 z_2/m_1 \\ v_1 z_1/m_1 \vee v_2 z_2/m_1 & 0 & v_2 z_1/m_1 \\ z_2/m_1 & z_1/m_2 & 0 \end{matrix} \right\|,$$

которые определяют автоматы A_1 и A_2, показанные соответственно на рис. 3.9, *а, б.*

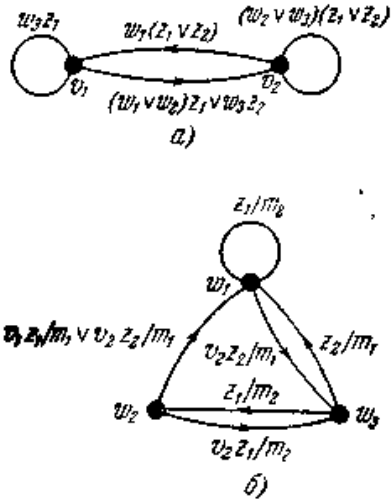


Рис. 3.9

Нами рассмотрена абстрактная декомпозиция произвольного автомата. Если исходный автомат *A* имеет ПКМС, в которой отсутствуют запрещенные переходы, то в результате декомпозиции получим автоматы A_1 и A_2, между которыми нет ни одной связи. Если при построении ПКМС в *A′* возникает хотя бы один запрещенный переход, то при декомпозиции это приведет к образованию связи между автоматами A_1 и A_2. Увеличение числа запрещенных переходов ведет к увеличению числа связей между автоматами A_1 и A_2. Возникновение новых связей, в свою очередь, приводит к появлению новых переменных в функциях возбуждения элементарных автоматов, т. е. к усложнению комбинационной части автомата. Отсюда следует, что нужно проводить декомпозицию автомата *A* на элементарные автоматы с минимальным числом связей между ними. Так как число связей зависит от числа запрещенных переходов в матрице соединений исходного автомата, то требуется таким образом изоморфно преобразовать матрицу соединений *R* автомата A, чтобы она содержала минимальное число запрещенных переходов.

Тривиальным алгоритмом нахождения ПКМС с минимальным числом запретов является алгоритм полного перебора в классе

эквивалентности по отношению изоморфизма алфавита состояний автомата. Он заключается в следующем.

После приведения всех автономных матриц R_z автомата A с n состояниями к виду ПКМС подсчитываем число запрещенных переходов, которое обозначим через φ_1. Применим произвольную подстановку $t_1 \in T$, где T — симметрическая группа подстановок алфавита состояний, к матрицам R_z и построим ПКМС R' с общим числом запретов φ_2. Сравним φ_1 и φ_2. Продолжая этот процесс, после $n!$ сравнений найдем ПКМС, число запретов в которой составляет φ_{min}. *Правильную клеточную матрицу соединений с минимальным числом запрещенных переходов* φ_{min} *будем называть квазиправильной матрицей соединений.*

Очевидно, что алгоритм полного перебора малоэффективен. Покажем, что при поиске квазиправильной матрицы соединений можно сократить перебор по сравнению с полным, если воспользоваться следующей теоремой.

Теорема 3.15. *Для того чтобы матрицу соединений R автомата A с* $n = k \cdot l$ *состояниями привести к виду квазиправильной матрицы соединений R', достаточно применить к матрице R не более* $\frac{n!}{k!\,l!} - 1$ *подстановок* $t \in T$.

Доказательство. Пусть дан произвольный автомат A с n состояниями. Представим матрицу соединений R объединением автономных матриц R_z по буквам входного алфавита $z \in Z$ и разобьем каждую из них на k^2 клеток порядка l. Если матрицы R_z не являются правильными клеточными матрицами соединений, то, применяя стандартный прием, представим каждую из них правильной клеточной матрицей соединений с числом запретов в каждой из матриц R_z, равным φ_{0z}. Ясно, что общее число φ_0 запретов в матрице R равно $\varphi_0 = \sum_{z \in Z} \varphi_{0z}$.

Разложим матрицу R в произведение матриц R_1 порядка k с числом отмеченных переходов каждой из автономных матриц R_{1z}, равным φ_{1z}, и R_2 порядка l с числом отмеченных переходов в каждой из матриц R_{2z}, равным φ_{2z}. Пусть число различных меток равно m. Тогда легко показать, что

$$\varphi_{0z} = \frac{1}{m}(\varphi_{1z} \cdot \varphi_{2z})$$

и

$$\varphi_0 = \sum_{z \in Z} \frac{1}{m}(\varphi_{1z} \cdot \varphi_{2z}).$$

Применим к любой из матриц R_{1z}, $z \in Z$, произвольную подстановку $t_1 \in T_1$, где T_1 — симметрическая группа подстановок порядка $k!$.

Очевидно, что число φ_{1z} отмеченных переходов в матрице R_{1z} остается постоянным при любом изоморфном преобразовании матрицы R_{1z}. В матрице R_z это соответствует перестановке клеток порядка l. Далее, применение к матрице R_{2z} произвольной подстановки $t_2 \in T_2$, где $|T_2| = l!$, также не изменяет числа φ_{2z} отмеченных переходов в матрице R_{2z}. На уровне R_z применение t_2 эквивалентно одновременному изоморфному преобразованию клеток порядка l.

Таким образом, любая подстановка $t = t_1 \times t_2$, т. е. $t \in \tilde{T}$, где $\tilde{T} = T_1 \times T_2$, не изменяет числа φ_{0z} запрещенных переходов в матрице R_z, а следовательно, и общего числа φ_0 запретов матрицы R. Так как $|\tilde{T}| = k!l!$, то для нахождения матрицы R' с числом запретов $\varphi_{\min}$ достаточно проверить по одной подстановке из каждого класса разложения симметрической группы T по подгруппе $\tilde{T}$, за исключением подстановок из $t_0 \cdot \tilde{T}$, где t_0 — тождественная подстановка алфавита состояний. Этим теорема 3.15 доказана.

Из теоремы 3.15 вытекает следствие.

Следствие 3.1. *Для любого автомата A с $n = k \cdot l$ состояниями существует $k!l!$ подстановок $t \in T$, которые преобразуют матрицу соединений R в квазиправильную матрицу соединений R'.*

Использование теоремы 3.15 позволяет значительно сократить перебор по сравнению с полным, однако при большом n число подстановок, подлежащих перебору, достаточно велико. Поэтому предложим эвристический прием, ведущий к существенному сокращению перебора при поиске подстановки, приводящей матрицу соединений автомата к виду квазиправильной матрицы соединений.

Определим предварительно понятия *состояний с тождественными переходами и выходами.*

Два состояния q_i и q_j - автомата A имеют *тождественные переходы по букве входного алфавита x,* если по этой букве они образуют следующие переходы:

а) q_i переходит в q_k и q_j переходит в q_k;

б) q_i переходит в q_k , а переход из q_j неопределен или наоборот;

b) переход из q_i и q_j по букве x неопределен;

г) q_i, q_j переходят в q_i, q_j или в такую пару состояний, для которых ранее были найдены тождественные переходы.

Аналогично два состояния q_i и q_j имеют *тождественные выходы по букве входного алфавита x,* если при подаче на вход автомата этой буквы на выходе его появляются следующие выходные сигналы:

а) в том случае, когда автомат находится в состоянии q_i или q_j, выходной сигнал равен y_k ;

б) в том случае, когда автомат находится в состоянии q_i, выходной сигнал равен y_k , а когда автомат находится в состоянии q_j, выходной сигнал неопределен или наоборот;
в) в том случае, когда автомат находится в состоянии q_i или q_j, выходной сигнал неопределен.
Понятия состояний с тождественными переходами и выходами дает возможность найти пару (π, ρ) разбиений алфавита состояний, эквивалентную подстановке, переводящей матрицу соединений R автомата A с $n = k \cdot l$ состояниями в квазиправильную матрицу соединений следующим образом:
1) по таблице переходов или графу автомата определяем l классов ρ разбиения по k элементов в каждом классе. Для этого в один класс ρ_i разбиения ρ включаем такие k состояний, которые имеют максимальное число тождественных переходов и выходов по всем буквам входного алфавита;
2) определяем k классов π разбиения по l элементов в каждом классе так, чтобы число тождественных переходов и выходов в каждом классе π_j- разбиения π было максимальным и выполнялось свойство $\pi_j \cap \rho_i = \{q\}$;
3) попарное пересечение классов ρ и π разбиений определяет подстановку $t \in T$, обратную подстановке, переводящей матрицу соединений R в квазиправильную матрицу соединений;
4) подстановку t^{-1} применяем к матрице R.
Если при определении классов π, ρ разбиений в каждом классе окажутся тождественными все переходы, то матрица соединений подстановкой t^{-1} будет переведена в правильную клеточную матрицу соединений.
Предложенную методику декомпозиции произвольного абстрактного автомата на два более простых автомата с минимальным числом связей между ними можно обобщить на случай декомпозиции автомата на элементарные абстрактные автоматы.
Сформулируем теперь алгоритм декомпозиции произвольного абстрактного автомата A на элементарные абстрактные автоматы с минимальным числом связей между ними. В качестве элементарных автоматов выбираем автоматы с двумя состояниями.
1°. *Подсчитываем число n состояний автомата A. Если* $n = 2 \cdot l$, *то переходим к* 2°. *Если* $n \neq 2l$, *то добавляем одно изолированное состояние и переходим к* 2°.
2°. *Записываем матрицы соединений* R_z *автономных автоматов* A_z *и разбиваем их на* 4 *клетки. Если каждая матрица* R_z *является* ПКМС, *то переходим к* 6°. *Если хотя бы одна из матриц* R_z *не является* ПКМС, *то переходим к* 3°.

3°. *Используя эвристический прием, строим пару* (π, ρ) *разбиений, определяющую прямую t и обратную* t^{-1} *подстановки множества состояний автомата А. Переходим к* 4°.
4°. *Применяем подстановку* t^{-1} *к матрице R и преобразуем ее в квазиправильную матрицу соединений. Если* $\varphi_{min} = 0$, *то переходим к* 6°. *В противном случае к* 5°.
5°. *Представляем квазиправильную матрицу соединений композицией матриц*

$$R_1 = \bigcup_{z \in Z} R_{1z} \quad и \quad R_2 = \bigcup_{z \in Z} R_{2z}.$$

Матрица соединений R_1 *является матрицей соединений элементарного абстрактного автомата* A_1. *Переходим к 7°*
6°. *Представляем полученную* ПКМС *произведением матриц* R_1 *и* R_2 *автоматов* A_1 *и* A_2. *Автомат* A_1 *представляет собой элементарный абстрактный автомат. Переходим к* 7°.
7°. *Если число состояний автомата* A_2 *равно* $l = 2$, *то переходим к* 8°. *Если* $l > 2$, *то переходим к* 1° *и проводим декомпозицию автомата* A_2 *так, как это сделано с автоматом А.*
8°. *Учитывая примененные подстановки, закодируем соответствующие переходы в элементарных абстрактных автоматах наборами полученных элементарных автоматов и запишем матрицы соединений элементарных абстрактных автоматов с учетом перекодировок. Конец работы алгоритма.*
Проиллюстрируем работу этого алгоритма на примере.

Пример 3.5. Пусть дан автомат Мили *A,* рассмотренный в предыдущем примере. Требуется провести декомпозицию автомата *A* на элементарные автоматы с минимальным числом связей между ними. Если привести матрицу соединений *R* к виду ПКМС с запретами, то она будет содержать 15 запрещенных переходов.
Используя эвристический прием, по графу автомата *A* построим пару (π, ρ) разбиений множества состояний, которая имеет вид

$$\rho = \{\rho_1, \rho_2, \rho_3\}, \quad \pi = \{\pi_1, \pi_2\},$$
$$\rho_1 = \{q_1, q_2\}, \quad \rho_2 = \{q_3, q_4\}, \quad \rho_3 = \{q_5, q_6\},$$
$$\pi_1 = \{q_2, q_4, q_5\}, \quad \pi_2 = \{q_1, q_3, q_6\}.$$

Пересекая классы π и ρ разбиений, получим подстановку

$$t = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ q_2 & q_4 & q_5 & q_1 & q_3 & q_6 \end{pmatrix}.$$

Применяя обратную подстановку

$$t^{-1} = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ q_4 & q_1 & q_5 & q_2 & q_3 & q_6 \end{pmatrix}$$

к матрице R и приводя ее к правильному виду, получим квазиправильную матрицу соединений

$$R' = \left\| \begin{array}{ccc:ccc} 0 & z_1/m_1 \vee \boxed{z_1/m_2} & 0 & 0 & \boxed{z_1/m_1} \vee \boxed{z_1/m_2} & 0 \\ \boxed{z_1/m_2} & \boxed{z_2/m_1} & 0 & z_1/m_2 & z_2/m_1 & 0 \\ 0 & z_2/m_1 & \boxed{z_1/m_2} & 0 & \boxed{z_2/m_1} & z_1/m_2 \\ \hdashline 0 & z_1/m_2 \vee \boxed{z_1/m_1} & 0 & 0 & 0 & 0 \\ z_1/m_2 & z_2/m_1 & 0 & 0 & 0 & 0 \\ 0 & z_2/m_1 & z_1/m_2 & 0 & 0 & 0 \end{array} \right\|,$$

которая содержит 8 запретов. Представим автономные матрицы R'_{z_1} и R'_{z_2} произведением двух матриц с отмеченными переходами

$$R'_{z_1} = R'_{1z_1} \times R'_{2z_1} = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \left\| \begin{matrix} \underline{z_1} & \underset{\circ}{z_1} \\ z_1 & 0 \end{matrix} \right\| \times \begin{matrix} \\ w_1 \\ w_2 \\ w_3 \end{matrix} \left\| \begin{matrix} 0 & z_1/m_1 \vee z_1/m_2 & 0 \\ z_1/m_2 & 0 & 0 \\ 0 & 0 & z_1/m_2 \end{matrix} \right\|,$$

$$R'_{z_2} = R'_{1z_2} \times R'_{2z_2} = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \left\| \begin{matrix} \underline{z_2} & \underset{\sim}{z_2} \\ z_2 & 0 \end{matrix} \right\| \times \begin{matrix} \\ w_1 \\ w_2 \\ w_3 \end{matrix} \left\| \begin{matrix} 0 & 0 & 0 \\ 0 & z_2/m_1 & 0 \\ 0 & z_2/m_1 & 0 \end{matrix} \right\|.$$

Вводя связи в матрицы сомножители и объединяя их по буквам входного алфавита, получим матрицы соединений R_1 и R_2, которые имеют вид

$$R_1 = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \left\| \begin{matrix} w_1 z_1 \vee w_3 z_2 & (w_2 \vee w_3)\, z_1 \vee w_2 z_2 \\ z_1 \vee z_2 & 0 \end{matrix} \right\|,$$

$$R_2 = \begin{matrix} \\ w_1 \\ w_2 \\ w_3 \end{matrix} \left\| \begin{matrix} 0 & v_1 z_1/m_1 \vee v_2 z_1/m_2 & 0 \\ z_1/m_2 & z_2/m_1 & 0 \\ 0 & z_2/m_1 & z_1/m_2 \end{matrix} \right\|.$$

Проведем теперь декомпозицию автомата A_2, заданного матрицей соединений R_2. Для этого дополним матрицу соединений R_2 изолированным состоянием w_4. Используя эвристический прием, по графу автомата A_2 находим пару (π', ρ') разбиений множества состояний

$$\pi' = \{\pi'_1,\ \pi'_2\}, \qquad \rho' = \{\rho'_1,\ \rho'_2\},$$
$$\pi'_1 = \{w_1,\ w_4\}, \qquad \pi'_2 = \{w_2,\ w_3\},$$
$$\rho'_1 = \{w_1,\ w_2\}, \qquad \rho'_2 = \{w_3,\ w_4\},$$

которая эквивалентна подстановке

$$t_1 = \begin{pmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_4 & w_2 & w_3 \end{pmatrix}.$$

Применяя подстановку

$$t_1^{-1} = \begin{pmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_3 & w_4 & w_2 \end{pmatrix}$$

к дополненной матрице соединений R_2 и вводя запрещенные переходы, получим квазиправильную матрицу соединений

$$R'_2 = \begin{Vmatrix} 0 & 0 & v_1z_1/m_1 \vee v_2z_1/m_2 & 0 \\ 0 & 0 & 0 & 0 \\ z_1/m_2 & 0 & z_2/m_1 \vee \boxed{z_1/m_2} & 0 \\ 0 & \boxed{z_1/m_2} & z_2/m_1 & z_1/m_2 \end{Vmatrix}.$$

Проводя разложение матрицы R'_2, приходим к матрицам соединений R_{21} и R_{22} элементарных абстрактных автоматов A_{21} и A_{22}, которые имеют вид

$$R_{21} = \begin{matrix} s_1 \\ s_2 \end{matrix} \begin{Vmatrix} 0 & v_1z_1 \vee v_2z_1 \\ h_1z_1 & z_2 \vee h_2z_1 \end{Vmatrix},$$

$$R_{22} = \begin{matrix} h_1 \\ h_2 \end{matrix} \begin{Vmatrix} v_1z_1/m_1 \vee v_2z_1/m_2 \vee s_2z_1/m_2 & 0 \\ 0 & s_2z_1/m_2 \vee z_2/m_1 \end{Vmatrix}.$$

Учитывая подстановку t_1, перекодируем состояния автомата A_2 следующим образом:

$$w_1 = (s_1,\ h_1), \qquad w_2 = (s_2,\ h_1), \qquad w_3 = (s_2,\ h_2), \qquad w_4 = (s_1,\ h_2).$$

Тогда матрицу соединений R_1 элементарного абстрактного автомата A_1 можно переписать в форме

$$R_1 = \begin{matrix} v_1 \\ v_2 \end{matrix} \begin{Vmatrix} s_1h_1z_1 \vee s_2h_2z_2 & s_2h_1z_1 \vee s_2h_2z_1 \vee s_2h_1z_2 \\ z_1 \vee z_2 & 0 \end{Vmatrix}.$$

В итоге построена декомпозиция абстрактного автомата Мили A на три элементарных абстрактных автомата A_1, A_{21}, A_{22}, показанных соответственно на рис. 3.10, *а, б, в,* каждый из которых содержит два состояния.

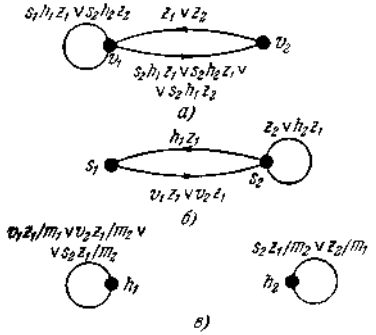


Рис. 3.10.

3.6. Декомпозиция автоматов с выделением заданных стандартных автоматов

При синтезе автоматов в однородных вычислительных средах, построенных на основе больших интегральных схем, возникает следующая задача. Пусть имеется сложный абстрактный автомат, который необходимо реализовать в вычислительной среде, и задан один или несколько стандартных автоматов (например, большая интегральная схема), из которых строится вычислительная среда. Требуется определить возможность реализации сложного автомата в вычислительной среде с помощью последовательного, параллельного или смешанного соединения стандартных автоматов. Если это невозможно, то необходимо выделить из исходного сложного автомата наибольшее число стандартных автоматов так, чтобы оставшийся нестандартный автомат совместно с выделенными автоматами функционировал бы как исходный автомат. Эту задачу будем называть *декомпозицией автоматов с выделением заданных стандартных автоматов.*

Если рассматривать полугруппы автоматов по различным операциям, которые содержательно соответствуют различным видам соединений автоматов между собой, то задачу декомпозиции автоматов с выделением стандартных автоматов формально можно трактовать как задачу делимости в полугруппах автоматов и сводить, таким образом, к решению уравнений в полугруппах абстрактных автоматов. Сравнительно простое решение задачи делимости найдено в полугруппе $\mathfrak{B}_*$ автоматов по операции суперпозиции. Решая суперпозиционные уравнения в этой полугруппе, мы тем самым решаем задачу последовательной декомпозиции с выделением

заданных автоматов Использование теорем 3.11—3.14 позволяет сводить задачи параллельной декомпозиции с выделением стандартных автоматов к решению суперпозиционных уравнений.
Рассмотрим вначале задачу *последовательной декомпозиции с выделением стандартных автоматов.* Решение этой задачи сводится к решению *суперпозиционных уравнений* вида

$$A * G = B, \quad (3.5)$$

$$H * C = B, \quad \textbf{(3.6)}$$

где *B*— исходный автомат, *A* и *C* — заданные стандартные автоматы, а *G* и *H* — неизвестные автоматы.
Пусть $\mathfrak{A}_*$ — бесконечная полугруппа абстрактных автоматов Мили.
Для произвольного автомата $A \in \mathfrak{A}_*$, индуцирующего частичное отображение множества слов $\mathfrak{S}(X)$ алфавита *X* на множество слов $\mathfrak{S}(Y)$ алфавита Y, можно определить левый E_X и правый E_Y единичные автоматы такие, что

$$\left.\begin{aligned} E_X * A &= A, \\ A * E_Y &= A. \end{aligned}\right\} \quad \textbf{(3.7)}$$

Для автомата $A \in \mathfrak{A}_*$, индуцирующего биективное отображение $\mathfrak{S}(X)$ на $\mathfrak{S}(Y)$, справедливо

$$\left.\begin{aligned} A * A^{-1} &= E_X, \\ A^{-1} * A &= E_Y. \end{aligned}\right\} \quad \textbf{(3.8)}$$

Если автомат *A* индуцирует биективное отображение множества $\mathfrak{S}(X)$ на себя, то

$$A * A^{-1} = A^{-1} * A = E. \quad (3.9)$$

Сформулируем теперь две теоремы, из которых следует метод решения суперпозиционных уравнений.
Теорема 3.16. *Пусть автомат A индуцирует биективное отображение* φ *множества* $\mathfrak{S}(X_1)$ *на* $\mathfrak{S}(Y_1)$, *а автомат B индуцирует произвольное отображение* ψ *множества* $\mathfrak{S}(X_1)$ *на* $\mathfrak{S}(Y_2)$. *В этом случае автомат G, индуцирующий отображение* ψ *множества* $\mathfrak{S}(Y_1)$ *на* $\mathfrak{S}(Y_2)$, *такой, что* $A * G = B$, *существует и однозначно определяется по формуле*

$$G = A^{-1} * B. \quad \textbf{(3.10)}$$

Доказательство. Пусть $B = A * G$. Для любого *G* из (3.7) и (3.8) справедливо $G = A^{-1} * A * G$.
Следовательно, учитывая ассоциативность суперпозиции, $G = A^{-1} * B$. Покажем, что найденное *G* есть решение уравнения

(3.5). Пусть $G = A^{-1} * B$. Имеем $A * A^{-1} * B = B$. Отсюда $A * G = B$. Докажем единственность. Пусть уравнение (3.5) имеет два разных решения *G* и G_1, т. е. $A * G = B$ и $A * G_1 = B$. Отсюда $A^{-1} * A * G = A^{-1} * B$ и $A^{-1} * A * G_1 = A^{-1} * B$. Теперь из ассоциативности суперпозиции, соотношений (3.7) и (3.8) получаем $E * G = A^{-1} * B$ и $E * G_1 = A^{-1} * B$. Поэтому $G = G_1$. Этим теорема 3.16 доказана.

Теорема 3.17. *Пусть автомат C индуцирует биективное отображение* ψ *множества* $\mathfrak{S}(X_2)$ *на* $\mathfrak{S}(Y_2)$, *а автомат B индуцирует произвольное отображение* η *множества* $\mathfrak{S}(X_1)$ *на* $\mathfrak{S}(Y_2)$. *В этом случае автомат H, индуцирующий отображение* φ *множества* $\mathfrak{S}(X_1)$ *на* $\mathfrak{S}(X_2)$, *такой, что* $H * C = B$, *существует и однозначно определяется по формуле*

$$H = B * C^{-1}. \quad (3.11)$$

Доказательство аналогично доказательству теоремы 3.16.

Из (3.9) и теорем 3.16 и 3.17 вытекает следствие.

Следствие 3.2. *Множество абстрактных автоматов* $\mathfrak{D}$, *индуцирующих биективное отображение множества* $\mathfrak{S}(X)$ *алфавита X на себя, по операции суперпозиции образует группу* $\mathfrak{D}$.

Проиллюстрируем решение уравнений типа (3.5) на примере.

Пример 3.6. Даны автоматы *A* и *B*, показанные на рис. 3.11 и 3.12.

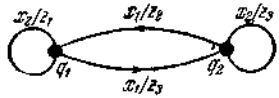


Рис. 3 11.

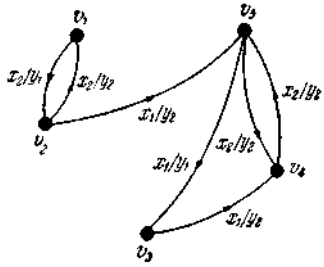


Рис. 3.12.

Найти автомат *G*, являющийся решением уравнения $A * G = B$.

Запишем матрицы соединений R_A и R_B автоматов A и B:

$$R_A = \begin{Vmatrix} x_2/z_1 & x_1/z_3 \\ x_1/z_2 & x_2/z_3 \end{Vmatrix},$$

$$R_B = \begin{Vmatrix} 0 & x_2/y_1 & 0 & 0 & 0 \\ x_2/y_2 & 0 & 0 & 0 & x_1/y_2 \\ 0 & 0 & 0 & x_1/y_2 & 0 \\ 0 & 0 & 0 & 0 & x_2/y_2 \\ 0 & 0 & x_1/y_1 & x_2/y_2 & 0 \end{Vmatrix}.$$

Определим матрицу соединений автомата A^{-1}. Получим

$$R_{A^{-1}} = \begin{Vmatrix} z_1/x_2 & z_3/x_1 \\ z_2/x_1 & z_3/x_2 \end{Vmatrix}.$$

По формуле (3.10) находим автомат G, матрица соединений которого имеет вид

$$R_G = \begin{Vmatrix} 0 & z_1/y_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ z_1/y_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & z_3/y_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & z_3/y_2 & 0 \\ 0 & 0 & 0 & 0 & z_1/y_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & z_1/y_2 & 0 & 0 & 0 & z_3/y_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & z_3/y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & z_2/y_2 & z_3/y_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & z_2/y_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & z_3/y_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & z_3/y_2 & 0 \end{Vmatrix}.$$

После минимизации получим

$$R_G = \begin{Vmatrix} 0 & z_1/y_1 & 0 \\ z_1/y_2 & 0 & z_3/y_2 \\ 0 & z_3/y_2 & z_2/y_1 \end{Vmatrix}.$$

Графоид автомата G показан на рис. 3.13.

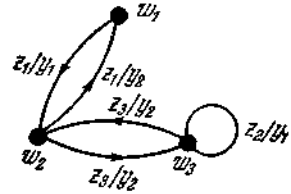


Рис. 3 13.

Легко проверить, что $A * G = B$.

При решении уравнений (3.5) и (3.6) мы полагали, что заданные стандартные автоматы реализуют биективное отображение. Поскольку они могут индуцировать произвольное отображение, то необходимо рассматривать случай сюръективного отображения.

Пусть автомат A индуцирует сюръективное отображение множества слов $\mathfrak{S}(X)$ на множество слов $\mathfrak{S}(Y)$. Матрица соединений R_A такого автомата содержитв какой-либо строке такие пары вход — выход, которые имеют одинаковые выходные сигналы. Поэтому матрица соединений $R_{A^{-1}}$ не будет автоматной матрицей.

Для автомата A введем понятие *иссечения.* Иссечением автомата A называется такой автомат A^0, для которого выполняется соотношение

$$A^0 * A = E_Y. \quad (3.12)$$

Чтобы найти матрицу соединений R_{A0} автомата A^0, необходимо в матрице $R_{A^{-1}}$ произвольным образом исключить неоднозначные переходы. Очевидно, что получение иссечения неоднозначно.

Сформулируем теоремы для случая, когда заданный стандартный автомат реализует сюръективное отображение.

Теорема 3.18. *Пусть автомат A индуцирует сюръективное отображение* φ *множества* $\mathfrak{S}(X_1)$ *на* $\mathfrak{S}(Y_1)$, *а автомат B индуцирует произвольное отображение* η *множества* $\mathfrak{S}(X_1)$ *на* $\mathfrak{S}(Y_2)$. *Если существует автомат* A^0 *такой, что* $A * A^0 = E_X$, *то автомат G, индуцирующий отображение* ψ *множества* $\mathfrak{S}(Y_1)$ *на* $\mathfrak{S}(Y_2)$, *такой, что* $A * G = B$, *существует и определяется выражением*

$$G = A^0 * B. \quad (3.13)$$

Теорема 3.19. *Пусть автомат C индуцирует сюръективное отображение* ψ *множества* $\mathfrak{S}(X_2)$ *на* $\mathfrak{S}(Y_2)$, *а автомат B индуцирует произвольное отображение* η *множества* $\mathfrak{S}(X_1)$ *на* $\mathfrak{S}(Y_2)$. *Если существует автомат* C^0 *такой, что* $C * C^0 = E_{X_2}$, *то автомат H, индуцирующий отображение* φ *множества* $\mathfrak{S}(X_1)$ *на* $\mathfrak{S}(X_2)$, *такой, что* $H * C = B$, *существует и определяется выражением*

$$H = B * C^0. \quad (3.14)$$

Проведем доказательство теоремы 3 19. Пусть $H * C = B$. Если $C * C^0 = E_{X_2}$, то $H * C * C^0 = B * C^0$. Тогда, учитывая (3.7), получим $H = B * C^0$. Покажем, что найденное H является решением

уравнения (3.6). Если $H = B * C^0$, то $H * C = B * C^0 * C$. Используя ассоциативность суперпозиции и (3.12), получим $B = H * C$. Этим теорема доказана.

Заметим, что найденные в соответствии с теоремами 3.18 и 3.19 решения не являются единственными, так как не единственны иссечения автоматов А и С.

Покажем на примере решение уравнения вида (3.6), полагая, что исходный стандартный автомат индуцирует сюръективное отображение.

Пример 3.7. Даны автоматы В и С, показанные соответственно на рис. 3.14 и 3.15.

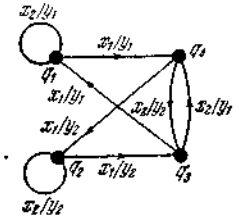


Рис. 3 14.

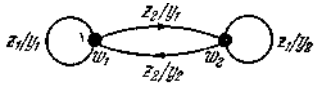


Рис. 3. 15

Найти автомат Н, который будучи последовательно соединен с автоматом С, совместно с ним функционировал бы как автомат *B*.

Запишем матрицы соединений R_B и R_C автоматов *B* и *C:*

$$R_B = \left\| \begin{array}{cccc} x_2/y_1 & 0 & 0 & x_1/y_1 \\ 0 & x_2/y_2 & x_1/y_2 & 0 \\ x_1/y_1 & 0 & 0 & x_2/y_1 \\ 0 & x_1/y_2 & x_2/y_2 & 0 \end{array} \right\|,$$

$$R_C = \left\| \begin{array}{cc} z_1/y_1 & z_2/y_1 \\ z_2/y_2 & z_1/y_2 \end{array} \right\|.$$

Найдем иссечение C^0 автомата С, матрица соединений которого имеет вид

$$R_{C^0} = \left\| \begin{array}{cc} 0 & y_1/z_2 \\ 0 & y_2/z_1 \end{array} \right\|.$$

Проверяем выполнение условия $C * C^0 = E_Z$. Беря суперпозицию R_C и R_{C^0}, после минимизации получим

$$R_{EZ} = \| z_1/z_1 \vee z_2/z_2 \|.$$

По формуле (3 14) находим автомат *H,* матрица соединений которого после минимизации принимает вид

$$R_H = \begin{Vmatrix} x_2/z_1 & x_1/z_2 \\ x_1/z_1 & x_2/z_2 \end{Vmatrix}.$$

Легко убедиться, что $H * C = B.$ Автомат *H* показан на рис. 3.16

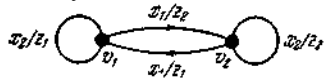


Рис. 3. 16

Рассмотрим теперь *декомпозицию с выделением стандартных автоматов,* когда выделяемый стандартный автомат и оставшаяся нестандартная часть автомата работают *параллельно* друг с другом. Эту задачу можно трактовать как задачу *решения уравнений* вида

$$G \times A \sim A \times G \sim B, \quad (3.15)$$

$$H \otimes A \sim A \otimes H \sim B, \quad (3.16)$$

$$K + A \sim A + K \sim B, \quad (3.17)$$

$$N \circ A \sim A \circ N \sim B, \quad (3.18)$$

где *B* — исходный автомат, *A* — заданный стандартный автомат, а *G, H, K, N* — неизвестные автоматы

Вообще говоря, используя только операции, записанные в уравнениях (3.15) — (3.18), нельзя предложить метод решения этих уравнений, за исключением того случая, когда заведомо изве стно, что исходный автомат принадлежит подполугруппе разложимых автоматов по данной операции, да и то этот метод сводится к полному перебору в классе изоморфизма автоматов

Доказанные в предыдущем параграфе теоремы 3.11— 3. 14 позволяют сводить уравнения (3. 15) — (3. 18) к *суперпозиционным уравнениям* и определять случаи, когда в результате решения полученных суперпозиционных уравнений находим *решения уравнений по другим операциям.*

Действительно, если имеется исходный автомат $B = (Z, Q, M, q_1 \in Q, U(z \in Z/m \in M))$ и задан стандартный автомат $A = (Y, W, T, w_1 \in W, P(y \in Y/t \in T))$, то для определения автомата $G = (X, V, L, v_1 \in V, F(x \in X/l \in L))$, который является решением уравнения (3. 15), на основании теоремы 3. 11 можно перейти к двум суперпозиционным уравнениям вида

$$(G \times E_Y) * (E_L \times A) = B, \quad (3.19)$$

$$(A \times E_X) * (E_T \times G) = B, \quad (3.20)$$

т. е. ,

$$G' * A' = B, \quad (3.19')$$

$$A'' * G'' = B. \quad (3.20')$$

Решая уравнение (3. 19'), находим автомат G' Если автомат G' может быть представлен в виде $G' = G \times E_Y$, то автомат G — решение уравнения (3. 15)

Если автомат G' не представим в виде $G' = G \times E_Y$, то в результате решения уравнения (3. 19') из автомата B выделен стандартный автомат A, который совместно с автоматом G' работает так, как это показано на рис. 3. 17

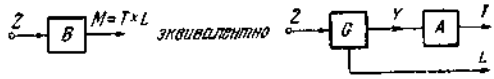


Рис. 3. 17

Если решается уравнение (3. 20') и автомат G'' можно представить в виде $G'' = E_T \times G$, то автомат G является решением уравнения (3. 15)

В том случае, когда автомат G'' нельзя представить в виде $G'' = E_T \times G$, то

в результате решения уравнения (3.20') из автомата B выделяется стандартный автомат A, который совместно с оставшейся частью G'' работает так, как это показано на рис. 3. 18.

Рис. 3. 18

Пример 3.8. Пусть имеется автомат B, матрица соединений которого имеет вид

$$R_B = \begin{Vmatrix} z_1/m_1 & z_2/m_2 & 0 & 0 & z_3/m_3 & z_4/m_4 \\ z_2/m_1 & z_1/m_2 & 0 & 0 & z_4/m_3 & z_3/m_4 \\ z_3/m_1 & z_4/m_2 & 0 & 0 & z_1/m_3 & z_2/m_4 \\ z_4/m_1 & z_3/m_2 & 0 & 0 & z_2/m_3 & z_1/m_4 \\ 0 & 0 & z_1/m_1 & z_2/m_2 & z_3/m_3 & z_4/m_4 \\ 0 & 0 & z_2/m_1 & z_1/m_2 & z_4/m_3 & z_3/m_4 \end{Vmatrix},$$

и задан стандартный автомат A с матрицей соединений

$$R_A = \left\| \begin{matrix} y_1/t_1 & y_2/t_2 \\ y_2/t_1 & y_1/t_2 \end{matrix} \right\|.$$

Необходимо найти автомат $G = (X, V, L, v_1 \in V, F(x \in X/l \in L))$, параллельная одновременная работа которого с автоматом *A* эквивалентна функционированию автомата *B*. Другими словами, необходимо найти автомат, который является решением уравнения (3.15).

Так как $Z = \{z_1, z_2, z_3, z_4\}$, $M = \{m_1, m_2, m_3, m_4\}$ и $Y = \{y_1, y_2\}$, $T = \{t_1, t_2\}$, а для того, чтобы автомат *G* был решением уравнения (3.15), необходимо выполнение условий $Z \subseteq X \times Y$ и $M \subseteq L \times T$, предположим, что входной и выходной алфавиты автомата *G* соответственно равны $X = \{x_1, x_2\}$, $L = \{l_1, l_2\}$. Определим $R_{A'}$. Обозначим $S = L \times Y = \{s_1, s_2, s_3, s_4\}$, $M = L \times T = \{m_1, m_2, m_3, m_4\}$ в естественной упорядоченности пар. Тогда

$$R_{A'} = R_{E_L} \times R_A = \left\| \begin{matrix} s_1/m_1 \vee s_3/m_3 & s_2/m_2 \vee s_4/m_4 \\ s_2/m_1 \vee s_4/m_3 & s_1/m_2 \vee s_3/m_4 \end{matrix} \right\|.$$

Матрица соединений $R_{A'}^{-1}$ автомата, обратного автомату *A'*, имеет вид

$$R_{A'}^{-1} = \left\| \begin{matrix} m_1/s_1 \vee m_3/s_3 & m_2/s_2 \vee m_4/s_4 \\ m_1/s_2 \vee m_3/s_4 & m_2/s_1 \vee m_4/s_3 \end{matrix} \right\|.$$

В результате решения суперпозиционного уравнения (3.19') по формуле (3.11) получим матрицу соединений

$$R_{G'} = \left\| \begin{matrix} z_1/s_1 \vee z_2/s_2 & 0 & z_3/s_3 \vee z_4/s_4 \\ z_3/s_1 \vee z_1/s_2 & 0 & z_1/s_3 \vee z_2/s_4 \\ 0 & z_1/s_1 \vee z_2/s_2 & z_3/s_3 \vee z_4/s_4 \end{matrix} \right\|,$$

которую, как нетрудно видеть, можно представить произведением матриц

$$R_G = \left\| \begin{matrix} x_1/l_1 & 0 & x_2/l_2 \\ x_2/l_1 & 0 & x_1/l_2 \\ 0 & x_1/l_1 & x_2/l_2 \end{matrix} \right\|$$

и

$$R_{E_Y} = \| y_1/y_1 \vee y_2/y_2 \|.$$

Таким образом, найдено решение уравнения (3.15), которое определяет автомат G. Легко проверить, что $G \times A \sim B$.

Автоматы *B, A* и *G* показаны соответственно на рис. 3.19, 3.20 и 3.21.

1

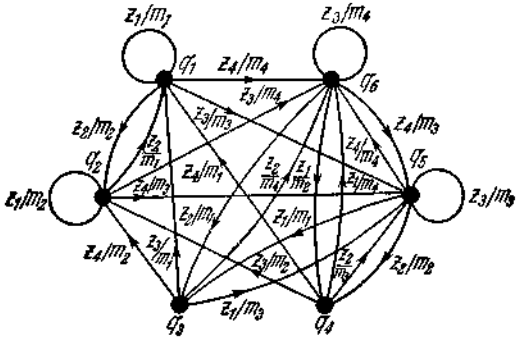


Рис. 3. 19.

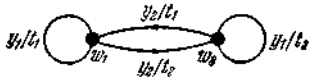


Рис. 3 20.

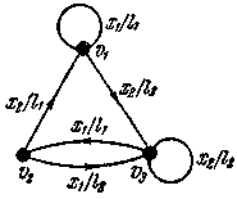


Рис. 3 21.

Пусть необходимо найти решение уравнения (3.16), в котором $B = (X, Q, M, q_1 \in Q, U(x \in X/m \in M))$ — исходный автомат, $A = (X, W, T, w_1 \in W, P(x \in X/t \in T))$, — заданный стандартный автомат, а $H = (X, V, L, v_1 \in V, F(x \in X/l \in L))$ — неизвестный автомат.

На основании теоремы 3. 12 от уравнения (3.16) можно перейти к двум уравнениям

$$(H \otimes E_X) * (E_L \times A) = B, \quad (3.21)$$

$$(A \otimes E_X) * (E_T \times H) = B, \quad (3.22)$$

т. е.

$$H' * A' = B, \quad (3.21')$$

$$A'' * H'' = B. \quad (3.22)$$

Решая суперпозиционное уравнение (3.21'), находим автомат H'. Если автомат H' можно представить в виде $H' = H \otimes E_X$, то найденный таким образом автомат H является решением уравнения (3.16). Если из автомата H' нельзя извлечь E_X, то в результате решения (3.21') из автомата B выделен стандартный автомат A, который совместно с автоматом H' работает так, как это показано на рис. 3.22.

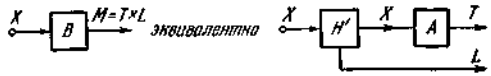


Рис. 3.22.

Аналогично рассуждая, если при решении уравнения (3.22′) из автомата H'' можно выделить E_T по формуле $H'' = E_T \times H$, то автомат H — решение уравнения (3.16). В противном случае при решении уравнения (3.22′) получаем декомпозицию автомата B на заданный автомат A и автомат H'', совместная работа которых, показанная на рис. 3 23, эквивалентна функционированию исходного автомата B.

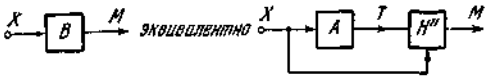


Рис. 3. 23

Далее на основании теоремы 3. 13 от уравнения (3.17) можно перейти к уравнениям

$$(K + E_Y) * (E_L + A) = B, \quad (8.23)$$

$$(A + E_X) * (E_T + K) = B, \quad (3.24)$$

т. е.

$$K' * A' = B, \quad (3.23')$$

$$A'' * K'' = B. \quad (3.24')$$

Решением уравнения (8.23') является автомат K'. Если K' можно представить в виде $K' = K + E_{Y,\_}$, то автомат K — решение уравнения (3. 17). Если из K' нельзя извлечь E_Y, то в результате решения уравнения (3. 23') из автомата B выделяем стандартный автомат A, который совместно с автоматом K' работает так, как показано на рис. 3. 24.

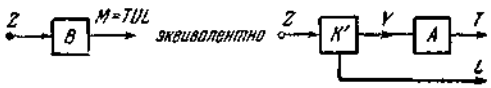


Рис. 3.24.

Если в результате решения уравнения (3.24') автомат K'' нельзя представить в виде $K'' = E_T + K$, то получаем декомпозицию автомата B на стандартный автомат A и автомат K'', совместная работа которых, показанная на рис. 3.25, эквивалентна работе автомата B.

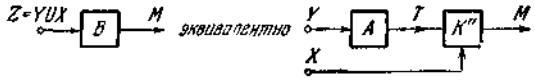


Рис. 3.25.

Пример 3.9. Пусть имеется автомат *B,* матрица соединений которого имеет вид

$$R_B = \left\| \begin{array}{cccccc} y_1/t_1 & y_2/t_2 & x_1/l_2 & 0 & x_2/l_2 & 0 \\ y_2/t_2 & y_1/t_1 & 0 & x_1/l_2 & 0 & x_2/l_2 \\ x_2/l_1 & 0 & y_1/t_1 & y_2/t_2 & x_1/l_2 & 0 \\ 0 & x_2/l_1 & y_2/t_2 & y_1/t_1 & 0 & x_1/l_2 \\ 0 & 0 & x_1/l_1 & 0 & x_2/l_2 \vee y_1/t_1 & y_2/t_2 \\ 0 & 0 & 0 & x_1/l_1 & y_2/t_2 & x_2/l_2 \vee y_1/t_1 \end{array} \right\|,$$

и задан стандартный автомат A с матрицей соединений

$$R_A = \left\| \begin{array}{cc} y_1/t_1 & y_2/t_2 \\ y_2/t_2 & y_1/t_1 \end{array} \right\|.$$

Необходимо найти автомат $K = (X, V, L, v_1 \in V, F(x \in X | l \in L))$, который является решением уравнения (3.17). Так как $Z = \{x_1, x_2, y_1, y_2\}$, $M = \{l_1, l_2, t_1, t_2\}$, а $Y = \{y_1, y_2\}$, $T = \{t_1, t_2\}$, то для того, чтобы автомат K был решением уравнения (3.17), необходимо $Z = X \cup Y$, $M = T \cup L$. Поэтому положим, что $X = \{x_1, x_2\}$, $L = \{l_1, l_2\}$.

На основании теоремы 3.13 находим матрицу соединений $R_{A'}$ автомата A'. Получим

$$R_{A'} = R_{E_L} \div R_A = \left\| \begin{array}{cc} l_1/l_1 \vee l_2/l_2 \vee y_1/t_1 & y_2/t_2 \\ y_2/t_2 & l_1/l_1 \vee l_2/l_2 \vee y_1/t_1 \end{array} \right\|.$$

Решая суперпозиционное уравнение (3.23') по формуле (3.11), находим матрицу соединений $R_{K'}$ автомата K', которая имеет вид

$$R_{K'} = \left\| \begin{array}{ccc} y_1/y_1 \vee y_2/y_2 & x_1/l_2 & x_2/l_2 \\ x_2/l_1 & y_1/y_1 \vee y_2/y_2 & x_1/l_2 \\ 0 & x_1/l_1 & y_1/y_1 \vee y_2/y_2 \vee x_2/l_2 \end{array} \right\|.$$

Матрицу $R_{K'}$ представляем суммой матриц R_K и R_{E_Y}. Получаем матрицы

$$R_K = \left\| \begin{array}{ccc} 0 & x_1/l_2 & x_2/l_2 \\ x_2/l_1 & 0 & x_1/l_2 \\ 0 & x_1/l_1 & x_2/l_2 \end{array} \right\|$$

и

$$R_{E_Y} = \| y_1/y_1 \vee y_2/y_2 \|,$$

которые определяют автоматы K и E_Y.

Таким образом, результатом решения уравнения (3.17) является автомат *K*. Легко проверить, что $K+A \sim B$. Автоматы *B, A* и *K* показаны соответственно на рис. 3.26, 3.27 и 3.28.

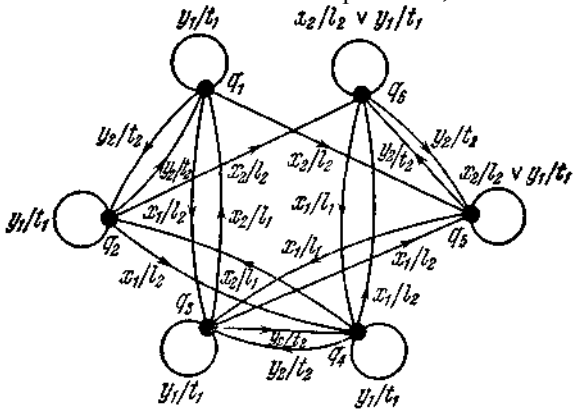


Рис. 3. 26

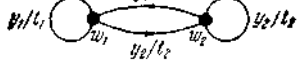


Рис. 3. 27.

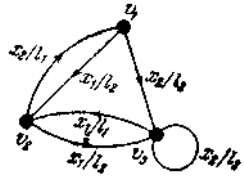


Рис. 3. 28.

Наконец, пусть требуется решить композиционное уравнение вида (3.18), в котором $B = (Z, Q, M, q_1 \in Q,$

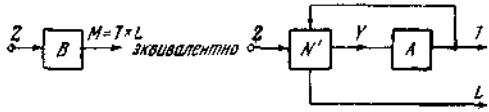


Рис. 8.29.

$U(z \in Z/m \in M))$ — исходный автомат, $A = (Y, W, T,$ $w_1 \in W, P(y \in Y, l \in L/t \in T))$ — заданный стандартный

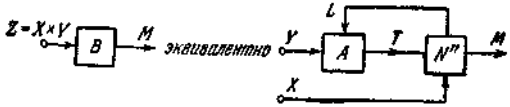


Рис. 3 30.

автомат, а $N = (X, V, L, v_1 \in V, F(x \in X, t \in T/t \in L))$ — неизвестный автомат.

На основании теоремы 3.14 от уравнения (3.18) можно перейти к уравнениям

$$(N \circ E_Y \circ E_T \circ E_L) * (A \circ E_L \circ E_T) = B, \quad (3.25)$$

$$(A \circ E_X \circ E_T \circ E_L) * (N \circ E_L \circ E_T) = B, \quad (3.26)$$

т. е.

$N' * A' = B$, (3.25')

$A'' * N'' = B$. **(3.26')**

Автомат N' является решением уравнения (3.25'). Если N' можно представить композицией автоматов $N' = N \circ E_Y \circ E_T \circ E_L$, то автомат N — решение уравнения (3.18). Если из N' нельзя извлечь E_Y, E_T и E_L, то в результате решения уравнения (3.25') из автомата B выделяем стандартный автомат A, который совместо с автоматом N' работает так, как показано на рис. 3.29.

Если в результате решения уравнения (3.26') автомат N'' нельзя представить в виде $N'' = N \circ E_L \circ E_T$, то получаем декомпозицию автомата B на стандартный автомат A и автомат N'', совместная работа которых эквивалентна работе автомата B и показана на рис. 3.30.

4. Структурный синтез автоматов

4.1. Этапы синтеза автоматов

В теории автоматов процесс синтеза автомата принято подразделять на отдельные этапы.

На первом, или предварительном, этапе синтеза формулируются (часто словесно) условия работы автомата, т. е. определяются условия его взаимодействия с другими устройствами или какими-либо объектами; выявляются необходимые входы и выходы автомата и намечается общий закон появления выходных сигналов в зависимости от воздействия на входы автомата. При синтезе достаточно сложного автомата часто его разбивают на отдельные блоки. В этом случае первый этап синтеза иногда называют *этапом блочного синтеза автомата.*

В Институте кибернетики АН УССР под руководством акад. В. М. Глушкова разработана формализованная методика блочного синтеза ЭВМ, позволяющая получать более экономичные решения, чем при интуитивном методе.

М. А. Гаврилов разработал блочный метод синтеза, позволяющий синтезировать один сложный автомат из простых блоков, выполняющих отдельные его функции. Такой метод дает возможность упростить формулировку условий работы (на языке таблиц переходов) и процесс построения автомата.
На этом этапе условия работы УА принято задавать в виде процессов управления на естественном или одном из формализованных языков, в частности на широко распространенном языке сетей Петри.
В работе приведены краткие сведения о блочном синтезе УА, в том числе и при программно-аппаратурной их реализации. Методы блочного синтеза УА при их реализации в базисе микропроцессоров изложены в ряде работ.
На втором этапе синтеза выявляются закон функционирования автомата (его функции переходов и функции выходов) и формальное описание автомата одним из принятых для этого способов. Этот этап синтеза принято называть *этапом абстрактного синтеза или синтезом абстрактного автомата.*
На этапе абстрактного синтеза автомата не интересуются теми физическими элементами, из которых он должен состоять. Нет нужды рассматривать, какие значения могут принимать входы, выходы и элементы памяти автомата. Важно знать лишь число возможных различных состояний входов и выходов автомата, число различных его внутренних состояний, а также законы изменения внутренних состояний автомата и выработки им тех или иных состояний выходов при поступлении на его входы той или иной последовательности состояний входа.
Исследование абстрактного синтеза автоматов было начато С. К-Клини, которым был предложен так называемый *язык регулярных событий.* В дальнейшем абстрактный синтез автомата, основанный на использовании этого языка, был усовершенствован В. М. Глушковым , который разработал алгоритм абстрактного синтеза, годный для программирования на ЭВМ.
Другим языком, на котором базируется абстрактный синтез автомата, является язык исчисления предикатов, исследованный Б. А. Трахтенбротом.
Результатом второго этапа синтеза является задание автомата одним из стандартных способов, к которым относятся таблицы и матрицы переходов, а также таблицы включений и диаграммы переходов
В некоторых случаях удобно задавать автомат временными функциями, а МА — логическими схемами алгоритмов.
На втором этапе синтеза, очевидно, выявляется и емкость памяти автомата. При этом, несмотря на стремление получить уже при

абстрактном синтезе автомат с минимальным числом внутренних состояний, в ряде случаев получаем автоматы, у которых число внутренних состояний заметно отличается от минимального. В связи с этим на следующем, третьем этапе производится *минимизация числа внутренних состояний автомата.*

На четвертом этапе синтеза производится *кодирование* или, как говорят, *размещение внутренних состояний.* Этот этап находится на грани абстрактного и структурного рассмотрения автомата. После кодирования внутренних состояний автомата, состояний входа и выхода выписываются канонические уравнения.

На пятом этапе синтеза завершается выбор структуры автомата построением так называемой *функциональной схемы* комбинационной части автомата (однотактной схемы) или, как говорят, логического преобразователя. При этом синтез автомата с памятью (многотактпой схемы) сводится к синтезу автомата без памяти (однотактной схемы) с помощью понятия *одно-тактного эквивалента*, когда s обратных связей *n, k*-полюсника обрывается и производится синтез преобразователя с $n+s$ входами и $k+s$ выходами.

Этот этап синтеза в теории автоматов наиболее развит.

Шестой этап синтеза включает электрический или другой расчет элементов схемы и моделирование автомата с целью проверки его работоспособности.

На седьмом этапе осуществляется размещение деталей на платах, составление монтажных схем и технической документации. При использовании интегральных схем и особенно схем с большой и сверхбольшой степенью интеграции (БИС и СБИС) на этом этапе осуществляется размещение элементов функциональной схемы по корпусам, а затем размещение корпуса на стандартных платах, часто называемых *типовыми элементами замены* (ТЭЗ) —типовые элементы замены по блокам и блоки по шкафам.

Разделение процесса синтеза на этапы, с одной стороны, облегчает процесс построения автомата, а с другой стороны, может привести к усложнению его структуры. Так, рассматривая на отдельных этапах процессы минимизации числа внутренних состояний, кодирование внутренних состояний и синтез структуры логического преобразователя, можно прийти к менее оптимальному решению по сравнению с тем случаем, когда эти три этапа были бы совмещены. Этот недостаток пытаются уменьшить тем, что на каждом этапе синтеза стараются учесть его влияние на следующий этап.

Разделение процесса синтеза автомата на такие семь этапов является достаточно условным, хотя это и позволяет облегчить решение всей

задачи синтеза автомата. Часто некоторые этапы синтеза не разделяются между собой, а иногда просто опускаются.
В ряде работ процесс синтеза излагается несколько иначе. В ряде работ изложена теория проектирования автоматов, основанная на семантике выполняемых преобразований.
Следует заметить, что наряду с изложенным выше делением процесса синтеза весь процесс проектирования сложного управляющего автомата чаще подразделяют на более крупные этапы: системное, программно-логическое, техническое и технологическое проектирование.
На этапе *системного проектирования,* включающего в себя блочный синтез автомата, определяют общую структуру АСУ, функции и назначение ее отдельных блоков. В результате выполнения этапа системного проектирования должны быть получены состав АСУ, структура соединений и общее *алгоритмическое описание* каждого из блоков.
На этапе *программно-логического проектирования* разрабатывают функциональные схемы каждого из блоков. При этом, если используется аппаратурная реализация автомата, то на этапе *логического проектирования* производят абстрактный синтез автомата, минимизацию числа его внутренних состояний и их кодирование, синтез функциональной схемы автомата, разрабатывают функциональные схемы каждого из блоков АСУ. При этом по общему алгоритмическому описанию выбирают принцип построения каждого из блоков и строят функциональные схемы в заданном базисе логических элементов. В результате выполнения этого логического проектирования должна быть получена для каждого блока АСУ функциональная схема, т. е. должно быть получено описание АСУ на языке *структурного описания.*
При программной реализации УА процесс логического синтеза, включающего частично второй и с третьего по пятый этапы, заменяют процессом программирования в базисе команд применяемого программного УА (микропроцессора, микроЭВМ и т. п.).
На этапе *технического проектирования* решаются следующие три главные задачи:
1) на основе функциональной схемы каждого из блоков производится разбиение логических элементов по корпусам с выбором типов корпусов (БИС или СБИС), корпусов по ТЭЗ, а ТЭЗ — по шкафам с учетом минимальной межкорпусной, межплатной и межшкафной проводности монтажа;
2) для полученных разбиений производится размещение элементов в корпусе (БИС или СБИС), корпусов — на ТЭЗ, а ТЭЗ — в

шкафах с учетом минимальной длины соединений и числа их пересечений;
3) для получения размещений производится трассировка печатных соединений на ТЭЗ и межплатных соединений в шкафу.
В результате выполнения этапа технического проектирования должна быть получена полная документация в установленной форме на изготовление УА.
При изготовлении УА на БИС и особенно СБИС этап технологического проектирования должен быть автоматизирован полностью практически на всех основных технологических операциях. В последнем случае после этапа технического проектирования должна быть выдана документация в таком виде, чтобы она могла быть использована непосредственно (без переработки вручную) в качестве исходной документации для выполнения автоматизированного технологического проектирования.
На основе опыта создания систем автоматизации проектирования ЭВМ и устройств автоматики можно выделить две основные разновидности таких систем: *автоматическая система сквозного проектирования и автоматизированная система проектирования.*
В первой системе не предполагается участие человека в процессе проектирования. Такая система требует введения только исходной информации, а само проектирование производится ЭВМ без вмешательства человека. Однако практика разработки таких систем показала, что не только при существующем парке ЭВМ, но и при парке ЭВМ в обозримый период времени эффективная система такого типа не может быть создана не только для всех этапов проектирования, но даже для каждого этапа в отдельности. Примером такой системы может служить система сквозного синтеза АВТОМАТ , основанная на алгоритмическом языке ЛЯПАС . Эга система может быть применена лишь для построения несложных ФБ.
В связи с этим практический интерес представляют системы автоматизации проектирования. В автоматизированной системе проектирования разумно сочетаются опыт и интуиция проектировщика с быстродействием ЭВМ, выполняющих громоздкие вычисления, благодаря чему проектировщик может рассмотреть большое число возможных вариантов.
Таким образом, в автоматизированной системе проектирования процесс проектирования основан на диалоге между человеком и машиной.

4.2. Канонический метод синтеза автоматов

В этой главе автоматы будут рассматриваться на структурном уровне или, другими словами, будет изучаться функционирование автоматов на уровне построения сложных схем из более простых. Основной задачей структурного синтеза автоматов является построение *функциональной (структурной) схемы* сложного автомата из более простых автоматов, которые называют *элементарными* автоматами, по графу исходного автомата. Более точно задачу можно сформулировать следующим образом.

Пусть задано некоторое конечное множество элементарных автоматов и задан произвольный конечный автомат А. Необходимо найти алгоритм, позволяющий по заданному графу или матрице соединений автомата строить некоторую композицию элементарных автоматов так, чтобы полученный в результате композиции автомат индуцировал отображение, совпадающее с отображением или продолжающее отображение, индуцируемое автоматом А.

(Здесь под композицией понимается отождествление (соединение) входных и выходных полюсов элементарных автоматов, в результате которою образуется схема сложного автомата из схем элементарных автоматов.)

Естественно, что не для всякого конечного множества элементарных автоматов данная задача разрешима. Систему элементарных автоматов называют *структурно полной,* если в пределах этой системы поставленная задача имеет решение.

Существует *два* класса элементарных автоматов: элементарные автоматы с памятью — *запоминающие элементы* и элементарные автоматы без памяти — *комбинационные* или *логические элементы.*

Обычно в качестве запоминающих элементов выбираются автоматы Мура, к которым предъявляются *требования полноты системы , переходов* и *полноты системы выходов.*

Требование полноты системы переходов предусматривает для любой упорядоченной пары состояний элементарного автомата наличие некоторого входного сигнала, который переводит первый элемент этой пары во второй.

Требование полноты системы выходов означает, что для каждого состояния элементарного автомата имеется соответствующий ему выходной сигнал, который отличается от выходных сигналов, соответствующих другим состояниям элементарного автомата. Если это требование не выполняется, то в общем случае нельзя отличить два различных состояния элементарного автомата Мура.

Что касается системы логических элементов, то к ней предъявляется требование функциональной полноты, которое предусматривает возможность реализации любой булевой функции с помощью логических элементов из данной системы.

Всякая система элементарных автоматов, которая содержит автомат Мура с нетривиальной памятью, обладающий полной системой переходов и полной системой выходов, и какую-нибудь функционально полную систему логических элементов, является структурно полной системой. Проблема полноты в классе автоматов Мура разработана в работах В. М. Глушкова, А. А. Летичевского , В. Б. Кудрявцева и др.

Существует общий конструктивный прием, называемый **каноническим методом структурного синтеза**, позволяющий свести задачу структурного синтеза произвольных конечных автоматов к задаче структурного синтеза логических схем. Остановимся подробнее на каноническом методе структурного синтеза автоматов.

Предварительно рассмотрим понятие кодирования состояний, входных и выходных сигналов конечного автомата. Пусть имеется автомат $A = (X,\ Q,\ Y,\ q_1 \in Q,\ F(x \in X / y \in Y))$ и набор запоминающих элементов $U_1, U_2, ..., U_p$, обладающих полной системой переходов и полной системой выходов. Пусть также произведение чисел состояний элементарных автоматов $U_1, U_2, ..., U_p$ не меньше числа состояний автомата A. Каждому состоянию q автомата A поставим в соответствие конечную упорядоченную последовательность ($U_1, U_2, ..., U_p$) состояний автоматов $U_1, U_2, ..., U_p$ так, что различным состояниям автомата A ставятся в соответствие различные последовательности состояний элементарных автоматов. Таким образом, состояния автомата A кодируются наборами состояний элементарных автоматов $U_1, U_2, ..., U_p$, в результате чего возникают структурные состояния автомата A. Так как при практическом синтезе схем используются, в основном, элементарные автоматы с двумя устойчивыми состояниями, которым присваиваются значения 0 и 1, то обычно состояния автомата A кодируются наборами $U_1,\ U_2,\ \ldots,\ U_p$ двоичных переменных $U_1, U_2, ..., U_p$, причем U_i — нулевое или единичное состояние элементарного автомата U_i. Такое кодирование называют двоичным. В дальнейшем будем пользоваться только двоичным кодированием при синтезе автоматов. Аналогично осуществляется кодирование входных и выходных сигналов автомата A.

Необходимо отметить, что в зависимости от того, каким образом выполнить *кодирование состояний, входных* и *выходных сигналов,* структурные схемы одного и того же автомата могут получиться

различными, так как каждому варианту кодирования соответствует структурная схема определенной сложности. Различным вариантам кодирования при одном и том же неизменном законе функционирования автомата соответствуют схемы различной сложности. Поэтому так называемая проблема кодирования состояний, входных и выходных сигналов заключается в том, чтобы из всего множества вариантов кодирования выбрать тот, которому соответствует *минимальная по сложности структурная схема автомата.* Более подробно вопросы кодирования в автоматах рассмотрены ранее. При каноническом методе синтеза обычно считают, что кодирование состояний, входных и выходных сигналов уже выполнено. Поэтому задача структурного синтеза автоматов сводится к выбору типов элементарных автоматов и отысканию такого способа их соединения между собой с помощью логических элементов, при котором структурная схема автомата функционирует в соответствии с заданными кодированными таблицами переходов и выходов. Рассмотрим задание этих таблиц на примере автомата, моделирующего условный рефлекс с забыванием. Для кодирования состояний этого автомата используем три двоичные переменные U_1, U_2, U_3. Пусть начальное состояние q_1 автомата закодировано набором трех двоичных переменных 000, состояние q_2 - набором 001 и т. д., q_7 — набором110, как это показано в таблице кодирования состояний (табл. 4.1).

Таблица 4.1

Q \ U	U_1	U_2	U_3
q_1	0	0	0
q_2	0	0	1
q_3	0	1	0
q_4	0	1	1
q_5	1	0	0
q_6	1	0	1
q_7	1	1	0

Таким образом, мы определили некоторый вариант кодирования состояний автомата. Пусть, далее, входные сигналы x_1, x_2 и x_3 закодированы некоторым образом двумя двоичными переменными $\tilde{m}_1$ и $\tilde{m}_2$ (табл. 4.2).

Таблица 4.2

X \ M	m_1	m_2
x_1	1	0
x_2	0	1
x_3	1	1

Кодированная таблица переходов этого автомата показана на табл. 4.3.

Таблица 4.3

№	1	2	3	4	5	6	7	8
	$m_1(t)$	$m_2(t)$	$U_1(t)$	$U_2(t)$	$U_3(t)$	$U_1(t+1)$	$U_2(t+1)$	$U_3(t+1)$
1	1	0	0	0	0	0	0	0
2	1	0	0	0	1	0	0	0
3	1	0	0	1	0	0	1	1
4	1	0	0	1	1	0	0	0
5	1	0	1	0	0	0	0	0
6	1	0	1	0	1	1	1	0
7	1	0	1	1	0	0	0	1
8	0	1	0	0	0	0	0	1
9	0	1	0	0	1	0	0	1
10	0	1	0	1	0	1	0	0
11	0	1	0	1	1	0	0	1
12	0	1	1	0	0	0	0	1
13	0	1	1	0	1	1	1	0
14	0	1	1	1	0	0	0	1
15	1	1	0	0	0	0	1	0
16	1	1	0	0	1	0	1	0
17	1	1	0	1	0	1	0	1
18	1	1	0	1	1	1	0	1
19	1	1	1	0	0	1	0	1
20	1	1	1	0	1	1	0	1
21	1	1	1	1	0	1	0	1

Эта таблица определяет зависимость состояний элементарных автоматов $U_1(t+1)$, $U_2(t+1)$, $U_3(t+1)$ в момент времени $t+1$ от значений закодированных входных сигналов $x_1(t)$, $x_2(t)$, $x_3(t)$ и

внутренних состояний $U_1(t)$, $U_2(t)$, $U_3(t)$ элементарных автоматов в предыдущий момент времени *t*.

Так как выходных сигналов в данном автомате всего два, то оникодируются одной двоичной переменной l: $y_2 \to l = 1$, $y_1 \to \bar{l} = 0$.

Заметим, что таблицы кодирования состояний, входных и выходных сигналов, а также кодированная таблица переходов совершенно одинаковы в автоматах Мили и Мура. Что же касается кодированной таблицы выходов, то они отличны друг от друга. В кодированной таблице выходов автомата Мили выходные переменные $l_1, l_2, \ldots, l_k$, наборами которых кодируются выходные сигналы, определены в зависимости от значений входного сигнала в момент времени *t* и внутренних состояний элементарных автоматов в тот же момент времени. Столбцы такой таблицы совпадали бы со столбцами 1, 2, 3, 4, 5 табл. 4.3, а вместо переменных $U_1(t+1)$, $U_2(t+1)$, $U_3(t+1)$ в остальных столбцах стояли бы переменные $l_1(t), l_2(t), \ldots, l_k(t)$.

В нашем примере автомат является автоматом Мура. Выходной сигнал, как известно, зависит только от состояний элементарных автоматов в момент времени *t*. Поэтому кодированная таблица имеет вид табл. 4.4.

Q \ U, l	U_1	U_2	U_3	l
q_1	0	0	0	0
q_2	0	0	1	1
q_3	0	1	0	1
q_4	0	1	1	0
q_5	1	0	0	1
q_6	1	0	1	1
q_7	1	1	0	1

Одна из основных задач, решаемых в процессе структурного синтеза автоматов, заключается в том, чтобы определить значение входных сигналов, которые необходимо подавать на каждый элементарный автомат и которые обеспечивают условия правильного функционирования автомата, заданные таблицами вида 4.3 и 4.4. Правильную работу элементарных автоматов обеспечивают с помощью так называемых *функции возбуждения элементарных автоматов* (ф.в.э. а.), которые реализуют зависимость входного сигнала $u_{ij}(t)$ на *i*-м входе элементарного автомата U_j от внутренних состояний всех элементарных автоматов в момент времени *t* и от входных сигналов

$x(t)$. Функции возбуждения элементарного автомата U_j, так же как и функция выходов $l_i(t)$, являются булевыми функциями и записываются в общем виде следующим образом:

$$u_{ij}(t) = f_1[U_1(t), \ldots, U_s(t), m_1(t), \ldots, m_r(t)], \quad (4.1)$$

$$l_i(t) = f_2[U_1(t), \ldots, U_s(t), m_1(t), \ldots, m_r(t)], \quad (4.2)$$

где s — число элементарных автоматов, а r — число двоичных переменных, кодирующих входные сигналы.

В качестве элементарных автоматов могут применяться различные типы запоминающих элементов: линии задержки, триггеры с раздельными и со счетным входами и их комбинациями. Выбор типов элементарных автоматов производится до получения ф. в. э. а. Вопрос рационального выбора элементарных автоматов рассмотрен, например, Е. Н. Вавиловым и Г. П. Портным.

Элементарные автоматы можно задавать таблицами переходов и матрицами переходов. Рассмотрим способ задания элементарных автоматов с помощью матриц переходов. Предварительно отметим, что в элементарных автоматах с двумя устойчивыми состояниями возможны только четыре типа переходов: из «0» в «0», из «0» в «1», из «1» в «0» и из «1» в «1». Для каждого из этих переходов имеются входные сигналы элементарного автомата, вызывающие этот переход. Если число входов автомата больше чем один, для некоторых переходов элементарного автомата значения входных сигналов на одном или нескольких входах могут не влиять на переход элементарного автомата. Такие сигналы будем обозначать неопределенными коэффициентами b_i.

Матрица переходов произвольного элементарного автомата имеет вид

$$\begin{array}{c|cccc} & x_1 & x_2 & \ldots & x_n \\ \hline 0 \to 0 & c_1^1 & c_2^1 & \ldots & c_n^1 \\ 0 \to 1 & c_1^2 & c_2^2 & \ldots & c_n^2 \\ 1 \to 0 & c_1^3 & c_2^3 & \ldots & c_n^3 \\ 1 \to 1 & c_1^4 & c_2^4 & \ldots & c_n^4 \end{array}. \quad (4.3)$$

В этой матрице строки помечены всеми возможными переходами элементарного автомата $U(t) \to U(t+1)$. Число столбцов равно числу входов элементарного автомата. Элемент матрицы c^i_j обозначает входной сигнал на j-м входе элементарного автомата, под действием которого происходит переход элементарного автомата, соответствующий i-й строке матрицы. Элемент матрицы c^i_j может принимать одно из трех значений: 0, 1 и b.

В качестве примера приведем матрицу переходов триггера с раздельными входами

$$\begin{array}{c|cc} & u_0 & u_1 \\ \hline 0\to 0 & b_1 & 0 \\ 0\to 1 & 0 & 1 \\ 1\to 0 & 1 & 0 \\ 1\to 1 & 0 & b_2 \end{array}. \quad (4.4)$$

Здесь u_0 и u_1 — входные сигналы, подаваемые соответственно на нулевой и единичный входы триггера. Переход триггера из нулевого состояния в нулевое осуществляется подачей на единичный вход триггера сигнала $u_1 = 0$. Неопределенный коэффициент b_1 показывает, что подача сигналов 0 либо 1 на нулевой вход не влияет на данный переход триггера. Это нетрудно понять из содержательного рассмотрения работы триггера.

Линия задержки имеет всего один вход, поэтому ее матрица переходов имеет вид

$$\begin{array}{c|c} & u \\ \hline 0\to 0 & 0 \\ 0\to 1 & 1 \\ 1\to 0 & 0 \\ 1\to 1 & 1 \end{array}. \quad (4.5)$$

Триггеру со счетным входом соответствует матрица переходов

$$\begin{array}{c|c} & u \\ \hline 0\to 0 & 0 \\ 0\to 1 & 1 \\ 1\to 0 & 1 \\ 1\to 1 & 0 \end{array}. \quad (4.6)$$

После выбора (или задания) типа элементарных автоматов по кодированной таблице переходов с использованием матриц переходов элементарных автоматов находят ф. в. э. а.

Для автомата, заданного табл. 4.3, в качестве элементарных автоматов выберем триггеры с раздельными входами и построим новую кодированную таблицу переходов (табл. 4 5), определяющую зависимость функций $u_{01}(t)$, $u_{11}(t)$, $u_{02}(t)$, $u_{12}(t)$, $u_{03}(t)$, $u_{13}(t)$ от типов переходов элементарных автоматов U_1, U_2, U_3 и от переменных m_1, m_2 следующим образом.

Таблица 4.5

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	$m_1(t)$	$m_2(t)$	$U_1(t)$	$U_2(t)$	$U_3(t)$	$U(t+1)$	$U_2(t+1)$	$U_3(t+1)$	$u_{01}(t)$	$u_{11}(t)$	$u_{02}(t)$	$u_{12}(t)$	$u_{03}(t)$	$u_{13}(t)$
1	1	0	0	0	0	0	0	0	b_1	0	b_1	0	b_1	0
2	1	0	0	0	1	0	0	0	b_2	0	b_2	0	1	0
3	1	0	0	1	0	0	1	1	b_3	0	0	b_5	0	1
4	1	0	0	1	1	0	0	0	b_4	0	1	0	1	0
5	1	0	1	0	0	0	0	0	1	0	b_4	0	b_2	0
6	1	0	1	0	1	1	1	0	0	b_5	0	1	1	0
7	1	0	1	1	0	0	0	1	1	0	1	0	0	1
8	0	1	0	0	0	0	0	1	b_6	0	b_5	0	0	1
9	0	1	0	0	1	0	0	1	b_7	0	b_6	0	0	b_3
10	0	1	0	1	0	1	0	0	0	1	1	0	b_4	0
11	0	1	0	1	1	0	0	1	b_8	0	1	0	0	b_7
12	0	1	1	0	0	0	0	1	1	0	b_7	0	0	1
13	0	1	1	0	1	1	1	0	0	b_9	0	1	1	0
14	0	1	1	1	0	0	0	1	1	0	1	0	0	1
15	1	1	0	0	0	0	1	0	b_{10}	0	0	1	b_6	0
16	1	1	0	0	1	0	1	0	b_{11}	0	0	1	1	0
17	1	1	0	1	0	1	0	1	0	1	1	0	0	1
18	1	1	0	1	1	1	0	1	0	1	1	0	0	b_7
19	1	1	1	0	0	1	0	1	0	b_{12}	b_8	0	0	1
20	1	1	1	0	1	1	0	1	0	b_{13}	b_9	0	0	b_8
21	1	1	1	1	0	1	0	1	0	b_{14}	1	0	0	1

Из матрицы переходов триггера с раздельными входами (4.4) следует, что для перевода первого элементарного автомата U_1 из состояния $U_1(t)$ =0 в состояние $U_1(t+1)$ =0 на единичный вход $u_{11}(t)$ необходимо подать сигнал, равный нулю, а на нулевом входе $u_{01}(t)$ значение сигнала произвольно, т. е. не определено. Так как в табл. 4.5 такие переходы для элементарного автомата U_1 имеются в 1, 2, 3, 4, 8, 9, 14, 15 и 16 строках, то на пересечении этих строк со столбцами 9 и 10 необходимо записать $u_{01}= b_i$, $u_{11}= 0$. На пересечении 10, 17 и 18 строк со столбцами 9 и 10 записаны входные сигналы $u_{01}= 0$ и $u_{11}= 1$, обеспечивающие переходы типа «0»→«1» элементарного автомата U_1. Аналогично заполняются клетки табл. 4.5 для остальных переходов первого элементарного автомата, а также столбцы, соответствующие второму и третьему элементарным автоматам.

Если бы в качестве элементарных автоматов были выбраны другие типы элементарных автоматов, то вместо матрицы переходов (4.4) нужно было бы использовать матрицы, соответствующие выбранному типу элементарных автоматов.

Результатом составления табл. 4.5 является получение ф.в. э. а., после чего производят минимизацию этих функций. Так как исходный автомат является частичным, то функции возбуждения будут представлять собой неполностью определенные булевы функции, поэтому их минимизация сводится, по существу, к минимизации неполностью определенных булевых функций.

Найдем минимальную форму ф. в. э. а. Для этого воспользуемся методом минимизации булевых функций с помощью диаграмм Вейча

(карт Карно). Диаграмма Вейча составляется для каждой функции возбуждения отдельно.

Таблица 4.6

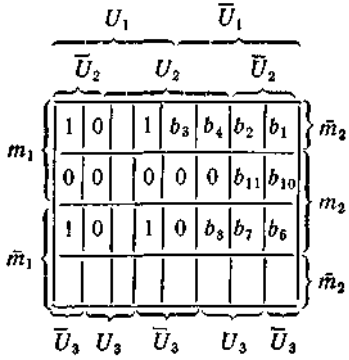
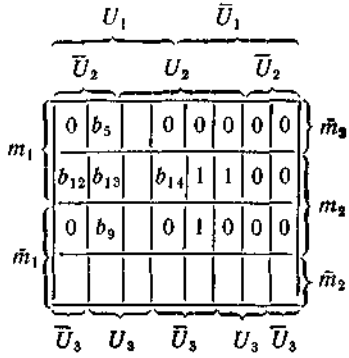


Таблица 4.7



В клетке диаграммы Вейча ставится 0, 1 либо b_i в зависимости от того, какое значение имеет функция возбуждения на наборе аргументов, соответствующем этой клетке диаграммы. В случае частичных автоматов некоторые клетки диаграммы оказываются незаполненными.

При минимизации они заполняются таким образом, чтобы функции возбуждения были минимальны.

На табл. 4.6 и 4.7 показаны примеры заполнения диаграмм Вейча для функций $u_{01}(t)$ и $u_{11}(t)$ соответственно. Приравнивая коэффициенты b_2, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}, b_{12}, b_{13} нулю, а коэффициенты b_1, b_3, b_{14}— единице и заполнив пустые клетки так, как это показано на табл. 4.8 и 4.9, получаем минимальные формы функций возбуждения первого элементарного автомата U_1:

$$u_{01}(t) = \bar{m}_2\bar{U}_3 \vee \bar{m}_1U_1\bar{U}_3,$$

$$u_{11}(t) = m_1m_2U_2 \vee m_2\bar{U}_1U_2\bar{U}_3.$$

Таблица 4.8

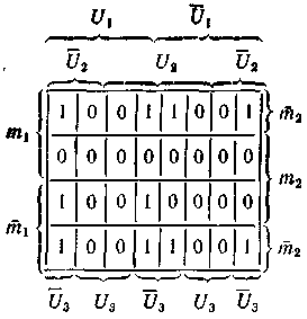
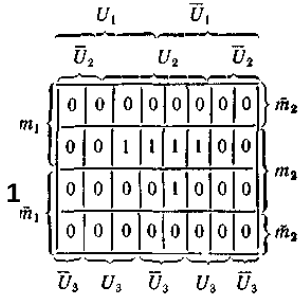


Таблица 4.9



1

Аналогично находим функции возбуждения элементарных автоматов U_2 и U_3, минимальные формы которых имеют вид

$$u_{02}(t) = U_2(m_2 \vee U_1 \vee U_3),$$
$$u_{12}(t) = (\bar{m}_1 \vee \bar{m}_2)\, U_1 U_3 \vee m_1 m_2 \bar{U}_1 \bar{U}_2,$$
$$u_{03}(t) = \bar{m}_2 U_3 \vee m_1 U_1 U_3 \vee m_1 \bar{U}_1 U_2,$$
$$u_{13}(t) = m_1 U_2 \bar{U}_3 \vee m_2 U_1 \bar{U}_3 \vee \bar{m}_1 \bar{U}_2 \bar{U}_3.$$

Получение функций выходов автомата в принципе не отличается от построения функций возбуждения. Если синтезируемый автомат является автоматом Мили, в кодированной таблице переходов и функций возбуждения надо было бы добавить столько столбцов, сколько двоичных переменных $l_1, l_2, \ldots, l_k$ потребовалось бы для кодирования выходных сигналов. Так как каждой строке такой таблицы соответствует подача входного сигнала x_i на автомат, находящийся в состоянии q_j, то эта строка отмечается соответствующим набором выходных переменных $l_1, l_2, \ldots, l_k$.Затем для каждой двоичной переменной l_i находится минимальное выражение таким же образом, как и для ф.в.э. а.

В том случае, когда синтезируемый автомат является автоматом Мура, функцию выходов можно получить непосредственно по таблице кодированных выходов (см., например, табл. 4.4). В нашем примере функция выходов представляется одной переменной l, которая является функцией трех аргументов — U_1, U_2, U_3. Диаграмма Вейча этой функции выглядит так, как показано на табл. 4.10. Доопределяя функцию, получим табл. 4.11.

Таблица 4.10

	U_1		$\bar{U}_1$	
	$\bar{U}_2$	U_2	$\bar{U}_2$	
U_3	1		0	1
$\bar{U}_3$	1	1	1	0

Таблица 4.11

	U_1		$\bar{U}_1$	
	$\bar{U}_2$	U_2	$\bar{U}_2$	
U_3	1	1	0	1
$\bar{U}_3$	1	1	1	0

Минимальная форма функции выходов имеет вид

$$l = U_1 \vee U_2 \bar{U}_3 \vee \bar{U}_2 U_3.$$

По функциям возбуждения и выходов нетрудно построить структурную схему автомата (рис. 4.1), моделирующего условный рефлекс с забыванием.

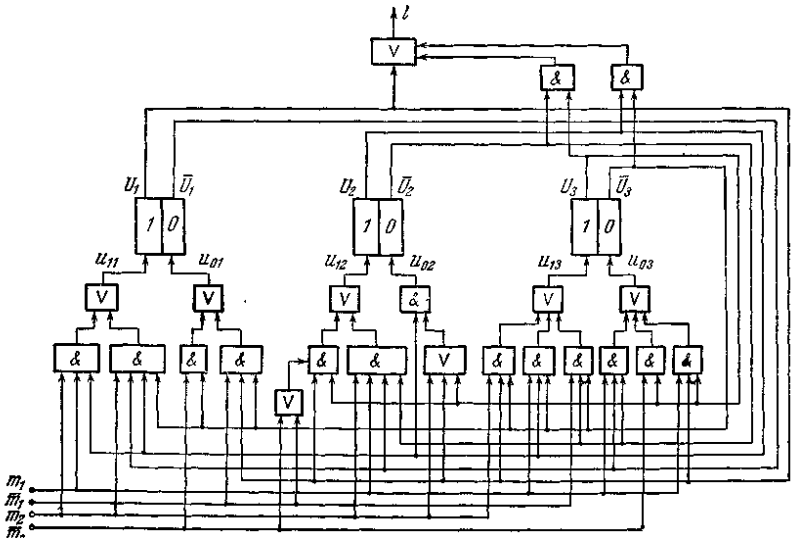


Рис. 4. 1.

На рис. 4.1 элементарные автоматы U_1, U_2, U_3 — тригеры с раздельными входами. На нулевые входы триггеров подаются сигналы, соответствующие функциям возбуждения $u_{0i}(t)$, на единичные входы — $u_{1i}(t)$. Эти функции реализованы с помощью функционально полной системы логических элементов в базисе $(\&, \vee, -)$. Каждая из ф.в.э. а. зависит как от переменных, кодирующих входные сигналы, так и от состояний элементарных автоматов. Прямые значения аргументов U_1, U_2, U_3 снимаются с единичных выходов триггеров, инверсные значения $\overline{U}_1, \overline{U}_2, \overline{U}_3$ — с нулевых выходов. Прямое значение сигнала l на выходе схемы представляет кодированный выходной сигнал y_2, а инверсное значение $\overline{l}$—выходной сигнал y_1. Входные сигналы автомата x_1, x_2, x_3 появляются в закодированном виде на входных полюсах $m_1, \overline{m}_1, m_2, \overline{m}_2$.

4.3. Построение функциональной схемы по графу автомата

Рассмотрим теперь метод получения функций возбуждения элементарных автоматов и функций выходов по графу автомата или по матрице соединений. Эта методика менее громоздка, чем построение функций возбуждения по кодированной таблице переходов автомата в

случае канонического метода синтеза, и очень удобна для перехода от этапа абстрактной декомпозиции к функциональной схеме автомата. Если автомат A содержит n состояний, то число элементарных автоматов, необходимых для его построения, как известно, определяется выражением $p \geqslant \log_2 n$.

Выберем в качестве элементарных автоматов множество автоматов $\{U_1, U_2, \ldots, U_p\}$ и закодируем внутренние состояния автомата A фиксированными наборами состояний множества элементарных автоматов произвольным образом. Например, состояние q_1 закодируем набором состояний $U_1, U_2, \ldots, U_{p-1}, U_p$, состояния q_2— набором $U_1, U_2, \ldots, U_{p-1}, \bar{U}_p$, состояние q_3 — набором $U_1, U_2, \ldots, \bar{U}_{p-1}, U_p$, состояние q_4 — набором $U_1, U_2 \ldots \ldots, \bar{U}_{p-1}, \bar{U}_p$ и т. д.

По матрице соединений автомата A запишем *обобщенные функции переходов элементарных автоматов,* которые имеют вид

$$U_i(t+1) = \varphi_i'(\tilde{U}_1(t), \ldots, \tilde{U}_p(t), z_1(t), \ldots, z_k(t)),$$

$$\bar{U}_i(t+1) = \varphi_i''(\tilde{U}_1(t), \ldots, \tilde{U}_p(t), z_1(t), \ldots, z_k(t)),$$

где $i \in I = \{1, 2, \ldots, p\}$,

$U_i(t+1)$ — единичное состояние i-го элементарного автомата в момент времени $t+1$;

$\bar{U}_i(t+1)$ — нулевое состояние i-го элементарного автомата в момент времени $t+1$;

$\tilde{U}_i(t)$ — нулевое или единичное состояние i-го элементарного автомата в момент времени t, которое входит в выражения с инверсией или без нее;

$Z_j(t)$ —входной сигнал в момент времени t.

Известно, что $U_i(t+1)$ и $\bar{U}_i(t+1)$ можно представить в следующей форме:

$$U_i(t+1) = g_{1i}'\bar{U}_i(t) \vee g_{1i}''U_i(t), \quad (4.7)$$

$$\bar{U}_i(t+1) = g_{0i}'U_i(t) \vee g_{0i}''\bar{U}_i(t), \quad (4.8)$$

где g_{1i}', g_{1i}'', g_{0i}', g_{0i}''— булевы функции аргументов $U_1(t), \ldots, U_{i-1}(t), U_{i+1}(t), \ldots, U_p(t), z_1(t), \ldots, z_k(t)$.

Из содержательного рассмотрения выражений (4.7) и (4.8) вытекает, что переключательные функции g_{1i}'' и g_{0i}'' могут принимать произвольные значения из множества {0, 1}, не влияя на правильное функционирование автомата. Поэтому значения g_{1i}'' и g_{0i}'' выбираются с целью получения минимальных форм функций возбуж-

дения элементарных автоматов. Учитывая это, выражения (4.7) и (4.8) запишем

$$U'_i(t+1) = g'_{1i}\bar{U}_i(t) \vee b_{1i}g''_{1i}U_i(t), \quad (4.9)$$

$$\bar{U}'_i(t+1) = g'_{0i}U_i(t) \vee b_{0i}g''_{0i}\bar{U}_i(t), \quad (4.10)$$

где b_{1i} и b_{0i} — неопределенные коэффициенты, которые при различных дизъюнктивных членах $b_{1i}g''_{1i}U_i(t)$ и $b_{0i}g''_{0i}\bar{U}_i(t)$ могут принимать различные независимые друг от друга значения из {0, 1}. Если в качестве элементарных автоматов выбираются триггеры с раздельными входами, то выражения (4.9) и (4.10) являются, по сути дела, функциями возбуждения элементарных автоматов, которые представляют работу вполне определенного абстрактного автомата, когда $\log_2 n$ — целое число. В том случае, если $\log_2 n$ не равен целому числу или исходный автомат является частичным, функции возбуждения будут иметь следующий вид:

$$u_{1i}(t) = g'_{1i}\bar{U}_i(t) \vee b_{1i}g''_{1i}U_i(t) \vee a_{1i}S,$$

$$u_{0i}(t) = g'_{0i}U_i(t) \vee b_{0i}g''_{0i}\bar{U}_i(t) \vee a_{0i}S,$$

где a_{1i} и a_{0i} - — неопределенные коэффициенты, а $S = \overline{U_i(t+1) \vee \bar{U}_i(t+1)}$ и вводится для получения минимальной формы функций возбуждения элементарных автоматов.

Из выражений (4.7) и (4.8) можно получить функцию возбуждения для триггера со счетным входом

$$u_{si}(t) = g'_{1i}\bar{U}_i(t) \vee g'_{0i}U_i(t) \vee b_iS.$$

По аналогии нетрудно записать выражения для функций возбуждения других типов элементарных автоматов.

Определим теперь по матрице соединений автомата функции выходов. С помощью переменных l_i выполним двоичное кодирование выходных сигналов $h_1, h_2, \ldots, h_r$ по аналогии с кодированием состояний таким образом, что каждая из выходных переменных l_i осуществляет разбиение множества выходных сигналов на два непересекающихся подмножества, которые кодируются через l_i и $\bar{l}_i$. Функции выходов $l_i(t)$ по матрице соединений автомата в общем виде можно записать следующим образом:

$$l_i(t) \bigvee_{\substack{j \in J \\ s \in S}} z_j q_s, \quad (4.11)$$

где $J = \{1, 2, \ldots, k\}$, $S = \{1, 2, \ldots, n\}$, а $z_j q_s$ — пара вход — состояние, отмеченная выходной буквой h, входящей в подмножество l_i. Заметим, что в l_i, будет входить столько дизъюнктивных членов, сколько содержится в матрице соединений ненулевых элементов,

отмеченных выходными буквами из подмножества l_i . Функции выходов (4.11) являются булевыми функциями и их минимизация выполняется известными методами.

Рассмотрим построение функциональной схемы по графу автомата на примере.

Пример 4.1. Пусть дан автомат *A,* показанный на рис. 4.2.

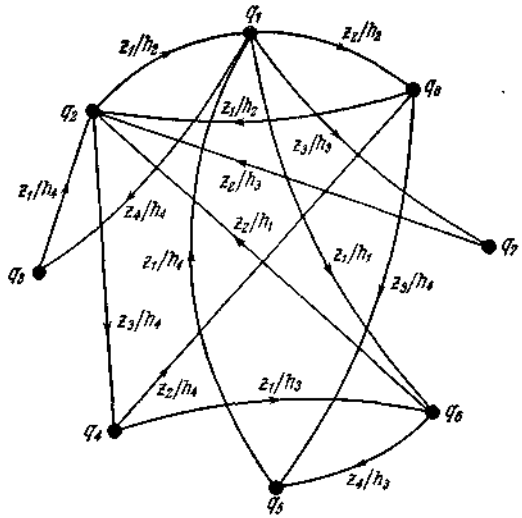


Рис. 4.2.

Требуется найти функции возбуждения элементарных автоматов и функции выходов и построить функциональную схему автомата *A.* В качестве элементарных автоматов использовать триггеры с раздельными входами, а в качестве функционально полного набора логических элементов — элементы, реализующие функции

$(\&, \vee, -)$.

Выбираем множество из трех элементарных автоматов $\{U_1, U_2, U_3\}$ и наборами состояний элементарных автоматов закодируем внутренние состояния автомата *A* на матрице соединений *R* следующим образом:

$$R = \begin{array}{c|c|cccc|cccc}
 & & U_1 & & & & \bar{U}_1 & & & \\
 & & U_2 & & \bar{U}_2 & & U_2 & & \bar{U}_2 & \\
 & & U_3 & \bar{U}_3 & U_3 & \bar{U}_3 & U_3 & \bar{U}_3 & U_3 & \bar{U}_3 \\
 & & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \\
\hline
U_1 & U_2\ U_3\ q_1 & 0 & 0 & z_4/h_4 & 0 & 0 & z_1/h_1 & z_3/h_3 & z_2/h_2 \\
 & U_2\ \bar{U}_3\ q_2 & z_1/h_2 & 0 & 0 & z_3/h_4 & 0 & 0 & 0 & 0 \\
 & \bar{U}_2\ U_3\ q_3 & 0 & z_1/h_4 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & \bar{U}_2\ \bar{U}_3\ q_4 & 0 & 0 & 0 & 0 & 0 & z_1/h_3 & 0 & z_2/h_4 \\
\hline
\bar{U}_1 & U_2\ U_3\ q_5 & z_1/h_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & U_2\ \bar{U}_3\ q_6 & 0 & z_2/h_1 & 0 & 0 & z_4/h_3 & 0 & 0 & 0 \\
 & \bar{U}_2\ U_3\ q_7 & 0 & z_2/h_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & \bar{U}_2\ \bar{U}_3\ q_8 & 0 & z_1/h_2 & 0 & 0 & z_3/h_4 & 0 & 0 & 0
\end{array}.$$

Выбирая две двоичные переменные m_1 и m_2 для кодирования входных сигналов, будем иметь

$$z_1 \to m_1 m_2, \quad z_2 \to m_1 \bar{m}_2, \quad z_3 \to \bar{m}_1 m_2, \quad z_4 \to \bar{m}_1 \bar{m}_2.$$

По матрице соединений R записываем обобщенную функцию переходов элементарного автомата U_1:

$$U_1(t+1) = \bar{m}_1\bar{m}_2 U_1 U_2 U_3 \vee m_1 m_2 U_1 U_2 \bar{U}_3 \vee \bar{m}_1 m_2 U_1 U_2 \bar{U}_3 \vee$$
$$\vee m_1 m_2 U_1 \bar{U}_2 U_3 \vee m_1 m_2 \bar{U}_1 U_2 U_3 \vee m_1 \bar{m}_2 \bar{U}_1 U_2 \bar{U}_3 \vee$$
$$\vee m_1 \bar{m}_2 \bar{U}_1 \bar{U}_2 U_3 \vee m_1 m_2 \bar{U}_1 \bar{U}_2 \bar{U}_3,$$

$$\bar{U}_1(t+1) = m_1 m_2 U_1 U_2 U_3 \vee \bar{m}_1 m_2 U_1 U_2 U_3 \vee m_1 \bar{m}_2 U_1 U_2 U_3 \vee$$
$$\vee m_1 m_2 U_1 \bar{U}_2 \bar{U}_3 \vee m_1 \bar{m}_2 U_1 \bar{U}_2 \bar{U}_3 \vee \bar{m}_1 \bar{m}_2 \bar{U}_1 U_2 \bar{U}_3 \vee$$
$$\vee \bar{m}_1 m_2 \bar{U}_1 \bar{U}_2 \bar{U}_3.$$

Отсюда получаем функции возбуждения первого элементарного автомата

$$u_{11}(t) = m_1 m_2 \bar{U}_1 U_2 U_3 \vee m_1 \bar{m}_2 \bar{U}_1 U_2 \bar{U}_3 \vee m_1 \bar{m}_2 \bar{U}_1 \bar{U}_2 U_3 \vee$$
$$\vee m_1 m_2 \bar{U}_1 \bar{U}_2 \bar{U}_3 \vee b_1 \bar{m}_1 \bar{m}_2 U_1 U_2 U_3 \vee b_2 m_1 m_2 U_1 U_2 \bar{U}_3 \vee$$
$$\vee b_3 \bar{m}_1 m_2 U_1 U_2 \bar{U}_3 \vee b_4 m_1 m_2 U_1 \bar{U}_2 U_3 \vee b_{11} S,$$

$$u_{01}(t) = m_1 m_2 U_1 U_2 U_3 \vee \bar{m}_1 m_2 U_1 U_2 U_3 \vee m_1 \bar{m}_2 U_1 U_2 U_3 \vee$$
$$\vee m_1 m_2 \bar{U}_1 \bar{U}_2 \bar{U}_3 \vee m_1 \bar{m}_2 U_1 \bar{U}_2 \bar{U}_3 \vee b_1 \bar{m}_1 \bar{m}_2 \bar{U}_1 U_2 \bar{U}_3 \vee$$
$$\vee b_2 \bar{m}_1 m_2 \bar{U}_1 \bar{U}_2 \bar{U}_3 \vee b_{01} S,$$

где $S = \overline{U_1(t+1) \vee \bar{U}_1(t+1)}$.

Аналогично можно получить функции возбуждения элементарных автоматов U_2 и U_3. После минимизации функции возбуждения элементарных автоматов U_1, U_2 и U_3 примут следующий вид:

$$u_{11}(t) = m_1\bar{U}_1, \qquad u_{01}(t) = \bar{m}_1 m_2 U_3 \vee m_1 U_1 U_2 U_3 \vee U_1\bar{U}_2\bar{U}_3,$$

$$u_{12}(t) = \bar{U}_1 \vee m_2\bar{U}_2, \qquad u_{02}(t) = U_1(\bar{m}_1 \vee \bar{m}_2),$$

$$u_{13}(t) = \bar{m}_1\bar{U}_1 \vee m_1 m_2 U_2\bar{U}_3, \qquad u_{03}(t) = m_1 U_1 U_3 \vee m_1\bar{m}_2.$$

Покажем получение функций выходов автомата A по матрице соединений R. Пусть кодирование выходных сигналов h_1, h_2, h_3, h_4 выполнено с помощью двух переменных l_1 и l_2 следующим образом:

$h_1 \to l_1 l_2$

$h_2 \to l_1\bar{l}_2$,

$h_3 \to \bar{l}_1 l_2$,

$h_4 \to \bar{l}_1\bar{l}_2$.

Так как подмножество выходных сигналов, отмеченное прямым значением переменной l_1, состоит из двух выходных сигналов h_1 и h_2, то функция $l_1(t)$ содержит все пары $z_j q_s$, отмеченные на матрице соединений R выходными сигналами h_1 и h_2. Получаем

$$l_1(t) = z_1 q_1 \vee z_2 q_1 \vee z_1 q_2 \vee z_2 q_6 \vee z_1 q_8.$$

Аналогично запишем

$$\bar{l}_1(t) = z_4 q_1 \vee z_3 q_1 \vee z_3 q_2 \vee z_1 q_3 \vee z_1 q_4 \vee z_2 q_4 \vee$$
$$\vee z_1 q_5 \vee z_4 q_6 \vee z_2 q_7 \vee z_3 q_8.$$

Подставив вместо z_j и q_s их кодированные значения, получим

$$l_1(t) = m_1 m_2 U_1 U_2 U_3 \vee m_1\bar{m}_2 U_1 U_2 U_3 \vee m_1 m_2 U_1 U_2\bar{U}_3 \vee$$
$$\vee m_1\bar{m}_2\bar{U}_1 U_2\bar{U}_3 \vee m_1 m_2\bar{U}_1\bar{U}_2\bar{U}_3,$$

$$l(t) = \bar{m}_1\bar{m}_2 U_1 U_2 U_3 \vee \bar{m}_1 m_2 U_1 U_2 U_3 \vee \bar{m}_1 m_2 U_1 U_2\bar{U}_3 \vee$$
$$\vee m_1 m_2 U_1\bar{U}_2 U_3 \vee m_1 m_2 U_1\bar{U}_2\bar{U}_3 \vee m_1\bar{m}_2 U_1\bar{U}_2\bar{U}_3 \vee$$
$$\vee m_1 m_2\bar{U}_1 U_2 U_3 \vee \bar{m}_1\bar{m}_2\bar{U}_1 U_2\bar{U}_3 \vee m_1\bar{m}_2\bar{U}_1\bar{U}_2 U_3 \vee$$
$$\vee \bar{m}_1 m_2\bar{U}_1\bar{U}_2\bar{U}_3.$$

После минимизации находим следующую форму функции выходов:

$$l_1(t) = m_1 U_1 U_2 \vee m_1\bar{U}_1\bar{U}_3 = m_1(U_1 U_2 \vee \bar{U}_1\bar{U}_3).$$

Аналогично получаем минимальную форму функции выходов

$$l_2(t) = \bar{m}_2\bar{U}_1 \vee m_2 U_1(U_2 U_3 \vee \bar{U}_2\bar{U}_3).$$

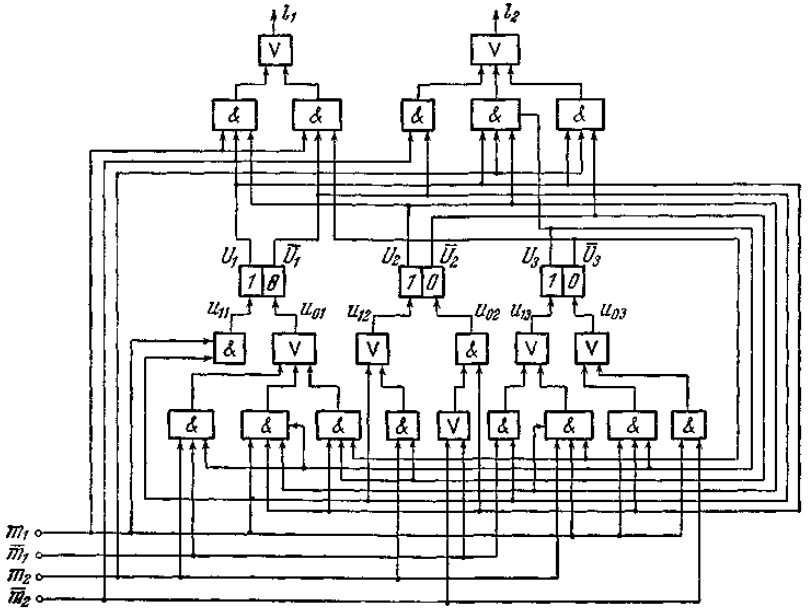


Рис. 4. 3.

По найденным функциям возбуждения элементарных автоматов и функциям выходов можно построить функциональную схему автомата *A,* которая показана на рис. 4.3.

4. 4. Декомпозиционный метод синтеза автоматов

Ранее была рассмотрена оптимальная декомпозиция произвольных абстрактных автоматов на элементарные абстрактные автоматы, в качестве которых могут быть выбраны любые абстрактные автоматы с простым числом состояний, в частности, автоматы с двумя состояниями. В результате декомпозиции получаем матрицы соединений элементарных абстрактных автоматов, совместная работа которых эквивалентна функционированию исходного абстрактного автомата. Выбирая конкретные типы элементарных автоматов (триггеры, линии задержки и т. п.), по матрицам соединений элементарных абстрактных автоматов легко построить функции возбуждения и функции выходов конкретных элементарных автоматов, представляющих работу исходного автомата. Получение функций возбуждения и выходов конкретных элементарных автоматов, как известно, является заклю-

чительным этапом структурного синтеза, так как по ним однозначно определяется структурная схема автомата. Поэтому структурный синтез автоматов, по сути дела, выносится на абстрактный этап и сводится к оптимальной декомпозиции автомата на элементарные абстрактные автоматы и записи функций возбуждения и функций выходов конкретных элементарных автоматов по матрицам соединений элементарных абстрактных автоматов.

Преимущество декомпозиционного метода синтеза автоматов по сравнению с каноническим методом заключается, во-первых, в том, что не требуется строить сложную кодированную таблицу переходов, во-вторых, решается проблема оптимального кодирования состояний автомата, которая приводит к минимальной комбинационной части функциональной схемы автомата, и, в-третьих, он позволяет строить оптимальную или близкую к ней функциональную схему автомата при использовании элементарных автоматов со многими устойчивыми состояниями (например, тремя, пятью и т. д.) и логическими элементами в недвоичной логике (троичной, пятеричной и т. д.). Другими словами, декомпозиционный метод позволяет осуществлять синтез цифровых автоматов при использовании логики более высокого порядка по сравнению с двоичной.

Рассмотрим применение декомпозиционного метода синтеза на примере построения схем двух автоматов, один из которых является автоматом Мура, а второй — автоматом Мили.

Пример 4.2. Построить оптимальную структурную схему автомата А, моделирующего выработку условного рефлекса с забыванием, отмеченная таблица переходов которого имеет вид таблицы:

Y	y_1	y_2	y_2	y_1	y_1	y_2	y_2
X \ Q	q_1	q_2	q_3	q_4	q_5	q_6	q_7
x_1	q_1	q_1	q_4	q_1	q_1	q_7	q_2
x_2	q_2	q_2	q_5	q_2	q_2	q_7	q_2
x_3	q_3	q_3	q_6	q_6	q_6	q_6	q_6

Дополним таблицу переходов автомата *А* одним изолированным состоянием и запишем матрицу соединений, которая будет иметь вид

$$R_A = \begin{array}{c} \begin{array}{cccccccc} (y_1) & (y_2) & (y_2) & (y_1) & (y_2) & (y_2) & (y_2) & (-) \end{array} \\ \left\| \begin{array}{cccccccc} x_1 & x_2 & x_3 & 0 & 0 & 0 & 0 & 0 \\ x_1 & x_2 & x_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & x_2 & x_3 & 0 & 0 \\ x_1 & x_2 & 0 & 0 & 0 & x_3 & 0 & 0 \\ x_1 & x_2 & 0 & 0 & 0 & x_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_3 & x_1 \vee x_2 & 0 \\ 0 & x_1 \vee x_2 & 0 & 0 & 0 & x_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right\| \end{array}.$$

Если привести матрицу R_A к виду правильной клеточной матрицы с запретами, то она будет содержать 39 запрещенных переходов. Найдем пару (π, ρ) разбиений множества состояний автомата *A,* определяющую подстановку, преобразующую матрицу соединений R_A к виду квазиправильной матрицы соединений. Используя извесный прием, находим ρ разбиение, в каждом классе которого содержится максимальное число состояний с тождественными переходами (так как рассматривается автомат Мура, то выходы во внимание можно не принимать.)

Получаем

$$\rho = \{\rho_1, \rho_2, \rho_3, \rho_4\}, \quad \rho_1 = \{q_1, q_2\}, \quad \rho_2 = \{q_3, q_8\},$$
$$\rho_3 = \{q_4, q_5\}, \quad \rho_4 = \{q_6, q_7\}.$$

После этого находим π разбиение, которое имеет вид

$$\pi = \{\pi_1, \pi_2\}, \quad \pi_1 = \{q_1, q_3, q_4, q_6\}, \quad \pi_2 = \{q_2, q_8, q_5, q_7\}.$$

На этом заканчивается первый шаг декомпозиции. Для приведения второй матрицы соединений (после разбиения R_A на две матрицы) к виду квазиправильной матрицы соединений упорядочим классы ρ разбиения, т. е. выполним второй шаг декомпозиции. Строим пару (π', ρ') разбиения множества классов разбиения ρ. Получаем

$$\rho' = \{\rho'_1, \rho'_2\}, \quad \rho'_1 = \{\rho_1, \rho_2\}, \quad \rho'_2 = \{\rho_3, \rho_4\},$$
$$\pi' = \{\pi'_1, \pi'_2\}, \quad \pi'_1 = \{\rho_1, \rho_3\}, \quad \pi'_2 = \{\rho_2, \rho_4\}.$$

В результате находим следующую подстановку классов ρ разбиения:

$$t_1 = \begin{pmatrix} \rho_1 & \rho_2 & \rho_3 & \rho_4 \\ \rho_1 & \rho_3 & \rho_2 & \rho_4 \end{pmatrix},$$

откуда следует, что необходимо брать классы ρ разбиения в следующей упорядоченности:

$$\rho_1 = \{q_1, q_2\}, \quad \rho_2 = \{q_4, q_5\}, \quad \rho_3 = \{q_3, q_8\}, \quad \rho_4 = \{q_6, q_7\}.$$

Пересекая классы π и нового ρ разбиения, окончательно получаем следующие подстановки:

$$t=\begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \\ q_1 & q_4 & q_3 & q_6 & q_2 & q_5 & q_8 & q_7 \end{pmatrix}$$

и

$$t^{-1}=\begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \\ q_1 & q_5 & q_3 & q_2 & q_6 & q_4 & q_8 & q_7 \end{pmatrix}.$$

Применяя подстановку t^{-1} к матрице R_A и преобразуя ее в квазиправильную матрицу соединений, приходим к матрице

$$R'_A=\left\|\begin{array}{cccc|cccc} x_1 & 0 & x_3 & 0 & \boxed{x_1}\vee x_2 & 0 & 0 & 0 \\ x_1 & 0 & 0 & x_3 & \boxed{x_1}\vee x_2 & 0 & 0 & 0 \\ 0 & x_1 & 0 & x_3 & 0 & \boxed{x_1}\vee x_2 & 0 & 0 \\ \boxed{x_1} & 0 & 0 & \boxed{x_1}\vee x_3 & \boxed{x_1}\vee\boxed{x_2} & 0 & 0 & x_1\vee x_2 \\ \hline x_1 & 0 & x_3 & 0 & \boxed{x_1}\vee x_2 & 0 & 0 & 0 \\ x_1 & 0 & 0 & x_3 & \boxed{x_1}\vee x_2 & 0 & 0 & 0 \\ 0 & \boxed{x_1} & 0 & \boxed{x_3} & 0 & \boxed{x_1}\vee\boxed{x_2} & 0 & 0 \\ \boxed{x_1} & 0 & 0 & \boxed{x_1}\vee x_3 & x_1\vee x_2 & 0 & 0 & \boxed{x_1}\vee\boxed{x_2} \end{array}\right\|,$$

которая содержит 17 запрещенных переходов.

Разложим матрицу R'_A на две матрицы R_1 и R'_2 с отмеченными переходами. Тогда получим

$$R_1=\begin{array}{c} \\ v_1 \\ v_2 \end{array}\left\|\begin{array}{cc} \underset{\sim}{x_1}\vee x_3 & \underset{\sim}{x_1}\vee\underset{\sim}{x_2} \\ \underset{\sim}{x_1}\vee x_3 & \underset{\sim}{x_1}\vee\underset{\sim}{x_2} \end{array}\right\|,$$

$$R'_2=\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{array}\left\|\begin{array}{cccc} x_1\vee x_2 & 0 & x_3 & 0 \\ x_1\vee x_2 & 0 & 0 & x_3 \\ 0 & x_1\vee x_2 & 0 & x_3 \\ \underset{\sim}{x_1}\vee\underset{\sim}{x_2} & 0 & 0 & x_1\vee x_2\vee x_3 \end{array}\right\|.$$

После введения связей в матрицы R_1 и R'_2 запишем их в виде

$$R_1 = \begin{matrix} v_1 \\ v_2 \end{matrix}\left\| \begin{matrix} (w_1 \vee w_2 \vee w_3)\, x_1 \vee x_3 & w_4 x_1 \vee x_2 \\ w_1 (x_1 \vee x_3) \vee w_2 (x_1 \vee x_3) \vee w_4 x_3 & w_1 x_2 \vee w_2 x_2 \vee w_1 (x_1 \vee x_2) \end{matrix} \right\|,$$

$$R_2' = \begin{matrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{matrix}\left\| \begin{matrix} x_1 \vee x_2 & 0 & x_3 & 0 \\ x_1 \vee x_2 & 0 & 0 & x_3 \\ 0 & v_1 x_1 \vee v_1 x_2 & 0 & v_1 x_3 \\ v_2 x_1 \vee v_2 x_2 & 0 & 0 & v_1 x_1 \vee v_1 x_2 \vee x_3 \end{matrix} \right\|.$$

Приведем матрицу R'_2 к правильной клеточной матрице с запретами

$$R_2' = \begin{matrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{matrix}\left\| \begin{matrix} v_1x_1 \vee v_2x_1 \vee \\ \vee v_1x_2 \vee v_2x_2 & \boxed{v_1x_1} \vee \boxed{v_1x_2} & v_1x_3 & \boxed{v_1x_1} \\ v_1x_1 \vee v_2x_1 \vee \\ \vee v_1x_2 \vee v_2x_2 & \boxed{v_1x_1} \vee \boxed{v_1x_2} & 0 & v_1x_3 \\ \boxed{v_1x_1} \vee \boxed{v_2x_1} \vee \\ \vee \boxed{v_1x_2} \vee \boxed{v_2x_2} & v_1x_1 \vee v_1x_2 & \boxed{v_1x_1} \vee \boxed{v_1x_2} \vee \\ \vee \boxed{v_1x_3} & \boxed{v_1x_1} \vee \boxed{v_1x_2} \vee \\ \vee v_1x_3 \\ \boxed{v_1x_1} \vee v_2x_1 \vee \\ \vee \boxed{v_1x_2} \vee v_2x_2 & \boxed{v_1x_1} \vee \boxed{v_1x_2} & \boxed{v_1x_1} \vee \boxed{v_1x_2} & v_1x_1 \vee v_1x_2 \vee v_1x_3 \end{matrix} \right\|.$$

Разложим матрицу R'_2 на две матрицы R_2 и R_3 с отмеченными переходами

$$R_2 = \begin{matrix} h_1 \\ h_2 \end{matrix}\left\| \begin{matrix} v_1x_1 \vee v_2x_1 \vee v_1x_2 \vee v_2x_2 & v_1x_3 \\ v_1x_1 \vee v_2x_1 \vee v_1x_2 \vee v_2x_2 & v_1x_1 \vee v_1x_2 \vee v_1x_3 \end{matrix} \right\|,$$

$$R_3 = \begin{matrix} s_1 \\ s_2 \end{matrix}\left\| \begin{matrix} v_1x_1 \vee v_2x_1 \vee v_1x_2 \vee v_2x_2 \vee v_1x_3 & v_1x_1 \vee v_1x_2 \vee v_1x_3 \\ v_1x_1 \vee v_2x_1 \vee v_1x_2 \vee v_2x_2 & v_1x_1 \vee v_1x_2 \vee v_1x_3 \end{matrix} \right\|.$$

Вводя связи в матрицы R_2 и R_3, получим

$$R_2 = \begin{matrix} h_1 \\ h_2 \end{matrix}\left\| \begin{matrix} v_1x_1 \vee v_2x_1 \vee v_1x_2 \vee v_2x_2 & v_1x_3 \\ s_1v_1x_1 \vee s_2v_2x_1 \vee s_1v_1x_2 \vee s_2v_2x_2 & s_2v_1x_1 \vee s_2v_1x_2 \vee v_1x_3 \end{matrix} \right\|,$$

$$R_3 = \begin{matrix} s_1 \\ s_2 \end{matrix}\left\| \begin{matrix} h_1v_1x_1 \vee h_1v_2x_1 \vee h_1v_1x_2 \vee \\ \vee h_1v_2x_2 \vee h_1v_1x_3 & h_2v_1x_1 \vee h_2v_1x_2 \vee h_2v_1x_3 \\ h_1v_1x_1 \vee v_2x_1 \vee h_1v_1x_2 \vee v_2x_2 & h_2v_1x_1 \vee h_2v_1x_2 \vee h_2v_1x_3 \end{matrix} \right\|.$$

Таким образом, в результате декомпозиции автомата A получены три элементарных абстрактных автомата, заданных матрицами соединений R_1, R_2 и R_3, совместная работа которых определяет работу автомата A. Закодируем состояния элементарных абстрактных автоматов и входные буквы автомата A следующим образом:

$$x_1 \to m_1 m_2,$$

$$v_1 \to U_1, \quad h_1 \to U_2, \quad s_1 \to U_3, \quad x_2 \to m_1 \bar{m}_2,$$

$$v_2 \to \bar{U}_1, \quad h_2 \to \bar{U}_2, \quad s_2 \to \bar{U}_3, \quad x_3 \to \bar{m}_1 m_2.$$

Тогда матрицы соединений элементарных абстрактных автоматов можно записать в виде

$$R_1 = \begin{array}{c|c|c} & \overbrace{}^{U_1} & \overbrace{\phantom{\bar{U}_1}}^{\bar{U}_1} \\ U_1 & U_2U_3m_1m_2 \vee U_2\bar{U}_3m_1m_2 \vee \vee \bar{U}_2U_3m_1m_2 \vee \bar{m}_1m_2 & \bar{U}_2\bar{U}_3m_1m_2 \vee m_1\bar{m}_2 \\ \bar{U}_1 & U_2U_3m_1m_2 \vee U_2U_3\bar{m}_1m_2 \vee \vee U_2\bar{U}_3m_1m_2 \vee \vee U_2\bar{U}_3\bar{m}_1m_2 \vee \bar{U}_2\bar{U}_3\bar{m}_1m_2 & \bar{U}_2\bar{U}_3m_1m_2 \vee U_2U_3m_1\bar{m}_2 \vee \vee U_2\bar{U}_3m_1\bar{m}_2 \vee \bar{U}_2\bar{U}_3m_1\bar{m}_2 \end{array},$$

$$R_2 = \begin{array}{c|c|c} & \overbrace{}^{U_2} & \overbrace{\phantom{\bar{U}_2}}^{\bar{U}_2} \\ U_2 & U_1m_1m_2 \vee \bar{U}_1m_1m_2 \vee \vee U_1m_1\bar{m}_2 \vee \bar{U}_1m_1\bar{m}_2 & U_1\bar{m}_1m_2 \\ \bar{U}_2 & U_1U_3m_1m_2 \vee \bar{U}_1\bar{U}_3m_1m_2 \vee \vee U_1U_3m_1\bar{m}_2 \vee \bar{U}_1\bar{U}_3m_1\bar{m}_2 & U_1\bar{U}_3m_1m_2 \vee U_1\bar{U}_3m_1\bar{m}_2 \vee \vee U_1\bar{m}_1m_2 \end{array},$$

$$R_3 = \begin{array}{c|c|c} & \overbrace{}^{U_3} & \overbrace{\phantom{\bar{U}_3}}^{\bar{U}_3} \\ U_3 & U_1U_2m_1m_2 \vee \bar{U}_1U_2m_1m_2 \vee \vee U_1U_2m_1\bar{m}_2 \vee \vee \bar{U}_1U_2m_1\bar{m}_2 \vee U_1U_2\bar{m}_1m_2 & U_1\bar{U}_2m_1m_2 \vee U_1\bar{U}_2m_1\bar{m}_2 \vee \vee U_1\bar{U}_2\bar{m}_1m_2 \\ \bar{U}_3 & U_1U_2m_1m_2 \vee \bar{U}_1m_1m_2 \vee \vee U_1U_2m_1\bar{m}_2 \vee \bar{U}_1m_1\bar{m}_2 & U_1\bar{U}_2m_1m_2 \vee U_1\bar{U}_2m_1\bar{m}_2 \vee \vee U_1\bar{U}_2\bar{m}_1m_2 \end{array}.$$

Выбирая в качестве элементарных автоматов триггеры с раздельными входами, по матрицам соединений элементарных абстрактных автоматов можно построить функции возбуждения конкретных элементарных автоматов (триггеров), которые определяются дизъюнкцией соответствующих элементов столбцов матриц соединений (разумеется, с учетом неопределенных коэффициентов b_i). По матрицам R_1, R_2 и R_3 после минимизации получаем функции возбуждения триггеров, которые имеют вид

$$u_{11} = \bar{m}_1 \vee m_2 U_2,$$
$$u_{01} = \bar{m}_2 \vee m_1 \bar{U}_2 \bar{U}_3,$$
$$u_{12} = m_1 (\bar{U}_1 \vee U_3),$$
$$u_{02} = \bar{m}_1,$$
$$u_{13} = m_1 (\bar{U}_1 \vee U_2),$$
$$u_{03} = U_1 \bar{U}_2.$$

Найдем функцию выходов автомата *A.* Закодируем выходной сигнал y_2 через l, а сигнал y_1 через $\bar{l}$. Так как A — автомат Мура, то, используя естественное кодирование состояний по матрице R'_A и учитывая подстановку t, преобразующую R'_A в R_A, функцию выходов l запишем в форме

$$l = U_1 U_2 \bar{U}_3 \vee U_1 \bar{U}_2 U_3 \vee U_1 \bar{U}_2 \bar{U}_3 \vee \bar{U}_1 U_2 U_3 \vee$$
$$\vee \bar{U}_1 \bar{U}_2 U_3 \vee \bar{U}_1 \bar{U}_2 \bar{U}_3,$$

которая после минимизации принимает вид

$$l = \bar{U}_2 \vee \bar{U}_1 U_3 \vee U_1 \bar{U}_3.$$

По полученным функциям возбуждения и функции выходов можно построить структурную схему автомата, которая показана на рис. 4.4.

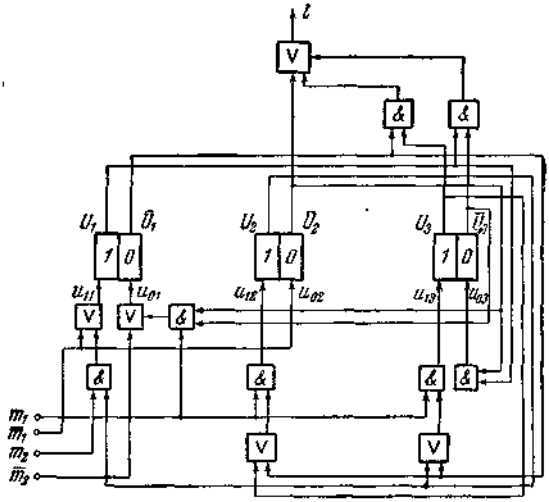


Рис. 4.4.

Пример 4.3. Требуется синтезировать автомат Мили — схему дешифратора последовательного действия, на вход которого

поступают трехразрядные двоичные числа. Обозначим входные сигналы через $x_1 = 1$ и $x_2 = 0$. Выходной алфавит Y автомата состоит из девяти букв $Y = \{y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9\}$. При подаче на вход дешифратора различных триад на выходе его появляются следующие сигналы:

Вход:	000	001	010	011	100	101	110	111
Выход:	y_8	y_7	y_6	y_5	y_4	y_3	y_2	y_1

причем считаем, что младшие разряды расположены слева. При подаче любых других последовательностей, которые могут состоять из одной или двух входных букв, на выходе дешифратора появляется сигнал y_9. Матрица соединений автомата, полученная в результате этапа абстрактного синтеза и дополненная одним изолированным состоянием, имеет вид

$$R = \begin{array}{c|cccccccc} q_1 & 0 & x_1/y_9 & x_2/y_9 & 0 & 0 & 0 & 0 & 0 \\ q_2 & 0 & 0 & 0 & x_1/y_9 & x_2/y_9 & 0 & 0 & 0 \\ q_3 & 0 & 0 & 0 & 0 & 0 & x_1/y_9 & x_2/y_9 & 0 \\ q_4 & x_1/y_1 \vee x_2/y_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_5 & x_1/y_3 \vee x_2/y_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_6 & x_1/y_5 \vee x_2/y_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_7 & x_1/y_7 \vee x_2/y_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}.$$

Если привести ее к виду правильной клеточной матрицы, то получим 27 запрещенных переходов. Находим пару (π, ρ) разбиений, где

$$\pi = \{\pi_1, \pi_2\}, \qquad \rho = \{\rho_1, \rho_2, \rho_3, \rho_4\},$$

$$\pi_1 = \{q_1, q_3, q_4, q_5\}, \quad \pi_2 = \{q_2, q_6, q_7, q_8\},$$

$$\rho_1 = \{q_4, q_6\}, \quad \rho_2 = \{q_2, q_3\}, \quad \rho_3 = \{q_5, q_7\}, \quad \rho_4 = \{q_1, q_8\},$$

которая определяет подстановки t и t^{-1} множества состояний автомата

$$t = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \\ q_4 & q_3 & q_5 & q_1 & q_6 & q_2 & q_7 & q_8 \end{pmatrix},$$

$$t^{-1} = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \\ q_4 & q_6 & q_2 & q_1 & q_3 & q_5 & q_7 & q_8 \end{pmatrix}.$$

Применяя подстановку t^{-1} к матрице R и преобразуя ее к виду правильной клеточной матрицы, получим

$$R' = \begin{array}{c|cccc:cccc|}
 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \\
\hline
q_1 & 0 & 0 & 0 & \begin{matrix} x_1/y_1 \vee x_2/y_2 \vee \\ \vee \boxed{x_1/y_5} \vee \boxed{x_2/y_6} \end{matrix} & 0 & 0 & 0 & \begin{matrix} \boxed{x_1/y_1} \vee \boxed{x_2/y_2} \vee \\ \vee \boxed{x_1/y_5} \vee \boxed{x_2/y_6} \end{matrix} \\
q_2 & \boxed{x_1/y_9} & 0 & \boxed{x_2/y_9} & 0 & x_1/y_9 & 0 & x_2/y_9 & 0 \\
q_3 & 0 & 0 & 0 & \begin{matrix} x_1/y_3 \vee x_2/y_4 \vee \\ \vee \boxed{x_1/y_7} \vee \boxed{x_2/y_8} \end{matrix} & 0 & 0 & 0 & \begin{matrix} \boxed{x_1/y_3} \vee \boxed{x_2/y_4} \vee \\ \vee \boxed{x_1/y_7} \vee \boxed{x_2/y_8} \end{matrix} \\
q_4 & 0 & \begin{matrix} x_2/y_9 \vee \\ \vee \boxed{x_1/y_9} \end{matrix} & 0 & 0 & 0 & \begin{matrix} x_1/y_9 \vee \\ \vee \boxed{x_2/y_9} \end{matrix} & 0 & 0 \\
\hdashline
q_5 & 0 & 0 & 0 & \begin{matrix} x_1/y_5 \vee x_2/y_6 \vee \\ \vee \boxed{x_1/y_1} \vee \boxed{x_2/y_2} \end{matrix} & 0 & 0 & 0 & 0 \\
q_6 & x_1/y_9 & 0 & x_2/y_9 & 0 & 0 & 0 & 0 & 0 \\
q_7 & 0 & 0 & 0 & \begin{matrix} x_1/y_7 \vee x_2/y_8 \vee \\ \vee \boxed{x_1/y_3} \vee \boxed{x_2/y_4} \end{matrix} & 0 & 0 & 0 & 0 \\
q_8 & 0 & \begin{matrix} \boxed{x_1/y_9} \vee \\ \vee \boxed{x_2/y_9} \end{matrix} & 0 & 0 & 0 & 0 & 0 & 0
\end{array}.$$

Разложим матрицу R' на две матрицы

$$R_1 = \begin{array}{c}v_1 \\ v_2\end{array}\left\| \begin{array}{cc} \underline{x}_1 \vee \underline{x}_2 & \underset{\sim}{x}_1 \vee \underset{\sim}{x}_2 \\ \underset{\circ}{x}_1 \vee \underset{\circ}{x}_2 & 0 \end{array} \right\|,$$

$$R_2' = \begin{array}{c|cccc|}
 & w_1 & w_2 & w_3 & w_4 \\
\hline
w_1 & 0 & 0 & 0 & \underset{\sim}{x}_1/\underset{\circ}{y}_1 \vee \underline{x}_1/\underline{y}_5 \vee \underset{\sim}{x}_2/\underset{\circ}{y}_2 \vee \underline{x}_2/\underline{y}_6 \\
w_2 & x_1/\underline{y}_9 & 0 & x_2/\underline{y}_9 & 0 \\
w_3 & 0 & 0 & 0 & \underset{\sim}{x}_1/\underset{\circ}{y}_3 \vee \underline{x}_1/\underline{y}_7 \vee \underset{\sim}{x}_2/\underset{\circ}{y}_4 \vee \underline{x}_2/\underline{y}_8 \\
w_4 & 0 & \underline{x}_1/\underset{\circ}{y}_9 \vee \underset{\sim}{x}_2/\underset{\circ}{y}_9 & 0 & 0
\end{array},$$

которые после введения связей принимают вид

$$R_1 = \begin{matrix} v_1 \\ v_2 \end{matrix}\left\| \begin{matrix} (w_1 \vee w_3)(x_1 \vee x_2) \vee w_4 x_2 & (w_2 \vee w_4) x_1 \vee w_2 x_2 \\ (w_1 \vee w_2 \vee w_3)(x_1 \vee x_2) & 0 \end{matrix} \right\|,$$

$$R_2' = \begin{matrix} & \overbrace{\quad}^{w_1} & \overbrace{\quad}^{w_2} & \overbrace{\quad}^{w_3} & \overbrace{\quad}^{w_4} \\ w_1 & 0 & 0 & 0 & \begin{matrix} v_1x_1/y_1 \vee v_2x_1/y_5 \vee \\ \vee v_1x_2/y_2 \vee v_2x_2/y_6 \end{matrix} \\ w_2 & x_1/y_9 & 0 & x_2/y_9 & 0 \\ w_3 & 0 & 0 & 0 & \begin{matrix} v_1x_1/y_3 \vee v_2x_1/y_7 \vee \\ \vee v_1x_2/y_4 \vee v_2x_2/y_8 \end{matrix} \\ w_4 & 0 & v_1x_1/y_9 \vee v_1x_2/y_9 & 0 & 0 \end{matrix}.$$

Применяя к матрице R'_2 подстановку

$$t_1^{-1} = t_1 = \begin{pmatrix} w_1 & w_2 & w_3 & w_4 \\ w_4 & w_2 & w_3 & w_1 \end{pmatrix},$$

определяемую парой (π', ρ') разбиений, где

$$\pi' = \{\pi'_1, \pi'_2\}, \quad \rho = \{\rho'_1, \rho'_2\}, \quad \pi'_1 = \{w_2, w_4\}, \quad \pi'_2 = \{w_1, w_3\},$$

$$\rho'_1 = \{w_3, w_4\}, \quad \rho'_2 = \{w_1, w_2\},$$

и приводя ее к ПКМС с запретами, получим

$R''_2 =$

	w_1	w_2	w_3	w_4
w_1	$\overline{v_1x_1/y_3} \vee \overline{v_1x_2/y_4}$	$v_1x_1/y_9 \vee v_1x_2/y_9$	$\overline{v_1x_1/y_3} \vee \overline{v_2x_1/y_7} \vee \overline{v_1x_2/y_4} \vee \overline{v_2x_2/y_8}$	$\overline{v_1x_1/y_9} \vee \overline{v_1x_2/y_9}$
w_2	$\overline{v_1x_1/y_1} \vee \overline{v_1x_2/y_2} \vee \overline{v_1x_2/y_6}$	$\overline{v_1x_1/y_9}$	$\overline{v_1x_1/y_1} \vee \overline{v_2x_1/y_5} \vee \overline{v_1x_2/y_9} \vee \overline{v_1x_2/y_9} \vee \overline{v_2x_2/y_6} \vee v_2x_2/y_9$	$v_1x_1/y_9 \vee v_2x_1/y_9$
w_3	$v_1x_1/y_3 \vee v_2x_1/y_7 \vee v_1x_2/y_4 \vee v_2x_2/y$	$\overline{v_1x_1/y_9} \vee \overline{v_1x_2/y_9}$	0	0
w_4	$v_1x_1/y_1 \vee v_2x_1/y \vee v_1x_2/y_2 \vee \overline{v_1x_2/y_9} \vee v_2x_2/y_6 \vee \overline{v_2x_2/y_9}$	$\overline{v_1x_1/y_9} \vee \overline{v_2x_1/y_9}$	0	0

После разложения матрицы R''_2 на матрицы R_2 и R_3 и введения в них связей будем иметь

$$R_2 = \begin{matrix} h_1 \\ h_2 \end{matrix} \left\| \begin{matrix} s_1 v_1 (x_1 \vee x_2) & s_2 (x_1 \vee x_2) \\ x_1 \vee x_2 & 0 \end{matrix} \right\|,$$

$$R_3 = \begin{matrix} \\ s_1 \\ s_2 \end{matrix} \left\{ \left\| \begin{matrix} h_2 v_1 x_1/y_3 \vee h_2 v_2 x_1/y_1 \vee \\ \vee h_2 v_1 x_2/y_4 \vee h_2 v_2 x_2/y_8 & h_1 v_1 x_1/y_9 \vee h_1 v_1 x_2/y_9 \\ h_2 v_1 x_1/y_1 \vee h_2 v_2 x_1/y_5 \vee \\ \vee h_2 v_1 x_2/y_2 \vee h_1 v_1 x_2/y_9 \vee & h_1 v_1 x_1/y_9 \vee h_1 v_2 x_1/y_9 \\ \vee h_2 v_2 x_2/y_6 \vee h_1 v_2 x_2/y_9 \end{matrix} \right\} \right.$$

Закодируем входные сигналы и состояния элементарных абстрактных автоматов следующим образом:

$$x_1 \to m, \quad v_1 \to U_1, \quad h_1 \to U_2, \quad s_1 \to U_3,$$

$$x_2 \to \bar{m}, \quad v_2 \to \bar{U}_1, \quad h_2 \to \bar{U}_2, \quad s_2 \to \bar{U}_3.$$

Выберем в качестве элементарных автоматов триггеры с раздельными входами. Тогда по матрицам соединений элементарных абстрактных автоматов после минимизации получим функции возбуждения триггеров

$$u_{11} = \bar{U}_1,$$

$$u_{01} = U_1 U_2 \bar{U}_3 \vee m U_1 U_2,$$

$$u_{12} = \bar{U}_2,$$

$$u_{02} = U_2 \bar{U}_3,$$

$$u_{13} = \bar{U}_2 \vee m \bar{U}_3,$$

$$u_{03} = U_2 U_3.$$

Найдем функции выходов автомата. По матрице соединений R_3 с учетом перекодировки можно записать

$$y_1 = m U_1 \bar{U}_2 \bar{U}_3, \qquad y_6 = \bar{m} \bar{U}_1 \bar{U}_2 \bar{U}_3,$$

$$y_2 = \bar{m} U_1 \bar{U}_2 \bar{U}_3, \qquad y_7 = m \bar{U}_1 \bar{U}_2 U_3,$$

$$y_3 = m U_1 \bar{U}_2 U_3, \qquad y_8 = \bar{m} \bar{U}_1 \bar{U}_2 U_3,$$

$$y_4 = \overline{m} U_1 \bar{U}_2 U_3, \qquad y_9 = m U_2 U_3 \vee \bar{m} U_2 U_3 \vee \bar{m} U_2 \bar{U}_3 \vee m U_2 \bar{U}_3.$$

$$y_5 = m \bar{U}_1 \bar{U}_2 \bar{U}_3,$$

Закодируем выходные сигналы четырьмя переменными l_1, l_2, l_3, l_4 следующим образом:

$$y_1 = l_1 l_2 l_3 l_4, \qquad y_6 = l_1 l_2 l_3 l_4,$$
$$y_2 = l_1 l_2 l_3 l_4, \qquad y_7 = l_1 l_2 l_3 l_4,$$
$$y_3 = l_1 l_2 \bar{l}_3 l_4, \qquad y_8 = l_1 \bar{l}_2 \bar{l}_3 l_4,$$
$$y_4 = l_1 l_2 l_3 l_4, \qquad y_9 = l_1 l_2 l_3 l_4.$$
$$y_5 = l_1 \bar{l}_2 l_3 l_4,$$

После минимизации функции выходов будут иметь вид

$$l_1 = \bar{U}_2,$$
$$l_2 = U_1 \vee U_2,$$
$$l_3 = U_2 \vee \bar{U}_3,$$
$$l_4 = U_2 \vee m.$$

По найденным функциям возбуждения триггеров и функциям выходов легко построить структурную схему дешифратора, которая показана на рис. 4.5.

-

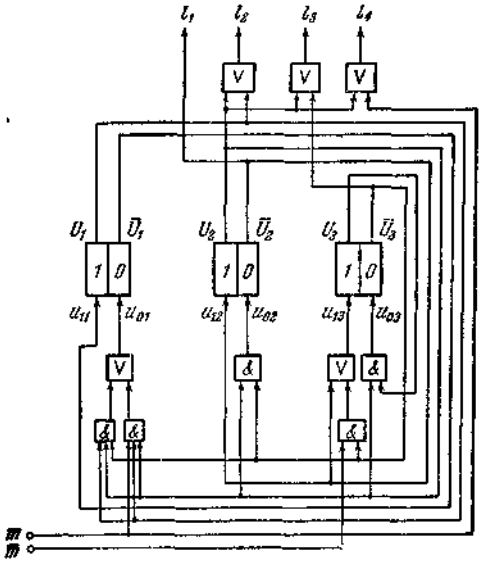


Рис. 4, 5

4. 5. О синтезе автоматов в универсальных вычислительных средах

Успехи микроэлектроники привели к необходимости разработки вопросов синтеза цифровых автоматов на новой основе. Примером этому служит задача представления исходного автомата совместной работой заданных стандартных автоматов в виде больших интегральных схем на МОП-гранзисторах, которая рассмотрена в предыдущей главе. Перспективным направлением по созданию надежных, быстродействующих автоматов в микроэлектронном исполнении являются однородные универсальные вычислительные среды, использование которых позволяет применять методы автоматического синтеза цифровых автоматов с программно изменяемой структурой. Задача синтеза автоматов в вычислительной среде (ВС) состоит из двух задач: общей задачи синтеза автоматов в ВС и задачи реализации логических сетей автоматов в ВС. **Эти задачи заключаются в получении программы настройки элементов вычислительной среды либо по абстрактному автомату, заданному графом или матрицей соединений, либо по структурной схеме (логической сети) автомата, причем задача реализации логической сети в ВС является частью общей задачи синтеза автоматов в ВС. Оптимизация синтеза автоматов в ВС, по существу, сводится к минимизации числа элементов вычислительной среды, настраиваемых на схему синтезируемого автомата.** Задача оптимизации синтеза автомата в ВС состоит из двух задач: оптимизации синтеза структурной схемы автомата (**логический аспект оптимизации**) и оптимизации размещения логической сети автомата в среде (**топологический аспект оптимизации**). Задача оптимизации структурной схемы заключается в минимизации числа функциональных элементов, образующих схему автомата, а задача оптимизации топологического размещения логической сети в среде сводится к минимизации количества связей и минимизации суммарной длины связей в схеме автомата. Оптимальная декомпозиция автоматов решает задачу минимизации числа функциональных элементов структурной схемы и задачу минимизации количества связей в схеме автомата.

Что касается задачи минимизации суммарной длины связей в схеме автомата, то этот вопрос обычно рассматривается не в логическом, а в топологическом проектировании схем цифровых автоматов.

Известно несколько подходов к проблеме синтеза автоматов в ВС. В ряде работ предлагаются методы, основанные на таком преобразо-

вании графа логической сети, в результате которого получается программа настройки ВС. Заметим, что решению важных задач оптимального топологического размещения логических сетей в ВС посвящено много работ. Известен подход, основанный на реализации автомата в ВС по временным логическим функциям. В ряде работ предлагаются методы реализации автоматов в криотронной ВС по каноническим уравнениям и по графу автомата без составления и преобразования логической сети. Достоинство этих методов — в простоте получения программы настройки криотронной ВС.

В этом параграфе излагается канонический метод синтеза произвольных автоматов в универсальной вычислительной среде, использующий идеи, предложенные для криотронной среды. В качестве функционального элемента среды может использоваться универсальный логический элемент, например, стрелка Пирса или штрих Шеффера, а в качестве соединительного— элемент, реализующий функции *P, D* и 0.

Условное обозначение соединительного элемента среды показано на рис. 4.6.

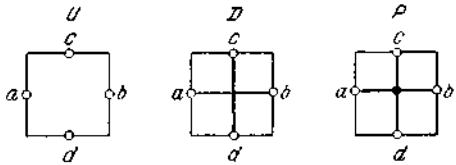


Рис. 4.6.

Функции *P, D* и 0, на одну из которых настраивается соединительный элемент, определяются матрицами

$$P = \begin{array}{c|cccc} & a & b & c & d \\ a & 0 & 1 & 1 & 1 \\ b & 1 & 0 & 1 & 1 \\ c & 1 & 1 & 0 & 1 \\ d & 1 & 1 & 1 & 0 \end{array}, \quad D = \begin{array}{c|cccc} & a & b & c & d \\ a & 0 & 1 & 0 & 0 \\ b & 1 & 0 & 0 & 0 \\ c & 0 & 0 & 0 & 1 \\ d & 0 & 0 & 1 & 0 \end{array}$$

и 0 — нулевой матрицей того же порядка.

Задачу синтеза конечного автомата в вычислительной среде сформулируем следующим образом. Пусть элемент ВС обладает автоматной и соединительной полнотой, т. е. элемент среды реализует логическую функцию ИЛИ — НЕ (стрелка Пирса), задержку, а также соединительные функции *P, D,* 0, и имеется автомат, заданный либо системой функций возбуждения элементарных автоматов и

функций выходов, либо матрицей соединений. Требуется получить программы настройки ВС для указанных способов задания автомата. Рассмотрим вначале реализацию элементарных логических функций в ВС. Как известно, инверсию переменной, а также конъюнкцию и дизъюнкцию двух переменных в среде можно представить так, как показано на рис. 4.7, а, б,в, где *F*— элемент среды, выполняющий логическую операцию ИЛИ —НЕ.

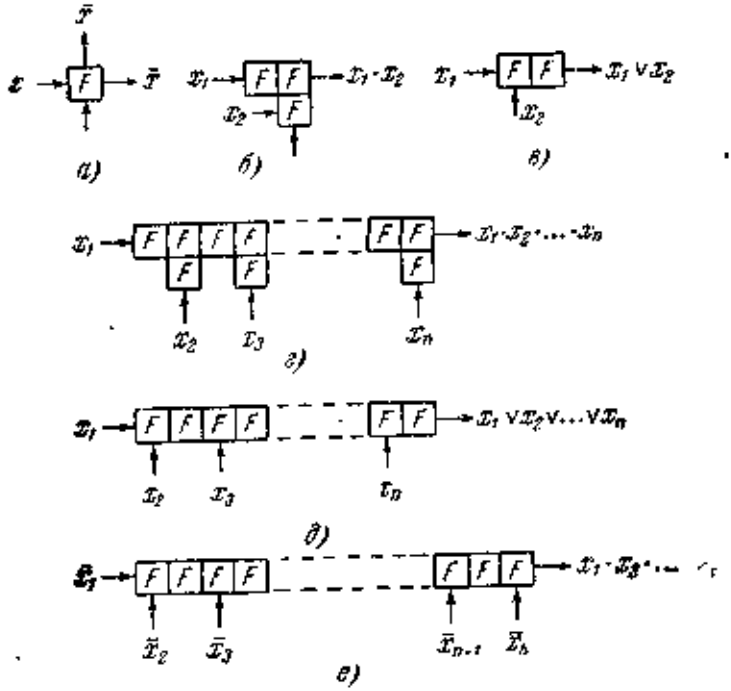


Рис. 4.7.

Реализация в среде конъюнкции и дизъюнкции *n* переменных показана соответственно на рис. 4.7, *г, д.* В том случае, когда переменные подаются своими инверсными значениями, конъюнкцию *n* переменных можно реализовать в среде в виде одной строки, как и дизъюнкцию *n* переменных (рис. 4.7, *е).*

Перейдем теперь к реализации в среде булевой функции. Будем считать, что она задана в дизъюнктивной нормальной форме (ДНФ), которую в общем виде можно записать следующим образом:

$$f(x_1, x_2, \ldots, x_n) = \bigvee_{i=1}^{r} \tilde{x}_{i_1} \cdot \tilde{x}_{i_2} \cdot \ldots \cdot \tilde{x}_{i_{k_i}},$$

где r — число элементарных конъюнкций (членов) ДНФ, $k_i \in K = \{1, 2, \ldots, n\}$, а $\tilde{x}_i$ —прямое или инверсное значение переменной x_i.

Вся программа реализации ДНФ функции $f(x_1,x_2, ...,x_n)$ в среде состоит из программы $П_{\&}$ реализации элементарных конъюнкций и программы $П_{\vee}$ реализации дизъюнкции элементарных конъюнкций.

Программа $П_{\&}$ получается из матрицы $M_{\&}$, которая строится по следующему алгоритму.

1°. *Нумеруем все члены ДНФ по порядку* 1,2, ..., r. *Переходим к* 2°.

2°. *Подсчитываем число* m *различных переменных в ДНФ. Если одна и та же переменная входит в ДНФ как в прямом, так и в инверсном виде, считаем, что это две различные переменные. Переходим к* 3°.

3°. *Строим матрицу размерности* $r \times 2m$. *Каждые два столбца отмечаем одной переменной. Если переменная* x_i *входит в ДНФ прямым и инверсным значениями, то условимся инверсное значение* $\bar{x}_i$ *ставить слева рядом с ее прямым значением. Строки матрицы, начиная сверху, нумеруются по порядку. Переходим к* 4°.

4°. *На пересечении строки, соответствующей* j *члену ДНФ и двух столбцов, отмеченных переменной* $\tilde{x}_i$, *ставим две единицы, если* $\tilde{x}_i$ *входит в* j*-член ДНФ. В противном случае ставим нули. Переходим к* 5°.

5°. *В каждой строке вычеркиваем вторую слева и первую справа единицы. Затем исключаем столбцы, содержащие только вычеркнутые единицы. Конец работы алгоритма.*

Программа $П_{\&}$ получается из матрицы $M_{\&}$ следующим образом. Первые слева единицы в каждой строке заменяются на элементы *P*, остальные единицы — на элементы *F*. После этого производим коррекцию полученной программы. Если одна переменная используется в *k* строках, то увеличиваем количество столбцов этой переменной в *k* раз и оставляем в каждом столбце только один элемент *F*. Это делается для того, чтобы нижние элементы *F* не влияли на работу верхних элементов среды в одном и том же столбце. Коррекции не подлежат лишь такие ситуации, когда в одном столбце находятся несколько элементов *P* или ниже элемента *F* находится один или несколько элементов *P*. После этого заменяем нулевые элементы на элементы *D*, а все значения переменных, подаваемых на $П_{\&}$, заменяем на инверсные. В построенной программе $П_{\&}$ справа на выходе каждой строки получается соответствующий член ДНФ.

Покажем построение $П_{\&}$ на примере. Пусть дизъюнктивная нормальная форма функции имеет вид

$$f(x_1, x_2, x_3, x_4, x_5) = x_2x_5 \vee x_3x_4 \vee x_1x_2x_4 \vee x_3x_5 \vee \bar{x}_1x_3. \quad (4.12)$$

Здесь $r = 5$, $m = 6$. Строим матрицу размером 5×12 и отмечаем столбцы различными переменными. Ставим единицы и нули в соответствующих клетках матрицы.

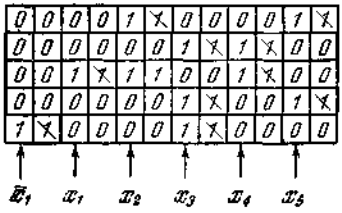


Рис. 4.8.

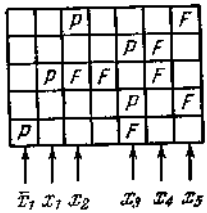


Рис. 4.9.

Получаем вначале матрицу $M_{\&}$ (рис. 4.8), затем нескорректированную программу $П_{\&}$ (рис. 4.9). После коррекции приходим к программе $П_{\&}$ настройки среды (рис. 4.10), реализующей данную функцию.

Легко видеть, что при реализации произвольного члена ДНФ $\tilde{x}_{i_1} \cdot \tilde{x}_{i_2} \cdot \ldots \cdot \tilde{x}_{i_{k_i}}$ первая по порядку переменная $\tilde{x}_{i_1}$ подается в программе $П_{\&}$ не на элемент F, а на элемент P. Назовем эту переменную образующей переменной данного члена ДНФ.

Оказывается, что сложность $П_{\&}$ (количество столбцов) в общем случае зависит от выбора образующих переменных. Если переменная $\tilde{x}_i$ является образующей переменной в n членах ДНФ, то вместо n столбцов можно использовать один столбец с n элементами P. Помимо этого, как уже отмечалось, в одном столбце могут находиться одновременно один элемент F и несколько элементов P под ним, но не наоборот. Следовательно, упорядоченность членов ДНФ влияет на сложность программы $П_{\&}$.

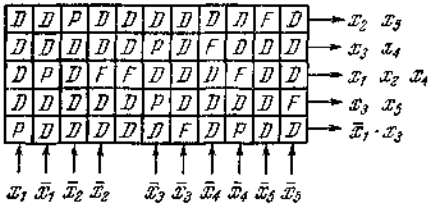


Рис. 4.10

Учитывая эти обстоятельства, можно предложить следующий алгоритм оптимизации программы $П_{\&}$ реализации логической функции.

1°. *Подсчитываем число вхождений* $\lambda^{(0)}(\tilde{x}_i)$ *каждой переменной* $\tilde{x}_i$ *во все члены ДНФ. Выбираем первую образующую переменную* $\tilde{x}_{i_0}$, *у которой наибольшая величина* $\lambda^{(0)}(\tilde{x}_i)$. *Включаем переменную* $\tilde{x}_{i_0}$ *в множество образующих переменных* M_1. *Получаем* $M_1 = \{\tilde{x}_{i_0}\}$. *Переходим к 2°.*

2°. *Исключаем из множества членов ДНФ такие, которые содержат переменную* $\tilde{x}_{i_0}$. *Получаем сокращенное множество* T_1 *членов ДНФ. Если множество* $T_1 \neq \varnothing$, *то переходим к 4°, в противном случае — к* 9°.

3°. *Исключаем из множества* T_{k-1}, *где* $k \in K = \{1, 2, \ldots, p\}$ *— номер цикла работы алгоритма, такие члены ДНФ, которые содержат переменную* $\tilde{x}_{i_{k-1}}$. *Получаем множество* T_k. *Если* $T_k \neq \varnothing$, *то переходим к* 4°, *в противном случае — к* 8°.

4°. *Из множества* N_k *всех переменных, входящих в члены ДНФ множества* T_k, *выбираем подмножество* R_k *таких переменных, которые входят совместно с одной из переменных множества* M_k, *хотя бы в один из исключенных ранее членов ДНФ. Если множество* $R_k \neq \emptyset$, *то переходим к* 5°, *в противном случае — к* 6°.

5°. *Подсчитываем для каждой переменной* $\tilde{x}_i$ *из множества* R_k *число вхождений* $\lambda^{(k)}(\tilde{x}_i)$ *в множество* T_k. *Выбираем* $\tilde{x}_{ik}$ *с наибольшей величиной* $\lambda^{(k)}(\tilde{x}_i)$. *Переходим к 7°.*

6°. *Подсчитываем для каждой переменной* $\tilde{x}_i$ *из множества* N_k *число вхождений* $\lambda^{(k)}(\tilde{x}_i)$ *в множество* T_k. *Выбираем* $\tilde{x}_{ik}$ *с наибольшей величиной* $\lambda^{(k)}(\tilde{x}_i)$. *Переходим к* 8°.

7°. *Получаем множество* M_{k+1}, *включая переменную* $\tilde{x}_{ik}$ *в множество* M_k. *Члену ДНФ, называемому отмеченным, в который образующая переменная* $\tilde{x}_{ik}$ *входит совместно с одной переменной множества* M_k, *присваиваем порядковый номер* $(q + 1)$, *где* q*— количество отмеченных ранее членов ДНФ*). *Переходим к* 3°.

(На $k + 1$ цикле работы алгоритма в пункте 4° при выборе подмножества R_{k+1} отмеченные члены ДНФ не учитываются.)

8°. *Получаем множество* M_{k+1}, *включая переменную* $\tilde{x}_{ik}$ *в множество* M_k. *Переходим к* 3°.

9°. *Конец работы алгоритма.*

Отметим, что если в ходе выполнения алгоритма окажется, что несколько переменных имеют одинаковое число вхождений $\lambda^{(k)}$, то необходимо выполнить перебор этих вариантов.

В результате алгоритма получаем упорядоченное множество образующих переменных $M_p = \{\tilde{x}_{i_0}, \tilde{x}_{i_1}, \ldots, \tilde{x}_{i_p}\}$ и упорядоченное множество отмеченных членов ДНФ. Образующие переменные из множества M_p располагаются по порядку в матрице $M_{\&}$, а отмеченные члены ДНФ реализуются в соответствующих строках, начиная сверху. Неотмеченные члены ДНФ и необразующие переменные располагаются в матрице $M_{\&}$ произвольным образом. Далее $M_{\&}$ и $П_{\&}$ строятся так, как описано выше.

Покажем работу алгоритма оптимизации на примере функции, программа которой дана на рис. 4 10.

Находим вначале $M_1 = x_3$. Тогда $T_1 = \{x_2x_5,\ x_1x_2x_4\}$, $N_1 = \{x_1, x_2, x_4, x_5\}, R_1 = \{x_4, x_5\}$. Так как $\lambda^{(1)}(x_4) = \lambda^{(1)}(x_5) = 1$, то получаем подмножества $M'_2 = \{x_3, x_4\}$ и $M''_2 = \{x_3, x_5\}$. Далее, во втором цикле алгоритма для M'_2 находим $T'_2 = \{x_2x_5\}$, $N'_2 = \{x_2, x_5\}$, $R'_2 = \{x_5\}$ и, следовательно, $M'_3 = \{x_3, x_4, x_5\}$. В первом цикле алгоритма отмечаем член x_3x_4 номером 1, а во втором цикле член x_3x_5 — номером 2. Поскольку $T'_3 = \emptyset$, строим оптимизированную программу $П'_{\&}$, которая показана на рис. 4.11.

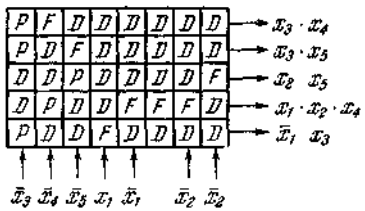


Рис. 4.11.

В случае выбора M''_2 получается программа $П''_{\&}$ такой же сложности. Отметим, что, несмотря на оптимизацию, такая реализация программы $П_{\&}$ требует большого числа элементов, особенно тогда, когда в каждом члене ДНФ содержится более двух переменных. Причиной этого служит то обстоятельство, что в общем случае для каждого нахождения переменной требуется два столбца в программе $П_{\&}$. Это является существенным недостатком такой программы.

Перейдем теперь к получению программы $П_{\vee}$, которая в общих чертах строится следующим образом. Множество r всех строк программы $П_{\&}$ разбивается на пары, начиная сверху. Для каждой пары рядом с первой сверху переменной ставятся в одной строке два элемента F, а во второй строке — элементы Р и 0 соответственно. Для исключения влияния

элементов различных пар друг на друга элементы $i + 1$ пары сдвигаются на один столбец вправо по отношению к элементам i пары. Получаем, таким образом, $\left]\frac{r}{2}\right[$ переменных, для которых разбиение на пары и расстановка элементов *F, P* и 0 повторяется вновь. Этот процесс продолжается до тех пор, пока на выходе не получится одна переменная. В последнем столбце под элементом *F* ставятся нули, а в незаполненных клетках программы — элементы *D*. Отметим два обстоятельства.

1) Элементы *F , F , P* и 0 очередной пары можно сдвигать влево на один столбец, если слева от первого элемента *F* стоит пустая клетка; элемент *P* в этом случае не ставится.

2) При нечетном числе переменных на *i* шаге образования пар переменных последняя нечетная переменная не используется; она используется как переменная только на следующем $(i + 1)$ шаге.

На этом построение программы $П_{\vee}$ заканчивается. На рис. 4.12 показаны программы $П_{\&}$ и $П_{\vee}$ реализации функции (4.12) в ВС.

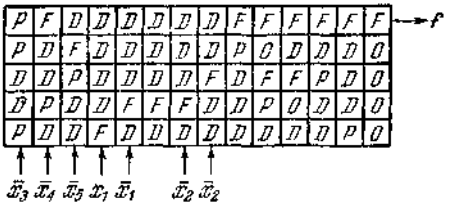


Рис. 4.12

Рассмотрим реализацию конечных автоматов в ВС. Будем считать, что автомат *A* с множеством входных двоичных переменных $M = \{m_1, m_2, \ldots, m_k\}$, множеством выходных двоичных переменных $L = \{l_1, l_2, \ldots, l_p\}$ и множеством внутренних двоичных переменных $U = \{U_1, U_2, \ldots, U_s\}$, наборами которых кодируется соответственно входные и выходные сигналы и состояния автомата, задан совокупностью минимизированных функций возбуждения элементарных автоматов

$$\left.\begin{aligned} u_{1i}(t) &= f_{1i}(\tilde{U}_1, \tilde{U}_2, \ldots, \tilde{U}_s, \tilde{m}_1, \tilde{m}_2, \ldots, \tilde{m}_k), \\ u_{2i}(t) &= f_{2i}(\tilde{U}_1, \tilde{U}_2, \ldots, \tilde{U}_s, \tilde{m}_1, \tilde{m}_2, \ldots, \tilde{m}_k), \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ u_{si}(t) &= f_{si}(\tilde{U}_1, \tilde{U}_2, \ldots, \tilde{U}_s, \tilde{m}_1, \tilde{m}_2, \ldots, \tilde{m}_k) \end{aligned}\right\} \quad (4.13)$$

по *i* входу и минимизированных функций выходов

$$\left.\begin{aligned} l_1(t) &= f_1(\tilde{U}_1, \tilde{U}_2, \ldots, \tilde{U}_s, \tilde{m}_1, \tilde{m}_2, \ldots, \tilde{m}_k),\\ l_2(t) &= f_2(\tilde{U}_1, \tilde{U}_2, \ldots, \tilde{U}_s, \tilde{m}_1, \tilde{m}_2, \ldots, \tilde{m}_k),\\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ l_p(t) &= f_p(\tilde{U}_1, \tilde{U}_2, \ldots, \tilde{U}_s, \tilde{m}_1, \tilde{m}_2, \ldots, \tilde{m}_k). \end{aligned}\right\} \quad (4.14)$$

Основная идея реализации автомата в ВС заключается в том, что массив среды настраивается на реализацию комбинационной части автомата, элементарных автоматов и требуемых соединений между ними.

Общая блок-программа реализации конечного автомата в ВС показана на рис. 4.13.

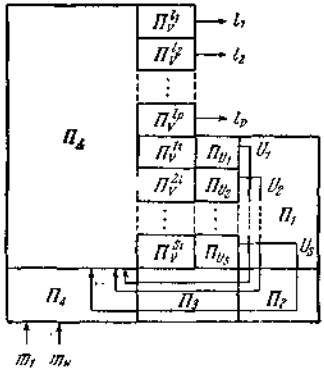


Рис. 4.13.

Программа $П_{\&}$ служит для получения членов ДНФ в выражениях (4.13) и (4.14). На программах $П_{\vee}^{ij}$, $j \in J = \{1, 2, \ldots, s\}$, $i \in I$, получается функция возбуждения по i входу j элементарного автомата. Функции выходов $l_1(t)$, $l_2(t), \ldots, l_p(t)$ реализуются соответственно на программах $П_{\vee}^{l_1}$, $П_{\vee}^{l_2}$, $\ldots$, $П_{\vee}^{l_p}$. Для того чтобы соседние программы $П_{\vee}$ для функций выходов и функций возбуждения элементарных автоматов не влияли друг на друга, между ними должны быть нулевые разделительные строки. В программе $П_{U_j}$, $j \in J$, реализуется элементарный автомат U_j. Программы $П_1$, $П_2$ и $П_3$ служат для подачи внутренних, а $П_4$ — входных и внутренних переменных на программу $П_{\&}$.

Блок-программа реализации автомата в ВС в конкретном случае может оказаться намного экономичнее, если строить программы $П_4$ и $П_{\vee}$ для групп элементарных автоматов или для каждого элементарного автомата в отдельности. Очевидно, что в этом случае

программы $П_1$ — $П_4$ будут строиться соответственно для каждой группы элементарных автома тов или для каждого элементарного автомата. В конкретных случаях блок-программу можно также частично видоизменить с целью уменьшения площади, занимаемой этой программой.

Элементарный автомат, в качестве которого будем использовать триггер с раздельными входами, можно выполнить из функциональных и соединительных элементов ВС так, как это показано на рис. 4.14.

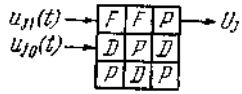


Рис. 4.14.

Если необходимо применить другой тип элементарных автоматов, то очевидно, что описываемый метод реализации автоматов в ВС можно использовать и в этом случае.

Программы $П_1$, $П_2$, $П_3$ и $П_4$ получаются из матриц M_1, M_2, M_3 и M_4. В матрице M_1 число строк равно числу строк, занимаемых программами $П_\vee$, а число столбцов равно *s*. На пересечении *j* столбца и строки, на которую подается выход элементарного автомата U_j, ставится единица, а остальные элементы матрицы M_1 равны нулю. Матрица M_2 является единичной матрицей порядка *s*, а M_3 — нулевой прямоугольной матрицей, имеющей s строк и число столбцов, равное наибольшей сумме числа столбцов в программах $П_\vee^{ii}$ и $П_{U_j}$, $j \in J$. Матрица M_4 содержит s строк и столько столбцов, сколько их имеется в программе $П_\&$. В этой матрице ставится единица на пересечении *j* строки и *i* столбца, если на *i* столбец программы $П_\&$ подается переменная $\tilde{U}_j$. Остальные элементы матрицы M_4 равны нулю.

Для того чтобы получить программы $П_1$, $П_2$, $П_3$ и $П_4$, необходимо в соответствующих матрицах заменить все единицы на элементы *P*, а нули — на элементы *D*.

Между программами $П_4$ и $П_\&$ необходимо иметь строку элементов ВС, которую назовем строкой сопряжения. Каждый элемент этой строки настраивается на выполнение либо функции *D*, либо функции *F*, либо функции *P* в зависимости от того, подается ли на программу $П_\&$ соответственно прямое значение переменной x_i, инверсное значение $\bar{x}_i$ или переменная не подается. Для того чтобы в строке сопряжения исключить взаимное влияние соседних элементов *F*, на которые подаются различные переменные, необходимо либо реализовать инверсию одной из переменных на другом участке ВС, либо поставить

между ними элемент 0, а следовательно, в программе $П_{\&}$ ввести столбец из элементов *D*.

Как уже отмечалось ранее, основной причиной, ведущей к усложнению программы $П_{\&}$, является то, что в одном столбце нельзя поместить два и более элементов *F*. Программу реализации автомата в ВС можно построить с меньшей затратой элементов, если несколько видоизменить элемент среды *F* следующим образом: на вертикальном выходе, который существенно не используется в программе $П_{\&}$, будем получать не функцию, а переменную, которая подается на вертикальный вход элемента *F*. На горизонтальном выходе элемента *F*, как и ранее, будет получаться функция двух переменных. В программе $П_{\&}$, построенной из таких модифицированных элементов, количество столбцов не зависит от числа вхождений переменных, а зависит только от числа различных переменных.

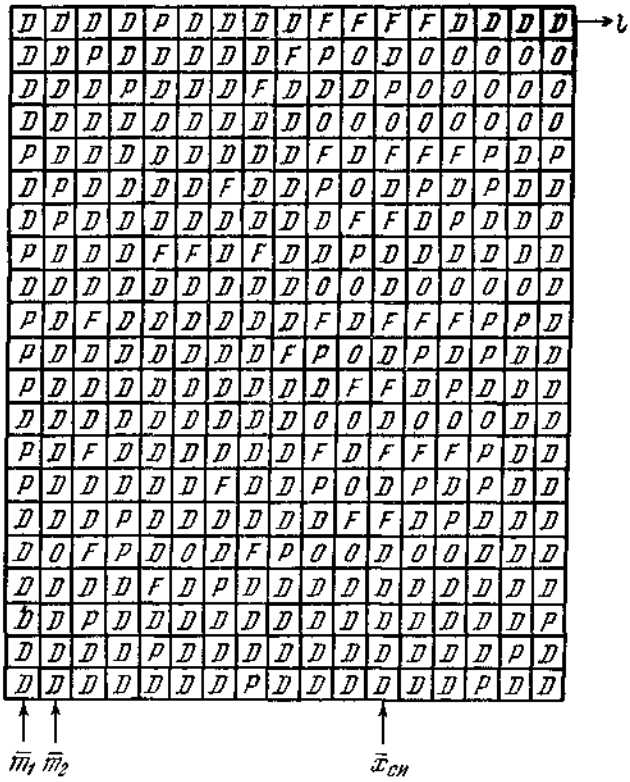


Рис. 4.15

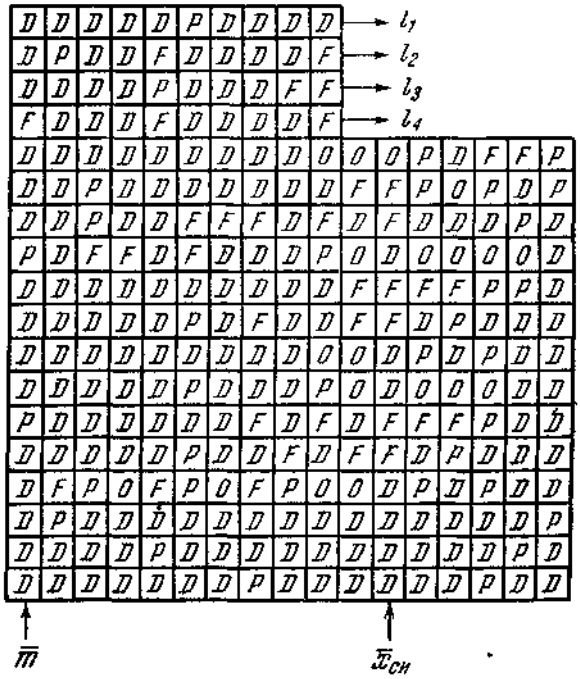


Рис. 4.16

Блок-программа реализации автомата в модифицированной ВС остается прежней. Отметим только, что при получении программы $П_{\&}$ из матрицы $M_{\&}$ отпадает необходимость в ее коррекции и несущественно видоизменяется программа $П_{\vee}$, так как в модифицированном элементе *F* нельзя получить функцию с вертикального выхода.

Приведенный метод реализации автоматов в ВС подразумевал, строго говоря, реализацию в среде асинхронных автоматов. Однако этот метод можно использовать и при реализации синхронных автоматов В этом случае на выходе каждой программы $П^{ij}_{\vee}$, $j \in J$, $i \in I$, на последний элемент *F* подается инверсное значение синхронизирующего сигнала. В общем случае для этого вводится дополнительный столбец из элементов ВС, а в среде с модифицированным элементом *F* дополнительные элементы ВС не требуются.

При синтезе абстрактных автоматов в ВС исходным заданием автомата является представление его в виде графа или матрицы соединений. По

графу автомата или матрице соединений требуется получить программу настройки ВС, реализующую отображение, индуцируемое исходным автоматом. Для этого вначале по матрице соединений автомата производим оптимальную декомпозицию, в результате которой получаем матрицы соединений элементарных абстрактных автоматов. По ним записываем функции возбуждения элементарных автоматов и функции выходов, минимизируем их и описанным выше способом получаем программу настройки ВС. В качестве примера синтеза синхронных автоматов в ВС на рис. 4.15 и 4.16 соответственно показаны программы настройки ВС, в которой используется модифицированный элемент, на реализацию автомата, моделирующего выработку условного рефлекса (пример 4.2) и дешифратора последовательного действия (пример 4.3), построенные по минимизированным функциям возбуждения триггеров и функциям выходов, найденным в результате декомпозиции.

5. Синтез микропрограммного автомата

5.1. Переход от формализованного описания к структурной схеме автомата

Наиболее распространенной *структурной схемой* микропрограммного автомата (МА) является схема Уилкса — Стринжера, упрощенно представленная на рис. 5.1.

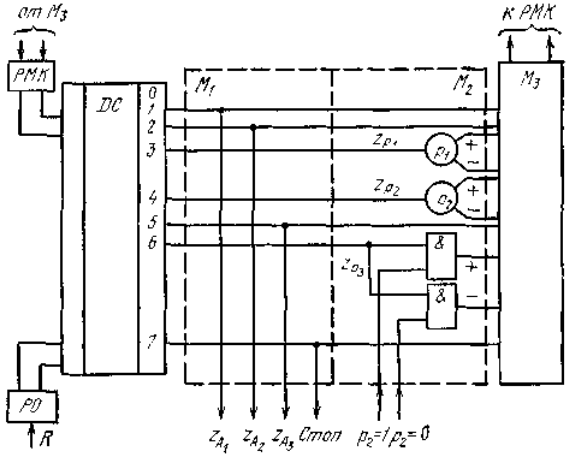


Рис. 5.1

Схема состоит из регистра микрокоманд РМК, дешифратора *DC*, матрицы внешних микроопераций M_1 матрицы внутренних микроопераций M_2, матрицы формирования кода следующей микрокоманды M_3 и регистра операций *РО*.
В случае применения микропрограммного автомата для построения ЦБУ каждая внешняя микрооперация Z_{A_i} является управляющим сигналом к $ОФБ_i$, а внутренняя Z_{p_j} — управляющим сигналом к *ЛФБ,,* проверяющему условие p_j, значениями которого являются выполнение (p_j= 1; отмечено знаком +) или невыполнение условия (p_j = 0; отмечено знаком —). На структурной схеме кружочком обозначена схема, которая служит для запроса от $ЛФБ_j$ значения проверяемого им условия p_j (на рис. 5.1 схема запроса от ЛФБ значения ЛУ показана лишь для второго логического условия p_2).
Выход дешифратора *DC* сопоставляется с микрокомандой, выполняемой за один такт работы автомата. Код каждой микрокоманды хранится в *РМК,* который может состоять из триггеров $T—T_m$. После выполнения данной микрокоманды в M_3 формируется код следующей микрокоманды, который передается в *РМК.* Последовательность микрокоманд образует *микропрограмму,* код которой хранится в *РО.* При поступлении в *РО* кода R_i автомат начинает вырабатывать в соответствии с микропрограммой, которой приписан код R_i , последовательность микрокоманд. При этом обычно предполагается, что код R_i может измениться лишь после того, как закончится выполнение соответствующей микропрограммы. Поэтому часто *РО* совмещают с *РМК* (рис. 5.2), а код R_i сопоставляется с кодом первой микрокоманды *i*-й микропрограммы.

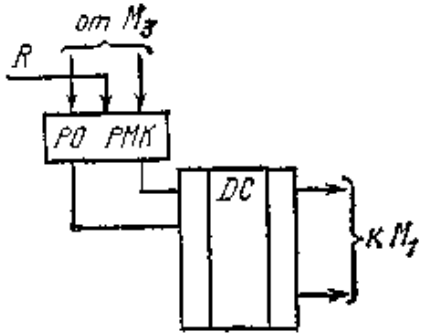


Рис. 5.2

Тогда микрокоманды всех микропрограмм образуют одну общую последовательность (рис. 5.3).

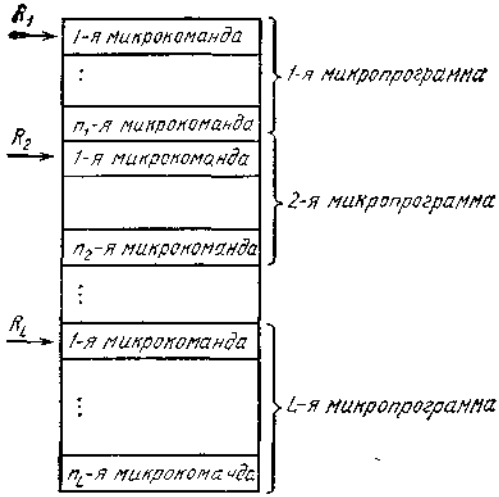


Рис. 5.3

При второй реализации микропрограммного автомата, очевидно, число разрядов *РО—РМК* определяется общим числом микрокоманд, тогда как при первой реализации — наибольшим числом микрокоманд в одной микропрограмме (разрядность *РМК)* и числом различных микропрограмм (разрядность *РО).*

В управляющем автомате вход *R* микропрограммного автомата является внешним входом, а входы в матрицу M_3 от *ЛФБ*— его внутренними входами, *РМК* образует память микропрограммного автомата, а каждый триггер *РМК* (разряд *РМК)* — его ЭП.

Если используется синхронный микропрограммный автомат, то импульсы от тактового генератора (ТГ) поступают на вход *DC.* Для того чтобы устранить состязания между ЭП синхронного микропрограммного автомата, *РМК* строится по принципу «удвоенной памяти» (рис. 5.4).

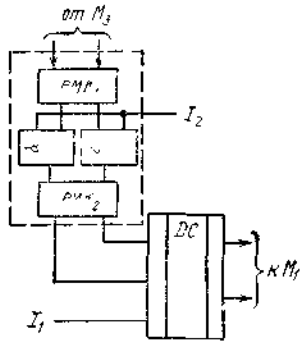


Рис. 5.4

В этом случае от ТГ поступают две последовательности импульсов: I_1 и I_2 (рис. 5.5), сдвинутые одна по отношению к другой.

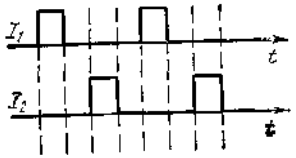


Рис. 5.5

При поступлении импульса из I_1 автомат в соответствии с кодом, хранящимся в $РМК_2$, вырабатывает сигнал на одном из выходов *DC*, который поступает на соответствующие внешние z_{A_i} и внутренние z_{p_j} выходы.

В этот же такт сигналы с выходов матрицы M_3 воздействуют на триггеры $РМК_1$, где формируется код следующей микрокоманды. При поступлении импульса из I_2 код очередной микрокоманды, которая будет выполняться при поступлении очередного импульса из I_1 передается из $РМК_1$ в $РМК_2$.

Легко видеть, что при использовании такого способа устранения состязаний между ЭП уменьшается быстродействие автомата. Для устранения этого недостатка применяют другую структурную схему МА, приведенную на рис. 5.6.

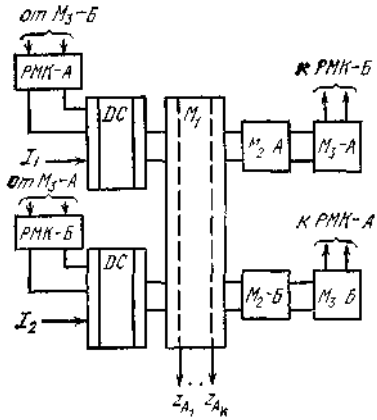


Рис. 5.6

В таком МА *РМК, DC,* матрицы M_2 и M_3 разделены на две части: *А* и *Б,* причем от M_3— *А* сигналы воздействуют на *РМК—Б,* и наоборот, от M_3—*Б* на *РМК—А.* Матрица M_1 является общей для двух частей. Во время воздействия импульса из последовательности I_1 автоматом вырабатывается выходное воздействие z_{A_i} и из матрицы $M_{3з}$—*А* передаются сигналы в *РМК— Б,* где формируется код следующей микрокоманды. Импульсы из последовательности I_2 воздействуют уже на часть *Б* автомата и при этом воздействии автомат вырабатывает выходное воздействие z_{A_j}, а код следующей микрокоманды формируется уже в *РМК—А* под действием сигналов из матрицы M_3—*Б* и т. д.

Таким образом, микропрограммный автомат данной структуры при импульсах от ТГ как из последовательности I_1, так и из последовательности I_2 вырабатывает выходное воздействие, что повышает его быстродействие. Однако в этом случае требуется все микрокоманды разделить на две группы: микрокоманды, выполняемые при воздействии импульсов от ТГ из последовательности I_1 (нечетные микрокоманды), и микрокоманды, выполняемые при воздействии импульсов от ТГ из последовательности I_2 (четные микрокоманды). При этом требуется иметь только такие микропрограммы, в которых от каждой нечетной микрокоманды был переход только к четным, а от каждой четной— только к нечетным микрокомандам, что не всегда можно сделать без введения «пустых» микрокоманд, при которых не выполняются внешние и внутренние микрооперации.

Микропрограмма, т. е. последовательность выполнения микроопераций, удобно описывается на языке ЛСА, причем с оператором *A,* сопоставляется внешняя микрооперация z_{A_i}, а с логическим условием p_j— внутренняя микрооперация z_{p_j}.

Например, условия работы МА, схема которого представлена на рис. 5.1, можно описать ЛСА

$$\mathfrak{A} = \downarrow^{2} A_1 A_2 p_1 \uparrow^{1} p_2 \uparrow^{1} A_3 p_2 \uparrow^{2} \downarrow^{1}. \quad (5.1)$$

Здесь с каждым членом ЛСА сопоставляется свой выход дешифратора. В соответствии с полученным числом выходов *DC* выбирается число разрядов *РМК.* При такой реализации в каждую микрокоманду входит только одна микрооперация (внешняя или внутренняя) и в течение каждого микротакта осуществляется включение только одного ОФБ или ЛФБ. Число внутренних состояний МА полностью определяется числом членов ЛСА. Объем матриц M_1 и M_2 зависит от числа операторов и логических условий в ЛСА. В частном случае, когда в ЛСА входят только операторы, матрица M_2 отсутствует. В матрице M_3 при этом каждый раз формируется номер следующей по порядку микрокоманды.

При наличии в ЛСА логических условий необходима матрица M_2, в которой формируются сигналы на включение ЛФБ. При ложном значении проверяемого в ЛФБ логического условия (если оно входит в ЛСА без отрицания) естественный порядок выполнения членов ЛСА нарушается. Тогда в M_3 должен быть сформирован номер той микрокоманды, который необходим для правильного выполнения ЛСА Очевидно, если с каждым членом ЛСА сопоставлять отдельную микрокоманду, для реализации даже достаточно простых алгоритмов потребуется большое число внутренних состояний МА. Однако в большинстве практических случаев нет необходимости отводить внутреннее состояние для каждого члена ЛСА. Оказывается, что **некоторые внешние микрооперации можно выполнять не последовательно, а одновременно — за один микротакт**. Это возможно в том случае, когда ОФБ, соответствующие этим микрооперациям, могут работать параллельно. **Тогда с каждым внутренним состоянием МА сопоставляется не один оператор, а группа одновременно выполняемых операторов ЛСА.** Естественно, при этом уменьшается число выходов дешифратора, что, в свою очередь, может привести к сокращению числа разрядов РМК. Кроме сокращения объема оборудования совмещение во времени отдельных микроопераций приводит к повышению быстродействия, так как

уменьшается число микротактов, необходимых для выполнения алгоритма.

Дальнейшее упрощение схемы МА может быть достигнуто за счет одновременного выполнения внешних и внутренних микроопераций. Тогда число микрокоманд МА будет определяться не числом членов ЛСА, а **числом групп одновременно выполняемых микроопераций.** Если например в ЛСА (5.1) можно выделить три такие группы:

$$\underbrace{\mathfrak{A} = \downarrow {}^{2} A_1 A_2 p_1 \uparrow {}^{1} p_2 \downarrow {}^{1}}_{1} \underbrace{A_3 p_2 \uparrow {}^{3}}_{2} \underbrace{\downarrow {}^{1}}_{3} \cdot , \quad (5.2)$$

то МА будет иметь только три внутренних состояния (рис. 5.7).

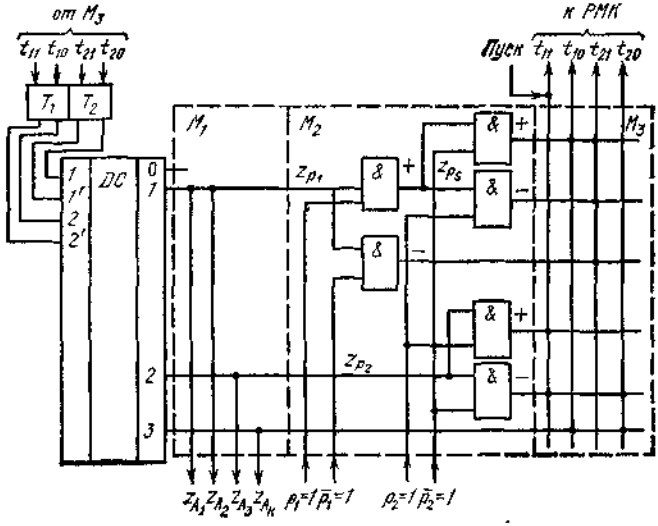


Рис. 5.7

Тогда достаточно иметь лишь двухразрядный РМК (триггеры T_1 и T_2). Матрица M_3 на рис. 5 7 построена при сопоставлении с микрокомандами 1, 2, 3 кодовых комбинаций 10, 01, 11 соответственно. Выходной сигнал t_{i1} этой матрицы переводит триггер T_i в единичное состояние, а сигнал t_{i0} — в нулевое.

Таким образом, рассмотрели способ постепенного укрупнения микрокоманд и в результате перешли от микрокоманды, в которую входила только одна внешняя или внутренняя микрооперация, к микрокоманде, в которую может входить целая группа как внешних, так и внутренних микроопераций. Для формирования таких микрокоманд кроме самой исходной ЛСА, которая должна быть

реализована микропрограммным автоматом, должна быть задана информация о возможности одновременного выполнения различных операторов и о распределении сдвигов для каждого оператора. Очевидно, что в одну микрокоманду не могут входить оператор A_i и логическое условие p_j, если значение p_j может быть изменено оператором A_i. Задача формирования наименьшего возможного числа микрокоманд ЛСА является сложной, и решение ее будет рассмотрено в следующем параграфе.

При рассмотрении различных возможных способов построения схемы МА считали, что в ЦБУ реализуется только один алгоритм. Однако программный способ управления используется именно тогда, когда в УА необходимо реализовать несколько различных алгоритмов.

Как уже отмечалось, при этом МА перестает быть автономным, так как имеются внешние входы $r_1, ..., r_n$. В зависимости от набора значений этих переменных в УА должен быть реализован один из алгоритмов $\mathfrak{A}_1, ..., \mathfrak{A}_N$. Например, в ЭВМ алгоритмы $\mathfrak{A}_1, ..., \mathfrak{A}_N$ описывают микропрограммы выполнения отдельных операций, а набор значений переменных $r_1,..., r_n$ соответствует коду операции. На наборе $r_1^{\sigma_1^i}, ..., r_n^{\sigma_n^i}$ конъюнкция R_i этих переменных принимает единичное значение, так что с каждым алгоритмом $\mathfrak{A}_i$ можно сопоставить конъюнкцию R_i переменных $r_1,..., r_n$.

Внешние сигналы $r_1,..., r_n$ могут поступать на МА с регистра операций. Очевидно, разрядность этого регистра определяется числом различных алгоритмов, реализуемых УА. Эти алгоритмы названы частными в отличие от общего алгоритма, описывающего работу ЦБУ в целом.

Наиболее распространенная схемная реализация неавтономного микропрограммного автомата приведена на рис. 5.1. Здесь кроме РМК, входящего в состав автономного МА, имеется РО, с которого на дешифратор поступают входные сигналы $r_1,..., r_n$. Как уже отмечалось, разрядность РМК определяется наибольшим числом микрокоманд, необходимых для реализации частной ЛСА. Число выходов из дешифратора равно суммарному числу микрокоманд, необ ходимых для реализации всех частных ЛСА. В наихудшем случае, когда в микрокоманду входит только один член ЛСА, число выходов из дешифратора будет равно суммарному числу членов всех частных ЛСА.

При такой реализации функции, описывающие выходные сигналы МА, и функции возбуждения элементов памяти, которые формируются в матрице M_3, зависят от всех сигналов с РО и РМК, т. е. от всех входных сигналов МА. Кроме того, так как обычно различные частные ЛСА содержат много одинаковых операторов и логических условий, в схеме

МА (рис. 5.1) им будут соответствовать различные выходы дешифратора. Поэтому для упрощения схемы МА желательно сократить число повторяющихся членов в ЛСА. Такая операция осуществляется в процессе объединения частных алгоритмов. Объединенной ЛСА будет соответствовать схемная реализация, приведенная на рис. 5.8.

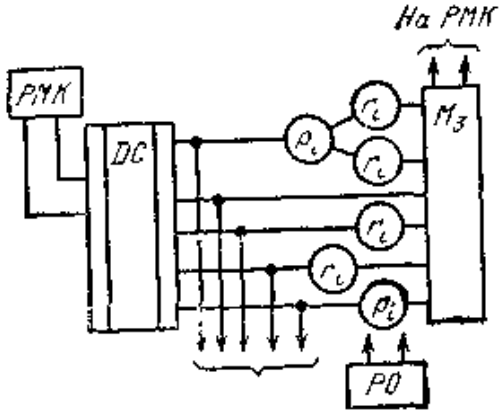


Рис. 5.8

Здесь на дешифратор воздействуют только сигналы от РМК, число выходов дешифратора равно числу различных микрокоманд, полученных по объединенной ЛСА. Однако матрица M_2 при этом усложняется, так как в ней проверяются значения ЛУ и внешних переменных, поступающих от РО.

В схеме МА, построенной по объединенной ЛСА, функции возбуждения элементов памяти зависят от всех входных сигналов автомата, а выходные функции — только от сигналов с элементов памяти. При такой реализации общий объем оборудования МА может сократиться. Однако для выбора того или иного способа реализации нужно оценить для каждого конкретного случая сложность отдельных блоков МА. Очевидно, что сложность матрицы M_3 будет зависеть от выбранного варианта кодирования внутренних состояний МА. От выбранного кода будет зависеть и устойчивость автомата по отношению к состязаниям ЭП. Таким образом, при кодировании микропрограммного автомата возникают те же задачи, что и при кодировании конечных автоматов общего типа.

Поскольку внутренние состояния МА сопоставляются с микрокомандами, очевидно, структура графа переходов МА в зна-

чителыюй степени будет зависеть от выбора микрокоманд. При слишком крупных микрокомандах граф перехода МА будет сильно связанным, что усложнит задачу устранения критических состязаний при кодировании автомата. Ситуация здесь полностью аналогична той, которая возникает при минимизации числа внутренних состояний конечного автомата: в результате этапа минимизации может быть получена такая таблица переходов, в которой нельзя устранить критические состязания, не введя дополнительных элементов памяти, т. е. те усилия, которые были затрачены на получение минимального автомата, оказываются бесполезными, так как затем потребуются не меньшие усилия, чтобы расширить таблицу переходов на этапе кодирования.

Таким образом, важной задачей является **разработка методов, позволяющих одновременно осуществлять минимизацию и кодирование автомата с учетом различных требований**. В следующем параграфе будет рассмотрен способ формирования микрокоманд, позволяющий сократить число внутренних состояний МА. Такая постановка задачи представляет интерес в том практически часто встречающемся случае, когда для устранения критических состязаний ЭП уже приняты какие-то меры, например применена двойная память.

5.2. Формирование микрокоманд по ЛСА

При использовании для построения МА схемы Уилкса каждая микрокоманда сопоставляется с внутренним состоянием автомата. В связи с этим задача выбора числа микрокоманд оказывается весьма важной, так как от ее решения существенно зависит сложность МА.

Будем считать, что работа МА задается одной ЛСА, в процессе выполнения которой значения всех входящих в нее ЛУ изменяются только операторами этой ЛСА или остаются неизменными до окончания выполнения ЛСА.

Прежде чем изложить способ формирования микрокоманд, необходимо ввести несколько понятий.

Два члена ЛСА X_i и X_j являются противоречивыми, еслиг 1) они не могут быть выполнены одновременно; 2) X_i является ЛУ, а X_j — оператором, причем логическое условие X_i входит в распределение сдвигов оператора X_j.

Член ЛСА X_i, который может быть выполнен непосредственно перед членом X_j, назовем предшественником X_j. Тогда X_j является последователем X_i. Очевидно, у любого члена ЛСА может быть

несколько предшественников. У члена ЛСА, являющегося оператором, всегда только один последователь, а у логического условия (двузначного) —два последователя, за счет чего при выполнении ЛСА образуются разветвления.

Группа членов ЛСА образует ветвь, если каждый $(i+1)$-й член этой ветви является последователем i-го члена. Один и тот же член ЛСА может входить в несколько различных ветвей. Группу членов ЛСА назовем совместимой, если в каждой ветви, образуемой членами этой группы, не содержится противоречивых членов. Совместимую группу членов ЛСА назовем максимальной, если добавление в нее любого другого члена ЛСА превращает ее в несовместимую.

Из приведенных определений следует, что каждая максимальная совместимая группа членов ЛСА может интерпретироваться как отдельная микрокоманда. Такая микрокоманда состоит из набора как внутренних, так и внешних микроопераций, но при определенном наборе значений логических условий одновременно выполняются микрооперации, входящие лишь в одну ветвь микрокоманды.

С каждым внутренним состоянием и, микропрограммного автомата сопоставляется одна максимальная совместимая группа M_i членов ЛСА. Если первый член группы M_j является последователем одного из членов группы M_i, то в МА должен быть переход из внутреннего состояния a_t в a_l. Очевидно, при этом МА будет реализовать заданную ЛСА.

Для получения совместимой группы M_{x_i} (т. е. группы, первым членом которой является X_i) выписывается член X_i ЛСА. Если у X_i имеется один последователь (т. е. X_i является оператором), то он приписывается справа от X_i. При наличии двух последователей образуется разветвление и каждый из них выписывается справа от X_i на отдельной ветви. Этот процесс повторяется для всех вновь приписанных членов группы в каждой ветви.

Ветвь группы M_{x_i} обрывается, если в нее вошел последний оператор ЛСА, либо если в нее должен быть включен член ЛСА, который является противоречивым хотя бы с одним членом этой ветви, либо если после X_j необходимо выписать X_i, который уже вошел в другую ветвь группы M_{x_i}. В последнем случае необходимо поставить стрелку от X_j к X_i. Формирование группы заканчивается после того, как обрываются все ее ветви.

Если в первую группу вошли не все члены ЛСА, образуется вторая группа, начиная с наименьшего по порядку члена ЛСА, не вошедшего

в предыдущую группу. Так процесс повторяется до тех пор, пока каждый член ЛСА не войдет хотя бы в одну группу.
Рассмотрим способ формирования микрокоманд на примере следующей ЛСА:

$$\mathfrak{A} = p_1{}^1 \uparrow {}^1 A_1 p_2 \uparrow {}^2 A_2 \downarrow {}^1 A_3 p_3 \uparrow {}^3 p_4 \uparrow {}^4 \downarrow {}^6 A_4 \downarrow {}^2 p_5 \uparrow {}^5 \downarrow {}^3 A_5 \downarrow {}^4 A_6$$
$$p_1{}^2 \uparrow {}^6 A_7 \downarrow {}^5. \qquad (5.3)$$

Информация о возможности одновременного выполнения операторов задана в табл. 5.1.

Таблица 5.1

	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7
A_1	×							
A_2	×	∨						
A_3	×	×	×					
A_4	×	×	∨	∨				
A_5	×	∨	×	∨	∨			
A_6	×	∨	×	∨	×	×		
A_7	×	×	∨	×	×	∨	∨	
A_k	∨	∨	∨	∨	∨	∨	∨	∨

В клетке такой таблицы на пересечении столбца A_i и строки A_j ставится знак $\vee$, если операторы A_i и A_j можно выполнять одновременно в любом из возможных значений ЛСА, знак × ставится в противном случае.
Кроме того, задано следующее распределение сдвигов:

$$A_0 — \{p_1\};\ A_1 — \{p_4\};\ A_2 — \{p_3,\ p_5\};\ A_3 — \{p_1\};$$
$$A_4 — \{—\};\ A_5 — \{—\};\ A_6 — \{p_1\};\ A_7 — \{—\}.$$

Образуем максимальные совместимые группы членов ЛСА. Берем оператор A_0 и приписываем справа его последователя:

$$A_0 \rightarrow [p^1{}_1].$$

Так как p_1 входит в распределение сдвигов оператора A_0, то ветвь обрывается на предыдущем члене (т. е. A_0), а $p^1{}_1$ указывается в

квадратных скобках. В связи с тем что у A_0 имелся только один последователь, группа M_{A_0} состоит из одного оператора A_0, т. е. $M_{A_0} = \{A_0 \rightarrow [p_1^1]\}$.

Условимся, что члены, указанные в квадратных скобках, не входят в данную группу, а являются начальными членами других групп. Образуем остальные максимальные совместимые группы, при этом верхняя стрелка после p_i будет означать, что p_i =1, а нижняя — p_i =0:

$$\left.\begin{array}{l} M_{p_1^1} = \left\{ p_1^1 \left[\begin{array}{l} \rightarrow A_1 \rightarrow p_2 \left[\begin{array}{l} \rightarrow A_2 \rightarrow [A_3] \\ \rightarrow p_3 \left[\begin{array}{l} \rightarrow A_5 \rightarrow [A_6] \\ \rightarrow A_k \end{array} \right. \end{array} \right. \\ \rightarrow [A_3] \end{array} \right. \right\}; \\ M_{A_3} = \left\{ A_3 \rightarrow p_3 \left[\begin{array}{l} \rightarrow p_4 \left[\begin{array}{l} \rightarrow A_4 \rightarrow p_5 \left[\begin{array}{l} \rightarrow A_5 \rightarrow [A_6] \\ \rightarrow A_k \end{array} \right. \\ A_6 \rightarrow [p_1^2] \end{array} \right. \\ \rightarrow A_5 \rightarrow [A_6] \end{array} \right. \right\}; \\ M_{A_6} = \{A_6 \rightarrow [p_1^2]\}; \\ M_{p_1^2} = \left\{ p_1^2 \left[\begin{array}{l} \rightarrow A_7 \rightarrow A_к \\ \rightarrow A_4 \rightarrow p_5 \left[\begin{array}{l} \rightarrow A_5 [A_6] \\ \rightarrow A_к \end{array} \right. \end{array} \right. \right\}. \end{array} \right\} \quad (5.4)$$

Получено пять максимальных совместимых групп членов ЛСА. Так как оператор A_0 является фиктивным, то исключим из рассмотрения первую группу и сопоставим с каждой из четырех максимальных совместимых групп свою микрокоманду, т. е. одно внутреннее состояние МА. При этом получим структурную схему МА, приведенную на рис. 5.9.

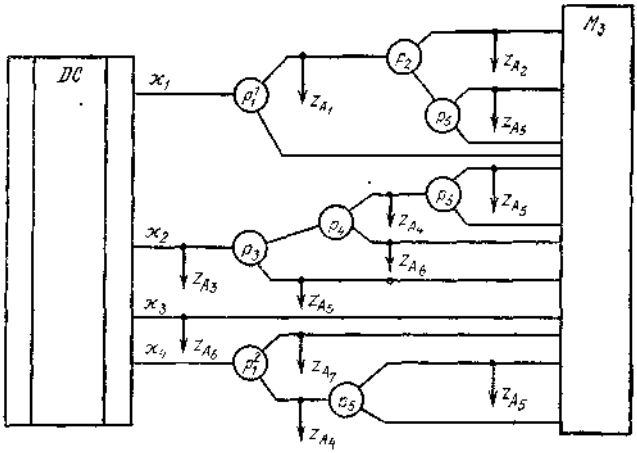


Рис. 5.9

Как видно, почти все полученные микрокоманды являются пересекающимися, так как одни и те же члены ЛСА входят в различные микрокоманды. Это является причиной того, что число вхождений операторов и ЛУ в полученную систему микрокоманд значительно превышает число их вхождений в исходную ЛСА. Однако ясно, что сложность схемы МА определяет именно полученная система микрокоманд. Таким образом, число членов заданной ЛСА не может служить хорошей характеристикой сложности схемы МА. Гораздо более точно определяет сложность схемы МА другая ЛСА, которую будем называть расширенной и обозначим $\mathfrak{A}^{p}_{i}$.

Расширенную ЛСА составляют по имеющейся системе микрокоманд, и число ее членов в точности равно числу вхождений операторов и логических условий в систему микрокоманд. Ясно, что даже если в исходной ЛСА $\mathfrak{A}_i$ нет повторяющихся членов, то в расширенной ЛСА $\mathfrak{A}^{p}_{i}$ одни и те же члены могут неоднократно повторяться.

Для получения системы микрокоманд (5.4) расширенная ЛСА будет иметь вид

$$\mathfrak{A}^{p} = p_1^{1}\uparrow^{1} A_1 p_2 \uparrow^{2} A_2 \downarrow^{1} A_3 p_3 \uparrow^{3} p_4 \uparrow^{4} A_4 p_5 \uparrow^{5} A_5 \downarrow^{7} A_6 \downarrow^{8} p_1^{2} \uparrow^{6} A_7$$
$$\omega \uparrow^{5} \downarrow^{6} A_1 p_5 \uparrow^{5} A_3 \omega \uparrow^{7} \downarrow^{4} A_6 \omega \uparrow^{8} \downarrow^{3} A_5 \omega \uparrow^{7} \downarrow^{2} p_5 \uparrow^{5} A_5 \omega \uparrow^{7} \downarrow^{5}.$$

Эта расширенная ЛСА содержит 21 член вместо 13 членов в исходной ЛСА (5.3).

Заметим, что в зависимости от заданных условий совместимости членов ЛСА и выбранного способа формирования микрокоманд по одной и той же ЛСА могут быть получены различные системы

микрокоманд и, следовательно, различные расширенные ЛСА. И так как схема МА строится по системе микрокоманд, то каждой схеме МА соответствует своя ЛСА $\mathfrak{A}^{p}$, тогда как одна и та же ЛСА $\mathfrak{A}$ будет соответствовать целому множеству схем МА. Следовательно, для оценки сложности схемы МА целесообразно использовать расширенную ЛСА.

Обратимся к полученной системе микрокоманд (5.4) и осуществим некоторые преобразования за счет выявления и многократного использования общих частей этих микрокоманд. В результате можно получить систему микрокоманд

$$
\left.
\begin{aligned}
M_{p_1^1} &= \left\{ p_1^1 \to \left[\begin{array}{l} \to A_1 \to p_2 \to \left[\begin{array}{l} \to A_2 \to [A_3] \\ \to \downarrow^1 p_5 \to \left[\begin{array}{l} \to A_5 \to [A_6] \\ \to A_K \end{array} \right. \end{array} \right. \\ \to [A_3] \end{array} \right. \right\}; \\
M_{A_3} &= \left\{ A_3 \to p_3 \to \left[\begin{array}{l} \to p_4 \to \left[\begin{array}{l} \to \downarrow^2 A_1 \to 1 \\ \to A_6 \to [p_1^2] \end{array} \right. \\ \to A_5 \to [A_6] \end{array} \right. \right\}; \\
M_{A_6} &= \{ A_6 \to [p_1^2] \}; \\
M_{p_1^2} &= \left\{ p_1^2 \to \left[\begin{array}{l} \to A_7 \to A_K \\ \to 2 \end{array} \right. \right\}.
\end{aligned}
\right\} \quad (5.5)
$$

Для перехода к одинаковым частям микрокоманд здесь применены пронумерованные стрелки, так же как в ЛСА. На рис. 5.10 приведена схема МА, построенная по этой системе микрокоманд.

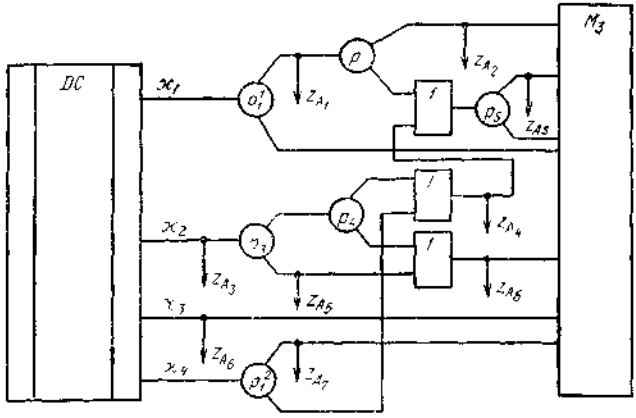


Рис. 5. 10

Посмотрим, возможно ли такое преобразование микрокоманд и приводит ли оно к упрощению схемы МА. Как видно, в отличие от схемы на рис. 5.9 в схеме МА появились дополнительные элементы ИЛИ, соответствующие неоднократному обращению к одному и тому же члену ЛСА. При этом оказывается, что одна и та же микрооперация должна выполняться при различных внутренних состояниях МА, что может вызвать определенные трудности при кодировании внутренних состояний автомата. Поясним это на примере схемы рис. 5.10.

Пусть внутренние состояния этого автомата закодированы следующим образом: $a_1 - 00$; $a_2 - 01$; $a_3 - 11$; $a_4 - 10$. Если выполнена микрооперация z_{A_1}, то автомат мог находиться в первом, втором или четвертом внутренних состояниях, так как эта операция входит в три микрокоманды. Тогда при выбранном коде в первом случае нужно будет включить оба ЭП (00→11), во втором — включить только первый ЭП (01→11) и в третьем — только второй ЭП (10→11). Очевидно, если используются ЭП с фиксацией воздействия с раздельными входами, то можно при любом из этих переходов включать оба ЭП и автомат будет работать правильно. При использовании ЭП со счетным входом в этом случае не удается обеспечить правильное функционирование автомата без введения в ЛСА проверок, характеризующих номер внутреннего состояния, в котором находится автомат. Сложность всего автомата должна быть оценена с учетом этих проверок; иногда при этом может оказаться, что вместо введения дополнительных проверок целесообразно общие члены различных микрокоманд повторить в каждой из них.

Многократное обращение к одной и той же внешней микрооперации также не упрощает схему матрицы M_1, а лишь превращает ее в многоуровневую схему, так как общее число выполнений какой-либо микрооперации при этом не уменьшается.

До сих пор сопоставляли микрокоманду с максимальной группой совместимых членов ЛСА. При этом достигается максимальное быстродействие, т. е. ЛСА выполняется за минимальное число тактов работы МА, а сам МА имеет минимальное или близкое к нему число внутренних состояний. Однако комбинационная часть автомата при этом может быть неминимальной. Использование в качестве микрокоманд немаксимальных групп совместимых членов ЛСА может привести к упрощению матриц M_1 и M_2 схемы МА.

Так, для нашего примера, выделив в отдельную микрокоманду общие части максимальных совместимых групп, можно получить следующую систему микрокоманд:

$$M_{p_1^1} = \left\{ p_1^1 - \left| \begin{array}{l} \to A_1 \to p_2 - \left| \begin{array}{l} \to A_2 \to [A_3] \\ \to [p_5] \end{array} \right. \\ \to [A_3] \end{array} \right. \right\};$$

$$M_{A_3} = \left\{ A_3 \to p_3 - \left| \begin{array}{l} \to p_4 - \left| \begin{array}{l} \to A_4 \to [p_5] \\ \to [A_6] \end{array} \right. \\ \to A_5 \to [A_6] \end{array} \right. \right\};$$

$$M_{A_6} = \{A_6 \to [p_1^2]\};$$

$$M_{p_1^2} = \left\{ p_1^2 - \left| \begin{array}{l} \to A_7 \to A_к \\ \to A_4 \to [p_5] \end{array} \right. \right\};$$

$$M_{p_5} = \left\{ p_5 - \left| \begin{array}{l} \to A_5 \to [A_6] \\ \to A_к \end{array} \right. \right\}.$$

Применение этой системы микрокоманд упрощает комбинационную часть МА, однако требует увеличения числа ЭП. Ситуация здесь аналогична той, которая возникает при синтезе автоматов классического типа: увеличение числа внутренних состояний автомата может привести к упрощению его комбинационной схемы.

Однако в настоящее время не существует алгоритма, позволяющего получать минимальную схему МА по заданной ЛСА.

В дальнейшем при формировании микрокоманд будем использовать максимальные группы совместимых членов ЛСА, что, как уже отмечалось, сокращает объем памяти МА и обеспечивает его наибольшее быстродействие.

При рассмотренном способе образования максимальных групп совместимых членов ЛСА, несмотря на размер получающихся при этом микрокоманд, характерным для них является то, что все входящие в каждую микрокоманду микрооперации выполняются одновременно за один микротакт. Такие микрокоманды будем называть *простыми.*

Дальнейшее упрощение структуры МА может быть достигнуто за счет перехода от простых к сложным микрокомандам. Для сложной микрокоманды характерно то, что входящие в нее микрооперации могут выполняться не одновременно, а последовательно во времени и набор этих микроопераций может изменяться в зависимости от значений условий, проверяемых при выполнении внутренних микроопераций, входящих в данную микрокоманду.

При формировании простых микрокоманд исходили из того, что каждый ФБ имеет один вход, воздействие с которого должно быть

снято после того, как этот ФБ закончил свою работу (т. е. выполнена соответствующая ему микрооперация). Однако на практике часто используются ФБ, имеющие не один, а два входа, один из которых можно назвать включающим, а другой — выключающим. Примерами таких ФБ могут быть триггеры, пневматические распределительные устройства с двусторонним приводом и др. Для управления такими ФБ требуются два сигнала и соответственно два оператора ЛСА— A_i (включающий) и A^*_i (выключающий). Сигнал, поданный на включающий (выключающий) вход ФБ, может воздействовать на него и после окончания работы ФБ, но обязательно должен быть снят как только появится сигнал на выключающем (включающем) входе ФБ, т. е. операторы A_i и A^*_i никогда не выполняются одновременно и могут быть названы *инверсными.*

Наличие инверсных операторов в ЛСА, описывающей работу микропрограммного автомата, позволяет построить автомат со сложными микрокомандами. Например, требуется осуществить последовательность выполнения операторов A_{i_1}, A_{i_2}, ..., A_{i_l}, причем ФБ, соответствующий оператору A_{ij} (j=1, 2,...,l), может быть включен лишь после того, как ФБ, соответствующий оператору $A_{i(j-1)}$, закончит свою работу, т. е. при формировании простых микрокоманд в каждую из них можно включить только по одному оператору и число внутренних состояний МА будет равно числу операторов. Однако, учитывая свойства двухвходовых ФБ, можно допустить выпол нение оператора A_{ij} одновременно с операторами $A_{i(j+1)}, \ldots, \ldots, A_{i(j+m)}$ $(m \leqslant l)$, если среди последних нет оператора A^*_{ij}, и сопоставить с каждым внутренним состоянием МА не один, а целую группу операторов, образующих сложную микрокоманду. Но теперь операторы $A_{ij}, \ldots, A_{i(j+m)}$, входящие в одну микрокоманду, должны выполняться последовательно, т. е. сигнал $z_{A_{i(j+1)}}$ не может появиться прежде, чем $ОФБ_{ij}$ не закончит свою работу. Чтобы при этом обеспечить заданную последовательность работы функциональных блоков, каждый раз перед выполнением оператора $A_{i(j+1)}$ необходимо проверить, выполнен ли оператор A_{ij}. Поэтому с ЛСА кроме обычных логических условий p, должны быть введены специальные ждущие логические условия, проверяющие выполнение соответствующих операторов. Будем обозначать такие логические условия через q_i, полагая q_i=1, если выполнен оператор A_i, и $\bar{q}_i$=1, если выполнен оператор A^*_i. Например, ЛСА может иметь вид

$$\mathfrak{A} = \downarrow^{1} A_1 q_1 \uparrow^{1} \downarrow^{2} A_2 q_2 \uparrow^{2} p_1 \uparrow^{1} \downarrow^{3} A_2^* \bar{q}_2 \uparrow^{3} \downarrow^{4} A_1^* \bar{q}_1 \uparrow^{4} \omega \uparrow^{1}.$$

По-прежнему будем считать, что в процессе выполнения ЛСА значения всех входящих в нее логических условий изменяются только операторами этой ЛСА. После того как алгоритм выполнен, значения условий p могут меняться извне, тогда как условия q не меняются. Другой особенностью логических условий q является то, что они не дают разветвлений в ЛСА, а как бы осуществляют выдержку времени, фиксируя момент выполнения соответствующих операторов. Поэтому такие ЛУ для простоты записи можно не включать в ЛСА, хотя соответствующие им логические блоки, конечно, будут входить в автомат.

Тогда ЛСА запишем в виде

$$\mathfrak{A} = \downarrow^{1} A_1 A_2 p_1 \uparrow^{1} A_2^* A_1^* \omega \uparrow^{1}.$$

На рис. 5.11 приведена структура МА со сложными микрокомандами, построенная по этой ЛСА.

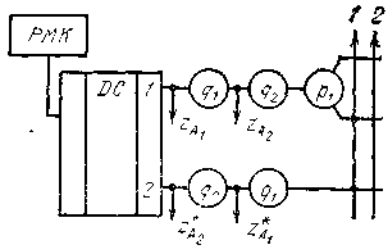


Рис. 5.11

Автомат имеет два внутренних состояния, тогда как при использовании простых микрокоманд для этой ЛСА, если операторы A_1 и A_2, A_1^* и A_2^* не могут выполняться одновременно, получился бы автомат с четырьмя внутренними состояниями, а при сопоставлении внутреннего состояния МА с одним членом ЛСА автомат имел бы пять внутренних состояний. Таким образом, укрупнение простых микрокоманд и образование сложных микрокоманд позволяют упростить структуру МА.

Если два оператора A_{ij} и $A_{i(j+1)}$ входят в одну простую микрокоманду, то сигналы $z_{A_{ij}}$ и $z_{A_{i(j+1)}}$ появляются одновременно и $ОБФ_{ij}$ и $ОФБ_{i(j+1)}$ работаютпараллельно. При вхождении операторов A_{ij} и $A_{i(j+1)}$ в две различные последовательно выполняемые микрокоманды сначала появляется сигнал z_{A_j}, включается $ОФБ_{ij}$ и только посге окончания его работы меняется

внутреннее состояние МА, исчезает сигнал $z_{A_{ij}}$ и появляется сигнал $z_{A_{i(j-1)}}$.

Если операторы A_{ij} и $A_{i(j+1)}$ включены в одну сложную микрокоманду, то сигнал $z_{A_{i(j-1)}}$ также может появиться лишь после того, как $ОФБ_{ij}$ закончит работу, но сигнал $z_{A_{ij}}$ при этом по-прежнему будет воздействовать на уже отработавший блок $ОФБ_{ij}$.

Способ формирования сложных микрокоманд аналогичен формированию простых микрокоманд, но теперь в одну микрокоманду могут входить все операторы, кроме инверсных. Поэтому при образовании сложных микрокоманд несколько изменится определение противоречивых членов ЛСА.

Два члена ЛСА X_i и X_j являются противоречивыми, если: 1) они являются инверсными операторами; 2) X_i является логическим условием, а X_j — оператором, который может быть выполнен после проверки X_i, причем логическое условие входит в распределение сдвигов оператора X_j.

Таким образом, в отличие от простой в сложную микрокоманду могут входить логическое условие p_i и оператор A_j, который меняет значение этого логического условия $(A_j — \{p_i\})$, если только проверка p_i должна быть осуществлена после выполнения оператора A_j.

Используя изложенный ранее способ формирования микрокоманд, с учетом этого определения противоречивости составим сложные микрокоманды для следующей ЛСА, ни одна пара операторов которой не может выполняться одновременно:

$$\mathfrak{U} = A_1 p_1 \uparrow^1 A_2 A^*_3 \downarrow^4 A^*_1 p_2 \uparrow^2 A_4 A_5 \omega \uparrow^3 \downarrow^1 A^*_2 A_3 \omega \uparrow^4 \downarrow^2 A^*_4 A_6 \downarrow^3.$$

Для этой ЛСА задано следующее распределение сдвигов:

$$A_1 — \{p_1,\ p_2\};\ A^*_1 — \{p_1,\ p_2\};\ A_2 — \{p_2\};\ A^*_2 — \{p_2\};$$

$$A_3 — \{—\};\ A^*_3 — \{—\};\ A_4 — \{p_1\};\ A^*_4 — \{p_1\};\ A_5 — \{—\};\ A_6 — \{—\}.$$

Начнем формирование сложных микрокоманд с группы N_{A1}, т. е. с группы, начинающейся с оператора A_1:

$$N_{A_1} = \left\{ A_1 \rightarrow p_1 — \begin{cases} \rightarrow A_2 \rightarrow A^*_3 \rightarrow \downarrow^1 [A^*_1] \\ \rightarrow A^*_2 \rightarrow A_3 \rightarrow 1 \end{cases} \right\};$$

$$N^*_{A_1} = \left\{ A^*_1 \rightarrow p_2 — \begin{cases} \rightarrow A_1 \rightarrow A_5 \rightarrow \downarrow^2 A_4 \\ \rightarrow A_3 \rightarrow A_6 \rightarrow 2 \end{cases} \right\}.$$

Получились две сложные микрокоманды; схема МА, внутренние состояния которого сопоставлены с этими микрокомандами, приведена на рис. 5.12.

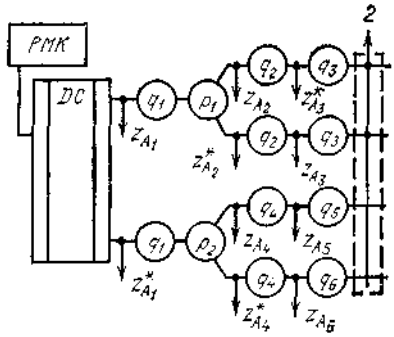


Рис. 5.12.

Для сравнения приведем совместимые группы членов этой ЛСА, соответствующие простым микрокомандам:

$$M_{A_1} = \{A_1 \to [p_1]\};$$

$$M_{p_1} = \left\{ p_1 - \begin{cases} \to A_2 \to [A^*_3] \\ \to A_2^* \to [A_3] \end{cases} \right\};$$

$$M_{A_3^*} = \{A_3^* \to [A_1']\}; \quad M_{A_3} = \{A_3 \to [A_1']\}; \quad M_{A_1^*} = \{A_1^* \to [p_2]\};$$

$$M_{p_2} = \left\{ p_2 - \begin{cases} \to A_4 - [A_5] \\ \to A_4^* \to [A_6] \end{cases} \right\};$$

$$M_{A_5} = \{A_5 \to A_K\};$$

$$M_{A_6} = \{A_6 \to A_1\}.$$

Автомат, построенный с использованием простых микрокоманд, имеет восемь внутренних состояний (рис. 5.13).

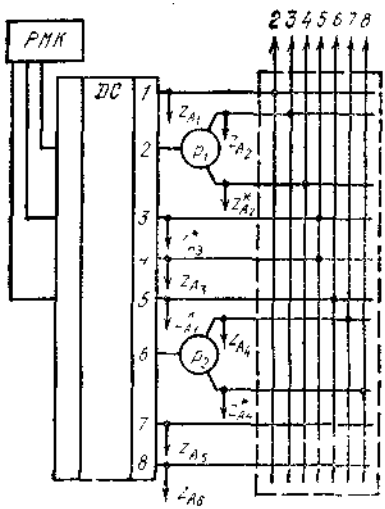


Рис. 5.13

Таким образом, переход к сложным микрокомандам усложняет матрицы M_1 и M_2, но упрощает дешифратор и уменьшает разрядность регистра микрокоманд.

5.3. Формирование микрокоманд для параллельного алгоритма

Один из возможных способов реализации параллельного алгоритма состоит в построении одного МА, который одновременно вырабатывает сигналы, соответствующие операторам различных параллельных участков алгоритма. При этом в одну микрокоманду должны входить члены параллельных подалгоритмов, что приводит к необходимости некоторого изменения метода формирования микрокоманд для ПЛСА.

Сохранив основные правила образования микрокоманд на основе максимальных групп совместимых членов ЛСА, сформулируем правила включения в микрокоманды операторов распараллеливания и соединения и отметим основные особенности микрокоманд, получаемых по ПЛСА.

Условимся, что оператор R_i несовместим со всеми своими предшественниками, следовательно, он не может входить в качестве последователя ни в одну ветвь микрокоманды, а всегда является начальным (или одним из начальных) членом микрокоманды.

Оператор R_i имеет несколько последователей и совместим с каждым из них. Поэтому в микрокоманду M_{R_i}, начинающуюся с оператора R_i, справа от него включаются все последователи, для каждого из которых затем рассматриваются по обычным правилам его последователи в соответствующем участке ПЛСА.

Следует иметь в виду, что различные ветви микрокоманды по-прежнему образуются только в результате проверки значений логических условий. Если в пределах параллельных участков нет логических условий, то все члены этих участков, вошедшие в одну микрокоманду, образуют одну ветвь. Однако каждый из этих членов имеет своего последователя, поэтому в отличие от микрокоманд, получаемых по ЛСА, формирование микрокоманды может начинаться не с одного, а с нескольких членов, принадлежащих различным параллельным участкам ПЛСА.

Если в параллельных участках ПЛСА имеются логические условия, то в микрокомандах образуются ветви, число которых определяется не числом параллельных участков, а числом логических условий, вошедших в микрокоманду. При данном наборе значений логических условий выполняется лишь одна ветвь микрокоманды, но так как в нее входят члены разных параллельных участков, то у нее будет не один, а несколько последователей, с которых начнется формирование следующей микрокоманды. Таким образом, наличие нескольких начальных членов является отличительной особенностью микрокоманд, формируемых по ПЛСА.

Рассмотрим правила включения в микрокоманду оператора S_j. Пусть этот оператор противоречив всем членам сходящихся в нем параллельных участков ПЛСА, тогда ни с одним из них он не может входить в одну ветвь микрокоманды. Поэтому все ветви микрокоманды обрываются на операторе S_j.

Для упрощения процесса формирования микрокоманд по ПЛСА введем понятие подчиненности членов ПЛСА оператору соединения S_j. Будем говорить, что член ПЛСА X_i подчинен оператору S_j, если X_i входит в параллельный участок ПЛСА, соединяющийся в операторе S_j. При формировании микрокоманд по ПЛСА каждый член X_i, подчиненный оператору S_j, отметим верхним индексом j (X_t^j). Очевидно, что член ПЛСА X_t^j и оператор S_j несовместимы. Поэтому если в одну ветвь микрокоманды входят члены разных параллельных участков и она обрывается на группе членов $[X_t^j S_j]$, то формирование следующей микрокоманды начинается только с члена X_t^j, а оператор S_j в эту микрокоманду не включается. Таким образом, оператор S_j будет исключен из всех микрокоманд, в которые входят

члены соединяющихся в S_j параллельных участков. Оператор S_j может быть включен в микрокоманду лишь после того, как в предыдущие микрокоманды будут включены все члены соединяющихся в нем параллельных участков. Это обеспечит возможность выполнения оператора S_j лишь после выполнения всех предшествующих ему параллельных участков ПЛСА.

Приведем несколько примеров формирования микрокоманд по ПЛСА. Рассмотрим вначале такую ПЛСА, в которой отсутствуют циклы в параллельных участках.

Возьмем приведенную ранее ПЛСА, исключив все операторы B_0 и B_k и изменив нумерацию стрелок, осуществим разметку подчиненных операторам соединения членов ПЛСА. Очевидно, один и тот же член ПЛСА может быть подчинен нескольким операторам соединения, тогда он отмечается несколькими верхними индексами. В результате получим

$$\mathfrak{A} = p_1 \uparrow {}^{1} A_1 \downarrow {}^{1} R_1 \omega \underset{3}{\uparrow}{}^{2} \overline{\downarrow}{}^{2} A_2{}^{2} A_3{}^{2} R_2{}^{2} \omega \underset{2}{\uparrow}{}^{3} \overline{\downarrow}{}^{3} A_7{}^{2} p_3{}^{2} \uparrow {}^{4} A_8{}^{2} \downarrow {}^{4} \omega \underset{}{\uparrow}{}^{5} \overline{\downarrow}{}^{3}$$
$$A_9{}^{2} p_4^2 \uparrow {}^{6} A_{10}^2 \downarrow {}^{6} \omega \underset{}{\uparrow}{}^{5} \overline{\downarrow}{}^{2} p_2^{1,2} \uparrow {}^{7} A_4^{1,2} \downarrow {}^{7} \omega \underset{}{\uparrow}{}^{8} \overline{\downarrow}{}^{2} A_5^{1,2} A_6^{1,2}$$
$$\omega \underset{}{\uparrow}{}^{8} \underset{2}{\overset{8}{\downarrow}} S_1^2 p_5^2 \uparrow {}^{9} A_{11}^2 \downarrow {}^{9} \omega \underset{}{\uparrow}{}^{5} \overline{\downarrow}{}^{5}_{3} S_3 A_{12}. \qquad \textbf{(5.6)}$$

Для простоты будем считать, что задано пустое распределение сдвигов и все условия совместимости заданы параллельными участками этой ПЛСА, т. е. все члены ПЛСА, входящие в параллельные участки, совместимы между собой, а все члены, принадлежащие последовательным участкам, несовместимы.

Начинаем формирование микрокоманд с первого члена ПЛСА (5.6), нумеруя подряд получаемые микрокоманды, так как в одной микрокоманде может быть теперь несколько начальных членов:

$$M_1 = A_0 \rightarrow p_1 \rightarrow \left| \begin{array}{l} \rightarrow A_1 \rightarrow [R_1] \\ \rightarrow [R_1] \end{array} \right. .$$

Обе ветви этой микрокоманды обрываются при включении оператора R_1, так как он несовместим ни с одним из своих предшественников. Следующая микрокоманда начинается с оператора R_1. Начальные члены следующих за оператором R параллельных участков в микрокоманде будем объединять фигурной скобкой:

$$M_2 = R_1 \begin{cases} A_2^2 \rightarrow [A_3^2] \\ p_2^{1,2} \begin{cases} \rightarrow A_4^{1,2} \rightarrow [S_1^2] \\ \rightarrow [S_1^2] \end{cases} \\ A_5^{1,2} \rightarrow [A_6^{1,2}]. \end{cases}$$

Верхние индексы членов ПЛСА определяют не только возможность включения операторов S_j в микрокоманду, но и говорят о совместимости членов ПЛСА. Если член X_i является последователем $X_m (X_m \rightarrow X_l)$, а X_i и X_m имеют один и тот же верхний индекс, то они входят в один последовательный участок ПЛСА и для нашего примера несовместимы. Поэтому операторы A_3^2, $A_6^{1,2}$, S_1^2 не могут быть включены в микрокоманду M_2.

Микрокоманда M_3 должна была бы начаться с членов A_3^2, $A_6^{1,2}$, S_1. Однако в соответствии с приведенным выше правилом $A_6^{1,2}$ и S_1^2 несовместимы, поэтому оператор S_1^2 не может быть включен в микрокоманду ,

$$j.M_3 = \begin{bmatrix} A_3^2 \rightarrow [R_2^2] \\ A_6^{1,2} \rightarrow [S_1^2]. \end{bmatrix}$$

Операторы R_2^2 и S_1^2 не связаны отношением следования и имеют одинаковые верхние индексы, значит, они входят в параллельные участки ПЛСА и являются совместимыми:

$$M_3 = \begin{cases} R_2^2 \begin{cases} A_7^2 \rightarrow p^2{}_3 \begin{cases} \rightarrow [A_8^2] \\ \rightarrow [S_2] \end{cases} \\ A_9^2 \rightarrow p_4^2 \begin{cases} \rightarrow [A_{10}^2] \\ \rightarrow [S_2] \end{cases} \end{cases} \\ S_1^2 \rightarrow p_5^2 \begin{cases} \rightarrow A_{11}^2 \rightarrow [S_2] \\ \rightarrow [S_2]. \end{cases} \end{cases}$$

Оператор S_2 несовместим ни с A_7^2, ни с A_{10}^2, поэтому не может быть включен ни в одну из следующих трех микрокоманд, которые являются возможными последователями микрокоманды M_4:

$$M_5 = \begin{bmatrix} A_8^2 \rightarrow [S_2] \\ A_{10}^2 \rightarrow [S_2]; \end{bmatrix}$$

$$M_6 = A_8^2 \rightarrow [S_2];$$

$$M_7 = A_{10}^2 \rightarrow [S_2].$$

После этих микрокоманд будут выполнены все члены предшествующих оператору S_2 участков ПЛСА, т. е.

$$M_8 = S_2 \rightarrow A_{12} \rightarrow A_к.$$

Таким образом, достаточно сложную ПЛСА (5.6) благодаря выявлению параллельных участков оказалось возможным реализовать в МА, имеющем восемь внутренних состояний, причем для выполнения алгоритма требуется шесть тактов работы МА.

Рассмотрим пример ПЛСА, содержащий циклы в параллельных участках, и отметим особенности формирования микрокоманд в этом случае. Пусть задана следующая ПЛСА, члены которой отмечены индексами соответствующих операторов соединения:

$$\mathfrak{A} = A_1 R_1 \omega \underline{\uparrow}_2^1 \overline{\downarrow}^1 A_2^{1,2} \downarrow^2 A_3^{1,2} A_8^{1,2} p_3^{1,2} \uparrow^2 A_9^{1,2}$$
$$\omega \underline{\uparrow}^3 \overline{\downarrow}^1 p_1^{1,2} \uparrow^4 A_4^{1,2} \downarrow^4 R_2^{1,2} \omega \underline{\uparrow}_2^5 \overline{\downarrow}^5 A_5^{1,2} p_2^{1,2} \uparrow^6 A_1^{1,2} \downarrow^6 A_7^{1,2}$$
$$\omega \underline{\uparrow}^3 \overline{\downarrow}^5 A_{10}^2 A_{11}^2 \omega \underline{\uparrow}^7 \overline{\downarrow}_2^3 S_1^2 A_{12}^2 \omega \underline{\uparrow}^7 \overline{\downarrow}_2^7 S_2 A_{13}. \qquad (5.7)$$

Сохраним прежние условия совместимости членов ПЛСА, т. е. члены параллельных участков совместимы, а члены последовательных участков несовместимы. Пусть для всех операторов, кроме A_3, A_5, A_8, задано пустое распределение сдвигов и $A_3 — \{p_3\}$, $A_5 — \{p_2\}$, $A_8 — \{p_3\}$. Сформируем микрокоманды по данной ПЛСА, пользуясь изложенной выше методикой:

$$M_1 = A_1 \rightarrow [R_1];$$

$$M_2 = R_1 \begin{cases} A_2^{1,2} \rightarrow [A_3^{1,2}] \\ p_1^{1,2} — \begin{cases} \rightarrow A_4^{1,2} \rightarrow [R_2^{1,2}] \\ \rightarrow [R_2^{1,2}]; \end{cases} \end{cases}$$

$$M_3 = \begin{cases} A_3^{1,2} \rightarrow [A_8^{1,2}] \\ R_2^{1,2} \begin{cases} A_5^{1,2} \rightarrow [p_2^{1,2}] \\ A_{10}^2 \rightarrow [A_{11}^2]; \end{cases} \end{cases}$$

$$M_4 = \begin{cases} A_8^{1,2} \to [p_3^{1,2}] \\ p_2^{1,2} - \begin{cases} \to A_6^{1,2} \to [A_7^{1,2}] \\ \to A_7^{1,2} \to [S_1^2] \end{cases} \\ A_{11}^2 \to [S_2]; \end{cases}$$

$$M_5 = \begin{cases} p_1^{1,2} - \begin{cases} \to A_9^{1,2} \to [S_1^2] \\ \to [A_3^{1,2}] \end{cases} \\ A_7^{1,2} \to [S_1^2]; \end{cases}$$

$$M_6 = p_3^{1,2} - \begin{cases} \to A_9^{1,2} \to [S_1^2] \\ \to [A_3^{1,2}]; \end{cases}$$

$$M_7 = S_1^2 \to A_{12}^2 \to [S_2];$$

$$M_8 = A_3^{1,2} \to [A_8^{1,2}];$$

$$M_9 = S_2 \to A_{13} \to A_к;$$

$$M_{10} = A_8^{1,2} \to [p_3^{1,2}].$$

Как видно, такая система микрокоманд позволит реализовать данную ПЛСА с наибольшим быстродействием, так как при выходе из цикла все члены параллельных участков оказываются выполненными и можно переходить к следующим участкам ПЛСА. Однако многие из полученных микрокоманд являются пересекающимися, так как члены ПЛСА (5.7), образующие цикл, входят в разные микрокоманды и, кроме того, для всех таких членов пришлось образовать отдельные микрокоманды. Таким образом, ускорение работы алгоритма достигается ценой усложнения комбинационной части МА и (как далее будет показано) увеличения числа его внутренних состояний. При наличии циклов в нескольких параллельных участках ПЛСА при таком способе формирования микрокоманд может происходить значительное усложнение схемы МА. Поэтому для ПЛСА, содержащих циклы в параллельных участках, предлагается другой способ формирования микрокоманд, который позволяет упростить схему МА, но несколько замедляет алгоритм.

Этот способ основан на том, что для всех членов параллельного участка, входящих в цикл, образуются отдельные микрокоманды независимо от членов других параллельных участков. И лишь в ветвь микрокоманды, которая соответствует выходу из цикла, включаются члены других параллельных участков, которые к этому моменту могут

быть выполнены. Таким образом, циклы параллельных участков выполняются последовательно, а члены ПЛСА, принадлежащие параллель- ным участкам и не входящие в циклы, выполняются одновременно.

Сформируем такие микрокоманды для ПЛСА (5.7):

$$M_1 = A_1 \to [R_1];$$

$$M_2 = R_1^{1,2} \begin{cases} A_2^{1,2} \to [A_3^{1,2}] \\ p_1^{1,2} - \begin{cases} \to A_4^{1,2} \to [R_2^{1,2}] \\ \to [R_2^{1,2}]. \end{cases} \end{cases}$$

От каждой из двух ветвей микрокоманды M_2 должен быть переход к микрокоманде, начинающейся с членов A_3, R_2. Однако, как видно из ПЛСА (5.7), оператор A_3 входит в цикл. Такие операторы в микрокомандах будем отмечать знаком $*$ и будем выполнять отдельно, не включая в их микрокоманды члены других параллельных участков (в данном случае R_2). Далее формируем микрокоманды для членов цикла начиная с оператора A_3:

$$M_3 = A_3^{1,2} \to [A_8^{1,2}]^*;$$

$$M_4 = A_8^{1,2} \to [p_3^{1,2}];$$

$$M_5 = p_3^{1,2} - \begin{cases} \to \begin{cases} A_9^{1,2} \to [S_1^2] \\ R_2^{1,2} \begin{cases} A_5^{1,2} \to [p_2^{1,2}] \\ A_{10}^2 \to [A_{11}^2]. \end{cases} \end{cases} \\ \to [A_3^{1,2}]. \end{cases}$$

Одна из ветвей микрокоманды M_5 (при $p_3 = 1$) соответствует выходу из цикла. В эту ветвь включим оператор R_2, который является последователем микрокоманды M_2 и относится к другому параллельному участку и, следовательно, совместим с A_9. Далее микрокоманды формируются по обычным правилам:

$$M_6 = \begin{cases} p_2^{1,2} - \begin{cases} \to A_6^{1,2} \to [A_7^{1,2}] \\ \to A_7^{1,2} \to [S_1^2] \end{cases} \\ A_{11}^2 \to [S_2]; \end{cases}$$

$$M_7 = A_7^{1,2} \to [S_1^2];$$

$$M_8 = S_1^2 \to A_{12}^2 \to [S_2];$$

$$M_9 = S_2 \to A_{13} \to A_к.$$

При таком способе формирования микрокоманд число их сократилось и сами микрокоманды упростились, что привело к упрощению схемы МА.

Когда требуется обеспечить максимально возможное быстродействие ЦБУ при наличии циклов в параллельных участках ПЛСА, более предпочтительной может оказаться реализация ЦБУ в виде иерархической структуры. При этом циклы разных параллельных участков могут быть выполнены одновременно в разных МА. Сложность и быстродействие ЦБУ будут существенно зависеть от выбора числа МА и от распределения функций между ЦМА и МА. После того как выбран способ реализации алгоритма с параллельными участками и сформированы микрокоманды для одного или нескольких МА, в которых будет реализована заданная ПЛСА, дальнейшие этапы синтеза МА совпадают с этапами синтеза МА по ЛСА.

Итак, разобрали метод формирования микрокоманд по ПЛСА для случая, когда все члены разных параллельных участков совместимы, т. е. условия зависимости микроопераций по ресурсам уже отражены в структуре параллельного алгоритма на начальном этапе проектирования УА при переходе от сети Петри к ПЛСА. Если этот этап не выполняется и управляющий алгоритм сразу представляется в виде ПЛСА, отдельные операторы, входящие в параллельные подалгоритмы, могут быть несовместимы из-за их зависимости по ресурсам, которая в этом случае должна быть задана дополнительно к ПЛСА. Очевидно, при этом не все члены разных параллельных участков алгоритма могут входить в одну микрокоманду. Это приводит к необходимости разработки нового метода формирования микрокоманд по ПЛСА, имеющей дополнительные ограничения на возможность одновременного выполнения операторов.

Рассматриваемый метод основан на использовании параллельной граф-схемы алгоритма ПГСА и состоит в ее преобразовании с учетом заданных условий совместимости членов На рис. 5.14 приведена ПГСА для следующей ПЛСА:

$$\mathfrak{M} = A_0 R_1 \omega \underline{\uparrow}_1^1 \overline{\downarrow}^1 A_1^{1,4} A_2^{1,4} A_3^{1,4} \omega \underline{\uparrow}^2 \overline{\downarrow}_3^2 S_1^4 A_4^4 \omega \underline{\uparrow}^3 \overline{\downarrow}^1 A_5^{1,2,4}$$
$$R_2 \omega \underline{\uparrow}_2^4 \overline{\downarrow}^4 \omega \underline{\uparrow}^2 \overline{\downarrow}^4 \omega \underline{\uparrow}^5 \overline{\downarrow}_2^5 S_2^{1,4} A_6^{1,4} \omega \underline{\uparrow}^2 \overline{\downarrow}^1 A_7^{1,2,3,4} R_3$$
$$\omega \underline{\uparrow}_3^6 \overline{\downarrow}^6 \omega \underline{\uparrow}^5 \overline{\downarrow}^6 A_8^{3,4} \omega \underline{\uparrow}^7 \overline{\downarrow}_3^7 S_3^4 A_{12}^4 \omega \underline{\uparrow}^3 \overline{\downarrow}^6 A_9^{3,4} R_4$$
$$\omega \underline{\uparrow}_2^8 \underline{\uparrow}^8 A_{10}^{3,4} \omega \underline{\uparrow}^7 \overline{\downarrow}^8 A_{11}^{3,4} \omega \underline{\uparrow}^7 \overline{\downarrow}_2^3 S_4 A_6. \qquad (5.8)$$

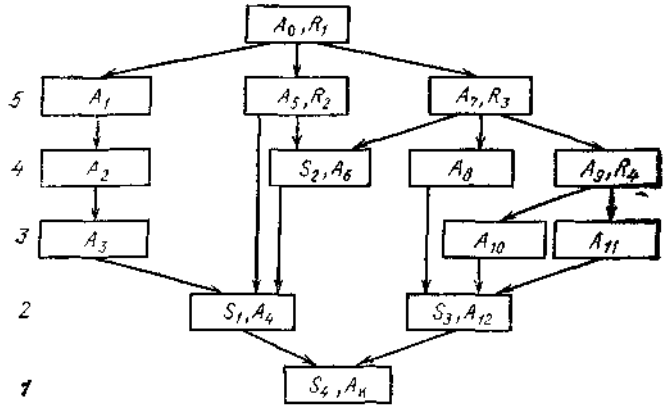


Рис. 5.14

Верхние индексы члена ПЛСА по-прежнему указывают номера тех операторов соединения, в которых содержащий его участок алгоритма объединен с другим параллельным участком. Для упрощения граф-схемы вершина оператора распараллеливания (соединения) объединена с вершиной его предшественника (последователя). Все вершины графа распределены по уровням *l*, нумерация которых начинается с конечной вершины. Дуги графа соединяют вершины, связанные отношением непосредственного следования. Иногда такую зависимость между микрооперациями, определяющую последовательность их выполнения, называют зависимостью по данным. Очевидно, если микрооперация A_i, производит данные, которые используются A_j, то она должна располагаться на более высоком уровне в ПГСА.

Параллельная граф-схема алгоритма содержит несколько путей, ведущих от начальной вершины в конечную. Самый длинный из них, т. е. путь, содержащий наибольшее число вершин, называется *критическим*. В общем случае может быть несколько таких путей. Число вершин в критическом пути определяет минимальное число микрокоманд, необходимое для данной ПГСА, так как все микрооперации, входящие в один такой путь, несовместимы и для каждой из них должна быть выделена отдельная микрокоманда.

Все вершины графа разделяют на две группы: стационарные и свободные. Вершины, входящие хотя бы в один критический путь, являются стационарными, так как уровень такой вершины не может быть изменен без увеличения числа микрокоманд. Каждой из свободных вершин может быть поставлено в соответствие некоторое подмножество уровней, на которых они могут располагаться без изменения общего числа микрокоманд.

В нашем примере (рис. 5.14) имеется семь путей от начальной вершины графа A_0 к конечной A_k. Причем пути $L_1 = \{A_0, A_1, A_2, A_3, A_4, A_к\}, L_2 = \{A_0, A_7, A_9, A_{11}, A_{12}, A_к\}, L_3 = \{A_0, A_7, A_9, A_{10}, A_{12}, A_к\}$ являются критическими и имеют длину, равную шести. Следовательно, для данной ПЛСА не может быть получено менее шести микрокоманд. Вершины $A_0, A_1, A_2, A_3, A_4, A_7, A_9, A_{10}, A_{11}, A_{12}, A_к$ являются стационарными, и любое допустимое изменение их уровней приведет к увеличению числа микрокоманд. Вершины A_5, A_6 и A_8 являются свободными, и их расположение в графе в некоторых пределах может меняться.

Число микрокоманд, равное длине критического пути, было бы не только необходимым, но и достаточным, если бы все микрооперации, входящие в параллельные пути, были совместимы. Однако, как уже отмечалось, существующая зависимость микроопераций по ресурсам накладывает ограничения на возможность включения их в одну микрокоманду. Такую зависимость будем задавать в виде таблицы с числом строк, равным числу различных операторов ПЛСА, каждому из которых сопоставлено множество совместимых с ним операторов (операторы *R* и S в таблицу не включаются). Заметим, что при составлении такой таблицы нет необходимости анализировать попарную совместимость всех микроопераций — достаточно лишь выявить независимость по ресурсам для микроопераций, входящих в разные параллельные участки ПЛСА. При этом для каждой микрооперации в таблице будут записаны полностью с ней совместимые, т е. не зависящие ни по данным, ни по ресурсам микрооперации.

Каждый член ПЛСА, кроме операторов распараллеливания и соединения, будем характеризовать двумя числами: уровнем *l*, который уже был введен для ПГСА, и весом *r*. *Весом микрооперации* A_i, назовем число вершин в самом длинном пути ПГСА от A_i к оператору конца $A_к$, включая A_i и $A_к$. При этом вес и уровень стационарной вершины совпадают, а для свободной вершины вес меньше ее уровня.

Пример таблицы, содержащей всю указанную информацию о микрооперациях для ПЛСА (5.8), приведен в табл. 5.2.

Таблица 5.2

Микрооперация	r	l	Совместимые микрооперации
A_0	6	6	—
A_1	5	5	A_5, A_6, A_7, A_8, A_{10}, A_{12}
A_2	4	4	A_5, A_6, A_{11}
A_3	3	3	A_5, A_7, A_8, A_9, A_{11}, A_{12}
A_4	2	2	A_8 A_{10}, A_{12}
A_5	4	5	A_8, A_9, A_{10}, A_{11}, A_{12}
A_6	3	4	A_9, A_{11}, A_{12}
A_7	5	5	—
A_8	3	4	A_9
A_9	4	4	—
A_{10}	3	3	A_{11}
A_{11}	3	3	—
A_{12}	2	2	—
A_p	1	1	—

Заметим, что для сокращения объема таблицы в ней для каждой микрооперации A_i выписаны только такие совместимые микрооперации A_j, для которых $j>i$. Такая таблица совместимости микроопераций с указанием их весов и уровней прежде всего может быть использована для определения числа микрокоманд, заведомо достаточного для выполнения ПЛСА при данной зависимости микроопераций по ресурсам. Для этого все микрооперации, имеющие вес r_i, объединим в множество N_i, которое затем разобьем на подмножества, включив в каждое из них максимальную группу совместимых микроопераций. Из полученных подмножеств выберем минимальное число таких подмножеств, которые включают все микрооперации множества N_i. Очевидно, если множество N_i содержит большое число микроопераций, то для решения данной задачи может быть использован метод минимизации числа внутренних состояний конечного автомата.

В результате для каждого множества N_i будет получено m_i групп совместимых микроопераций. При этом m_i уровней достаточно для распределения всех микроопераций множества N_i по уровням так, чтобы на каждом уровне находились только совместимые микрооперации, которые могут входить в одну микрокоманду.

Если вес микрооперации A_0 равен n, то величина $\sum_{i=1}^{n} m_i$ определяет верхнюю оценку числа микрокоманд ПЛСА при заданных условиях совместимости микроопераций. Поскольку микрооперации максимальных групп различных множеств N_i могут оказаться совместимыми, действительное число микрокоманд может быть

меньше. Следует иметь в виду, что такой подход позволяет получить верхнюю оценку числа микрокоманд только для линейного параллельного алгоритма.

Определим число микрокоманд, достаточное для выполнения ПЛСА (5.8) для зависимости микроопераций по ресурсам, приведенной в табл. 5.2. Полученный результат сведен в табл. 5.3.

Таблица 5.3

Вес r	N_i	Максимальные группы совместимых микроопераций	m_i
1	A_K		1
2	A_4, A_{12}	A_4, A_{12}	1
3	A_3, A_6, A_8, A_{10}, A_{11}	A_3, A_8; A_3, A_{11}; A_6, A_{11}; A_{10}, A_{11}	3
4	A_2, A_5, A_9	A_2, A_5; A_5, A_9	2
5	A_1, A_7	A_1, A_7	1
6	A_0	A_0	1

Поскольку $\sum_{i=1}^{6} m_i = 9$, девять микрокоманд достаточно для выполнения ПЛСА (5.8).

Имея верхнюю оценку числа микрокоманд, рассмотрим метод формирования микрокоманд по ПЛСА, который основан на изложенном ранее, но учитывает возможную несовместимость операторов — последователей оператора *R*. Если начальным членом микрокоманды M_i является оператор R_j, справа от него, как и раньше, выписываются его последователи, которые теперь являются лишь кандидатами на включение в данную микрокоманду. Далее необходимо определить, какие из них останутся в этой микрокоманде, а какие должны быть перенесены в последующие. Для этого на множестве этих микроопераций— последователей оператора R_j — образуются разбиения с минимальным числом блоков такие, что в каждый блок разбиении входят только совместимые микрооперации. Здесь возможны различные случаи: 1) получено одно разбиение, содержащее один блок, включающий, естественно, все микрооперации; 2) получено одно разбиение с несколькими блоками; 3) получено несколько разбиений (ясно, что все они содержат одинаковое число блоков).

В первом случае все микрооперации совместимы. Они включаются в микрокоманду M_i и справа от каждой из них выписывается заключенный в квадратные скобки ее последователь. Эти члены ПЛСА выписываются затем как кандидаты на включение в микрокоманду M_{i+1} и анализируются на совместимость. Если алгоритм является разветвленным и в микрокоманду M_i вошли не только внешние

(операторы), но и внутренние (логические условия) микрооперации, микрокоманда M_i имеет не одну, а несколько микрокоманд-последователей, число которых зависит от числа внутренних микроопераций в M_i. Все эти микрокоманды формируются после микрокоманды M_i.

Во втором случае, когда при формировании микрокоманды M_i получено многоблочное разбиение π_i микрооперации разных блоков этого разбиения не могут быть включены в одну микрокоманду и необходимо решить вопрос о том, какие из них целесообразно оставить в формируемой микрокоманде M_i. Для этого каждый блок разбиения будем характеризовать двумя параметрами: числом стационарных микроопераций в блоке *(s)* и его длиной, т. е. числом микроопераций *(v)*.

Из всех блоков разбиения π_i вначале выбирают блок, имеющий наибольшую величину s, а среди блоков, равноценных по этому параметру, — блок с наибольшей длиной. Если найдено несколько блоков с одинаковыми значениями s и *v*, то выбирают любой из них. Микрооперации выбранного блока включают в микрокоманду M_i, а их последователи являются кандидатами на включение в микрокоманды-последователи M_i. Микрооперации остальных блоков являются кандидатами на включение в микрокоманду M_{i+1}.

Если при формировании микрокоманды M_i образовалось несколько разбиений π^1_i, π^2_i . . . , то выбирают одно из них, содержащее блок с наибольшими значениями s и *v*. Микрооперации этого блока включают в M_i, определяют микрооперации— кандидаты на включение в микрокоманду M_{i+1}, образуют разбиения на множестве этих микроопераций, затем процесс повторяют.

В процессе формирования микрокоманд изменяются уровни микроопераций, не включаемых в рассматриваемую микрокоманду, и некоторых их последователей, так как выполнение этих микроопераций задерживается и они должны будут располагаться ближе к конечной вершине. В результате этого уровни отдельных микроопераций могут сравняться с их весами, т. е. они станут стационарными. Это необходимо учитывать при выборе блоков разбиений π_{i+1}, $\pi_{i+2...}$

Так же как и рассмотренный ранее, этот метод формирования микрокоманд по ПЛСА позволяет получить полную и замкнутую систему микрокоманд, так что каждая микрооперация войдет хотя бы в одну микрокоманду и для каждой микрокоманды M_i в полученной системе обязательно имеется все множество микрокоманд, являющихся последователями M_i.

Ниже приведена система микрокоманд, полученная изложенным методом для ПЛСА (5.8) и зависимости микроопераций по ресурсам, приведенной в табл. 5.2:

$$M_1 = A_0 \to [R_1];$$

$$M_2 = R_1 \begin{cases} \to A_1^{1,4} \to [A_2] \\ \to (A_5^{1,2,4}) \\ \to A_7^{1,2,3,4} \to [R_3] \end{cases} \quad \pi_2^1 = \{\overline{A_1,\ A_5};\ \overline{A_7}\} \quad \pi_2^2 = \{\overline{A_1,\ A_7};\ \overline{A_5}\}$$

$$M_3 \begin{cases} A_5^{1,2,4} \to [R_2] \\ (A_2^{1,4}) \\ R_3 \begin{cases} \to S_2^{1,4} \\ \to A_8^{3,4} \to]S_3] \\ A_9^{3,4} \to [R_4] \end{cases} \end{cases} \quad \pi_3^1 = \{\overline{A_2,\ A_5};\ \overline{A_8,\ A_9}\} \quad \pi_3^2 = \{\overline{A_2};\ \overline{A_5,\ A_8,\ A_9}\}$$

$$M_4 = \begin{cases} A_2^{1,4} \to [A_3] \\ R_2 \begin{cases} \to S_1^4 \\ \to S_2^{1,4} \to A_6^{1,4} \to [S_1] \end{cases} \\ S_3^4 \\ R_4 \begin{cases} \to (A_{10}^{3,4}) \\ \to A_{11}^{3,4} \to [S_3] \end{cases} \end{cases} \quad \pi_4^1 = \{\overline{A_2,\ A_6,\ A_{11}};\ \overline{A_{10}}\} \quad \pi_4^2 = \{\overline{A_2,\ A_6};\ \overline{A_{10},\ A_{11}}\}$$

$$M_5 = \begin{cases} A_{10}^{3,4} \to [S_3] \\ (A_3^{1,4}) \\ S_1^4 \\ S_2^4 \end{cases} \quad \pi_5 = \{\overline{A_{10}};\ \overline{A_3}\}$$

$$M_6 = \begin{cases} A_3^{1,4} \to [S_1] \\ S_3^4 \to A_{12}^4 \to [S_4] \end{cases} \quad \pi_6 = \{\overline{A_3,\ A_{12}}\}$$

$$M_7 = \begin{cases} S_1^4 \to A_4^4 \to [S_4] \\ S_4 \end{cases}$$

$$M_8 = S_4 \to A_k.$$

На примере нескольких микрокоманд поясним процедуру их получения. После микрокоманды M_1 содержащей только оператор A_0, составляется микрокоманда M_2. В нее входит оператор R_1, после которого начинаются три параллельных участка алгоритма. Следовательно, кандидатами на включение в микрокоманду M_2

являются микрооперации A_1, A_5, и A_7, однако не все они совместимы по ресурсам и, значит, не могут входить в одну микрокоманду.
На множестве этих микроопераций образуются два разбиения π_2^1 и π_2^2, которые выписаны справа от формируемой микрокоманды. Первый блок разбиения π_2^2 имеет s=2, тогда как для $\bar{m}_1^1$ и $\bar{m}_2^1$ $s=1$, а для $\bar{m}_2^2$ $s=0$.
Поэтому микрооперации A_1 и A_7, образующие блок $\bar{m}_1^2$, включаются в M_2. Микрооперация A_5 не включается в M_2, а является кандидатом для включения в микрокоманду M_3 вместе с последователями микроопераций A_1 и A_7 (A_2, A_3). Одновременно уменьшаются на единицу уровни микроопераций A_5 и A_6, и они становятся стационарными.
На множестве микроопераций A_5, A_2, A_8, A_9 —кандидатов на включение в микрокоманду M_3 — также образуются два разбиения, причем для $\bar{m}_1^1$ и $\bar{m}_2^2$ $s=2$. Однако блок $\bar{m}_2^2$ имеет большую длину, поэтому в M_3 включаются микрооперации A_5, A_8, A_9 . Аналогично выбирается блок разбиений π_1^4 и π_4^2. Для микрокоманды M_5 разбиение π_5 содержит два равноценных блока, выбирается любой из них (в данном случае $\bar{m}_5^1$). В результате получено восемь микрокоманд, т. е. меньше верхней оценки. На рис. 5.15 приведена ПГСА, преобразованная в соответствии с полученной системой микрокоманд.

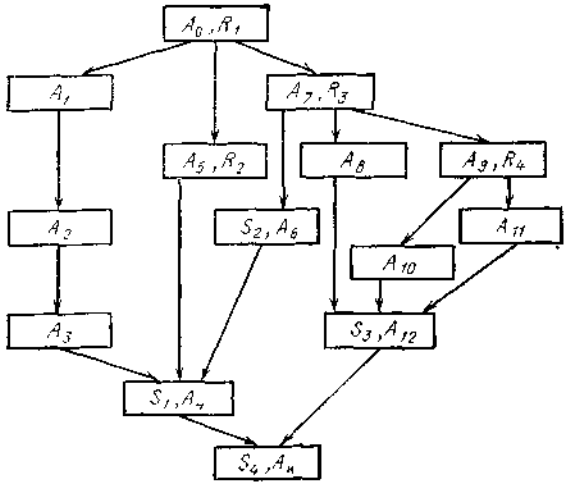


Рис. 5.15

Граф гмеет восемь уровней, все вершины одного уровня совместимы и образуют оану микрокоманду.

Очевидно, рассмотренный подход не гарантирует получение минимального числа микрокоманд, так как не позволяет учесть совместимость микроопераций, образующих последующие микрокоманды. От этого недостатка свободен точный метод формирования микрокоманд, основанный на использовании и анализе сети Петри. При этом от рассматриваемой ПЛСА в соответствии с правилами, приведенными ранее, осуществляется переход к сети Петри, описывающей тот же параллельный алгоритм. Кроме этого задаются условия совместимости операторов, например в виде треугольной таблицы. Для того, чтобы осуществить анализ сети Петри с учетом совместимости операторов, эти условия необходимо выразить в терминах языка сетей Петри. Это можно сделать, если ввести дополнительные позиции (аналогичные блокирующим), которые используются для устранения тупиковых состояний.

Если операторы ПЛСА A_i и A_j несовместимы, т. е. клетка i, j треугольной таблицы содержит знак ×, переходы t^i_t и $t^j_д$ сети Петри должны иметь общую входную позицию $a_б$, которая запретит их одновременную реализацию. Эта позиция является и выходной для блокируемых ею переходов, так как после выполнения одной из несовместимых микроопераций сразу может выполняться другая, для чего позиция $a_б$ должна содержать метку. Сопоставив таким образом с каждой клеткой треугольной таблицы, содержащей знак ×, свою блокирующую позицию, можно ввести в сеть Петри все заданные условия зависимости операторов по ресурсам. Очевидно, число таких позиций может быть очень большим, что затруднит анализ сети Петри. Однако оказывается возможным сократить число блокирующих позиций, если вводить общую позицию не для каждой пары несовместимых операторов, а для некоторого их подмножества. Для этого все множество операторов надо разбить на минимальное число подмножеств так, чтобы в каждое из них входили только зависимые по ресурсам операторы. Эта задача может быть решена с помощью метода получения максимальных групп совместимых состояний конечного автомата.

Дальнейшее укрупнение этих подмножеств и, следовательно, сокращение числа блокирующих позиций сети Петри можно осуществить, если учесть, что входящие в последовательный подалгоритм операторы не выполняются одновременно и, значит, также могут быть включены в подмножество несовместимых операторов, если это позволит сократить их число. При этом общую входную позицию будут иметь и некоторые последовательно

реализуемые переходы, однако это не повлияет на функционирование сети Петри.

Рассмотрим пример управляющего алгоритма, заданного ПГСА рис. 5.16.

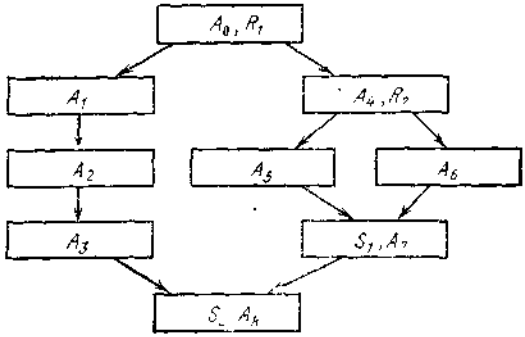


Рис. 5.16

Условия совместимости зададим в виде треугольной таблицы, строки и столбцы которой соответствуют операторам, а каждая клетка содержит одну из трех меток: Δ, $\times$, $\vee$. Если операторы A_i и A_j входят в последовательный участок алгоритма, то в клетке *i, j* ставится метка Δ, если A_i и A_j входят в параллельные участки, но зависят по ресурсам, то клетка *i, j* заполняется меткой $\times$. Если операторы A_i и A_j входят в параллельные участки и не зависят по ресурсам, в клетке *i, j* ставится метка $\vee$.

В табл. 5.4 приведен пример такой таблицы для графа, данного на рис. 5.16. (Считаем, что операторы A_0 и $A_к$ несовместимы с другими операторами, поэтому для упрощения таблицы в нее не включены.)

Таблица 5.4

2	Δ					
3	Δ	Δ				
4	$\times$	$\times$	$\times$			
5	$\vee$	$\times$	$\times$	Δ		
6	$\vee$	$\times$	$\times$	Δ	$\times$	
7	$\vee$	$\vee$	$\times$	Δ	Δ	Δ
	1	2	3	4	5	6

Для операторов, связанных знаком ×, получим шесть подмножеств зависимых по ресурсам операторов: $\{1, 4\}$, $\{2, 4\}$, $\{3, 4\}$, $\{2, 5, 6\}$, $\{3, 5, 6\}$, $\{3, 7\}$. Учет операторов, связанных отношением следования, позволяет получить три подмножества несовместимых операторов: $\{1, 4\}$, $\{2, 3, 4, 5, 6\}$, $\{3, 7\}$. Сопоставив с ними блокирующие позиции $a^1_б$, $a^2_б$, $a^3_б$ соответственно, получим сеть Петри для ПГСА (рис. 5.16) и условий совместимости, приведенных в табл. 5.4 (рис. 5.17).

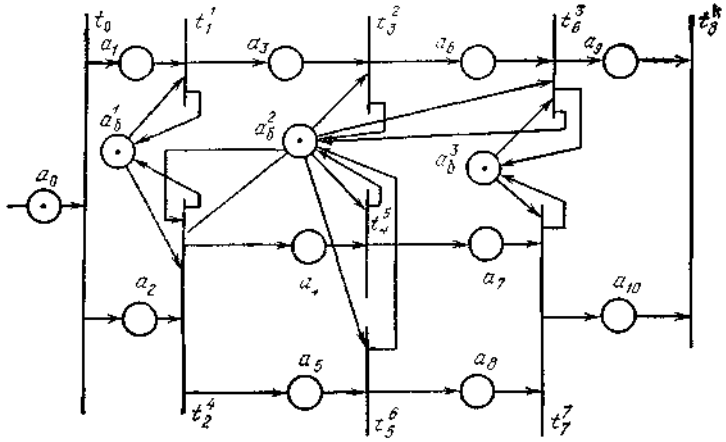


Рис. 5.17

Для полученной сети Петри составляется граф статических состояний. Поскольку в статическом состоянии ни один из переходов не находится в стадии реализации, а после реализации переходов блокирующие позиции вновь заполняются метками, в статическом состоянии все блокирующие позиции содержат метки. Поэтому для упрощения графа маркировок блокирующие позиции не будем включать в его вершины, которые различаются маркировкой только основных внутренних позиций. Каждая дуга графа помечается номерами одновременно реализуемых переходов и определяет таким образом операторы ПЛСА, которые могут быть включены в одну микрокоманду. Ясно, что минимальное число микрокоманд соответствует кратчайшему пути графа статических состояний сети Петри. Число микрокоманд равно числу дуг в этом пути, а их состав определяется переходами, которыми помечены дуги (следует помнить, что номера операторов совпадают с верхними индексами переходов).

На рис. 5.18 приведен граф статических состояний сети Петри рис. 5.17.

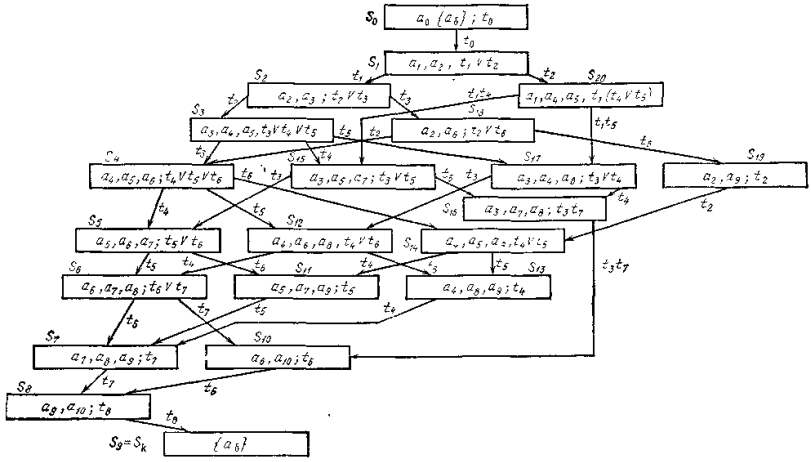


Рис. 5.18

Как видно, большинство путей графа имеют максимальную длину, равную девяти, что соответствует последовательному выполнению операторов. Единственный кратчайший путь имеет длину, равную семи, следовательно, ПГСА рис. 5.15 может быть разбита на семь микрокоманд:

$$M_1=A_0;\ M_2=A_4;\ M_3=A_1,\ A_5;\ M_4=A_6;$$

$$M_5=A_2,\ A_7;\ M_6=A_3;\ M_7=A_к.$$

Заметим, что рассмотренный ранее приближенный метод формирования микрокоманд по ПЛСА для данного примера может не привести к получению минимального решения, поскольку все пути ПГСА (рис. 5.15) являются критическими и все вершины стационарными, т. е. введенные критерии не позволяют осуществлять направленный выбор блоков разбиений. И все же приближенный метод необходим в тех случаях, когда заданная ПЛСА является сложной и анализ графа статических состояний сети Петри оказывается практически невозможным.

Рассмотренные методы формирования микрокоманд применимы и к разветвленным алгоритмам, но при этом следует иметь в виду, что минимальное число микрокоманд не всегда обеспечивает

максимальное быстродействие МА, реализующего этот алгоритм. Это объясняется тем, что число микрокоманд разветвленного алгоритма может не совпадать с числом тактов работы МА, необходимых для его выполнения. Поэтому при формировании микрокоманд необходимо знать, что является основным критерием при синтезе МА — число его внутренних состояний, определяемое числом микрокоманд, или быстродействие. Очевидно, для линейного алгоритма оба эти критерия совпадают, т. е. МА, имеющий минимальное число микрокоманд, обладает наибольшим быстродействием.

Система микрокоманд определяет объем памяти МА и его граф перехода. Следующий этап синтеза состоит в кодировании внутренних состояний МА.

5.4. Кодирование внутренних состояний микропрограммного автомата

Этап кодирования внутренних состояний при синтезе МА является столь же ответственным, как и при синтезе автоматов общего класса, так как от выбранного варианта кодирования зависят многие свойства МА, и в том числе сложность его комбинационной части.

Не останавливаясь на всех задачах, возникающих на этапе кодирования внутренних состояний МА, рассмотрим лишь влияние кодирования на сложность матрицы M_3 и дадим некоторые рекомендации по выбору кода, позволяющего упростить эту часть схемы МА. Под сложностью матрицы M_3 будем понимать число точек пересечения в ней, т. е. общее число входов в логические элементы ИЛИ, образующие блок M_3. Следует иметь в виду, что способ определения функций возбуждения элементов памяти зависит от типа выбранных элементов памяти. Поэтому при одном и том же коде матрица M_3 может иметь различную сложность в зависимости от свойств используемых элементов памяти. В дальнейшем будем считать, что в регистре микрокоманд МА используются элементы памяти с фиксацией воздействия с раздельными входами. Сигнал на включающий (выключающий) вход такого элемента должен быть подан каждый раз, когда необходимо перевести элемент памяти из нерабочего (рабочего) в рабочее (нерабочее) состояние.

Рассмотрение даже простых примеров показывает, что сложность матрицы может меняться в два раза и более в зависимости от выбранного кода. Поэтому выбор кода может существенно влиять на сложность всего МА.

Пусть условия работы МА заданы следующей ЛСА:

$$\mathfrak{A} = A_1\downarrow^3 A_2 p_1\uparrow^1 p_2\uparrow^2 p_3\uparrow^3 A_3\downarrow^2 A_4 p_2\uparrow^3 p_4\uparrow^4\downarrow^1 A_5\downarrow^4. \quad (5.9)$$

Закодируем внутренние состояния этого МА при следующей структуре микрокоманд:

$$M_1 = A_1 \rightarrow [A_2];$$

$$M_2 = A_2 \rightarrow [p_1];$$

$$M_3 = p_1 - \begin{cases} \rightarrow p_2 - \begin{cases} \rightarrow p_3 - \begin{cases} \rightarrow A_3 \rightarrow [A_4] \\ \rightarrow [A_2] \end{cases} \\ \rightarrow [A_1] \end{cases} \\ \rightarrow A_5 \rightarrow [A_K] \end{cases}$$

$$= A_4 \rightarrow [p_2]; \quad M_5 = p_2 - \begin{cases} \rightarrow p_3 - \begin{cases} \rightarrow A_5 \rightarrow [A_K] \\ \rightarrow [A_K] \end{cases} \\ \rightarrow [A_2] \end{cases}$$

$$M_6 = A_K.$$

Не раскрывая всех блоков МА, построим матрицу M_3 при выбранных шести микрокомандах для двух различных вариантов кодирования, которые приведены на соответствующих схемах автомата (рис. 5.19,а и *б).*

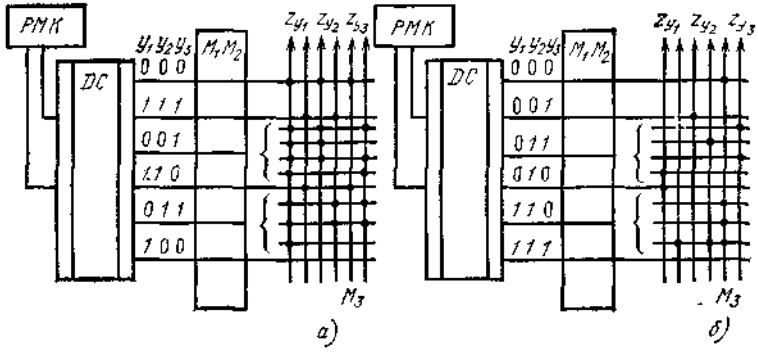


Рис. 5.19

Как видно, при первом варианте кодирования (рис. 5.19,а) матрица M_3 содержит в 2 раза больше точек, чем при втором (рис. 5.19,б). Конечно, желательно иметь возможность оценивать сложность матрицы M_3, не осуществляя ее построения, а зная лишь выбранный код. Введем для этого понятия «вес разбиения» и «вес кода».

При использовании s-разрядного кода на множестве внутренних состояний автомата образуется s двухблочных разбиений, каждое из которых соответствует одному разряду кода, т. е. одному элементу

памяти. Так, в данном примере при первом варианте кодирования образовалось следующее множество двухблочных разбиений:

$$\pi_1 = \{\overline{1, 3, 5};\ \overline{2, 4, 6}\};\quad \pi_2 = \{\overline{1, 3, 6};\ \overline{2, 4, 5}\};$$

$$\pi_3 = \{\overline{1, 4, 6};\ \overline{2, 3, 5}\}.$$

В зависимости от решаемой на этапе кодирования задачи к разбиениям, на основе которых строят код, могут предъявлять различные требования, однако одно условие является необходимым во всех случаях: произведение двухблочных разбиений, используемых для кодирования, должно быть нулевым, т е

$$\pi_1\pi_2 \ldots \pi_s = \pi_0.$$

В данном случае представляет интерес влияние каждого из разбиений на сложность матрицы M_3

Весом ρ_i двухблочного разбиения π_i назовем число переходов между всеми состояниями автомата, входящими в различные блоки этого разбиения. Для определения веса разбиения удобно пользоваться ориентированным графом переходов МА G или соответствующей ему матрицей смежности μ. Вершины графа переходов соответствуют внутренним состояниям МА.

Две вершины x_i и x_j соединены ребром, направленным из x_i в x_j, если в МА имеется переход из состояния x_i в x_j. Вес ребра x_i, x_j соответствует числу переходов между этими состояниями при различных наборах значений логических условий. Состояния входа МА, при которых осуществляются переходы, в данном случае нас не интересуют и на графе не указываются. На рис. 5.20 приведен граф переходов для МА, заданного ЛСА (5.9), при выбранных ранее микрокомандах.

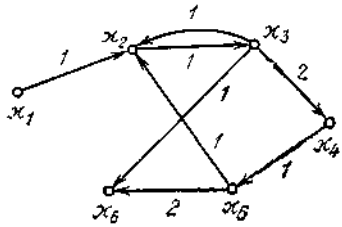


Рис. 5.20

Таблица 5.5 есть матрица смежности, соответствующая этому графу.

Таблица 5.5

	1	2	3	4	5	6
1		1				
2			1			
3		1		2		1
4					1	
5		1				2

Строки и столбцы матрицы также соответствуют внутренним состояниям МА. Элемент *i, j* матрицы равен весу ребра $\varkappa_i$, $\varkappa_j$ графа переходов.

Для определения веса ρ_i интересующего нас разбиения π_i в матрице μ нужно вычеркнуть все элементы, соответствующие переходам между теми состояниями МА, которые входят в один блок π_i . Сумма оставшихся членов матрицы μ дает вес разбиения π_i . Определим вес разбиения $\pi_3=\{\overline{1,\ 4,\ 6};\ \overline{2,\ 3,\ 5}\}$, которое используется в первом варианте кодирования. Исключив из матрицы μ элементы 2, 3; 3, 2 и 5, 2, поскольку состояния 2, 3 и 5 входят в один блок π_3, получим $\rho_3=7$. Вес разбиения π_i показывает, сколько элементов (это могут быть диоды или транзисторы) необходимо включить в матрицу M_3 для того, чтобы в соответствии с этим разбиением менять состояния *i*-го элемента памяти. Из схемы на рис. 5.19,а видно, что для изменения состояний третьего элемента памяти действительно требуется семь элементов в M_3.

Очевидно, сумма весов всех разбиений, используемых для кодирования, дает вес когда ρ_{Ki}, который определит сложность блока M_3 схемы МА.

Для первого варианта кодирования рассматриваемого примера получим: $\rho_2=7$, $\rho_1=10$ (при этом в матрице μ, сохраняются все элементы). В результате

$$\rho_{K_1} = \rho_1 + \rho_2 + \rho_3 = 24,$$

что соответствует 24 элементам в матрице M_3.

Для второго варианта кодирования этого же МА получим

$$\tau_1 = \{\overline{1, 2, 3, 4};\ \overline{5, 6}\}, \quad \rho_1 = 3;$$

$$\tau_2 = \{\overline{1, 2};\ \overline{3, 4, 5, 6}\}, \quad \rho_2 = 3;$$

$$\tau_3 = \{\overline{1, 4, 5};\ \overline{2, 3, 6}\}, \quad \rho_3 = 6; \quad \rho_{K_2} = 12.$$

Таким образом, зная вес кода, которым кодируются внутренние состояния МА, можно определить число элементов в матрице M_3, не осуществляя ее построения. Задача получения минимальной матрицы M_3 сводится к нахождению кода, имеющего минимальный вес. Наиболее простой, но требующий значительного перебора способ получения кода с минимальным весом состоит в выполнении последовательности следующих процедур:

1) в соответствии с числом полученных на предыдущем этапе синтеза МА микрокоманд выбрать число элементов памяти s;

2) для выбранного числа *s* составить множество допустимых двухблочных разбиений. Двухблочное разбиение является допустимым, если выполняется условие

$$1 +]\log_2 \#m[= s,$$

где $\#m$ — число состояний в большем блоке рабиения, т. е. использование допустимого разбиения не приводит к увеличению числа элементов памяти;

3) по матрице μ определить вес каждого двухблочного разбиения;

4) из полученного множества разбиений выбрать *s* разбиений, произведение которых является нулевым, а сумма весов минимальной.

Рассмотрим получение кода с минимальным весом для автомата, заданного ЛСА (5.9) и графом переходов (рис. 5.20). В табл. 5.6 приведены все возможные допустимые разбиения для автомата с шестью внутренними состояниями и вес каждого из разбиений, полученный по матрице и (см. табл. 5.5).

Таблица 5.6

№ разбиения	π_i	ρ_i	№ разбиения	π_i	ρ_i
1	$\{\overline{1, 2}; \overline{3, 4, 5, 6}\}$	3	14	$\{\overline{4, 6}; \overline{1, 2, 3, 5}\}$	6
2	$\{\overline{1, 3}; \overline{2, 4, 5, 6}\}$	6	15	$\{\overline{5, 6}; \overline{1, 2, 3, 4}\}$	3
3	$\{\overline{1, 4}; \overline{2, 3, 5, 6}\}$	4	16	$\{\overline{1, 2, 3}; \overline{4, 5, 6}\}$	4
4	$\{\overline{1, 5}; \overline{2, 3, 4, 6}\}$	5	17	$\{\overline{1, 2, 4}; \overline{3, 5, 6}\}$	6
5	$\{\overline{1, 6}; \overline{2, 3, 4, 5}\}$	4	18	$\{\overline{1, 2, 5}; \overline{3, 4, 6}\}$	5
6	$\{\overline{2, 3}; \overline{1, 4, 5, 6}\}$	5	19	$\{\overline{1, 2, 6}; \overline{3, 4, 5}\}$	6
7	$\{\overline{2, 4}; \overline{1, 3, 5, 6}\}$	7	20	$\{\overline{1, 3, 4}; \overline{2, 5, 6}\}$	5
8	$\{\overline{2, 5}; \overline{1, 3, 4, 6}\}$	6	21	$\{\overline{1, 3, 5}; \overline{2, 4, 6}\}$	10
9	$\{\overline{2, 6}; \overline{1, 3, 4, 5}\}$	7	22	$\{\overline{1, 3, 6}; \overline{2, 4, 5}\}$	7
10	$\{\overline{3, 4}; \overline{1, 2, 5, 6}\}$	4	23	$\{\overline{1, 4, 5}; \overline{2, 3, 6}\}$	6
11	$\{\overline{3, 5}; \overline{1, 2, 4, 6}\}$	9	24	$\{\overline{1, 4, 6}; \overline{2, 3, 5}\}$	7
12	$\{\overline{3, 6}, \overline{1, 2, 4, 5}\}$	6	25	$\{\overline{1, 5, 6}; \overline{2, 3, 4}\}$	4
13	$\{\overline{4, 5}; \overline{1, 2, 3, 6}\}$	5			

Выбираем разбиение π_i с наименьшим весом и подбираем к нему такое разбиение π_j (тоже имеющее наименьший вес), чтобы в наибольший блок разбиения $\pi_i\pi_j$ входило не более двух состояний, так как в противном случае потребуется более трех элементов памяти. Этим требованиям удовлетворяют разбиения

π_1 и π_{15}: $\pi_1\pi_{15}=\{\overline{1, 2}; \overline{3, 4}; \overline{5, 6}\}$. Очевидно, третье разбиение, которое будет использоваться для кодирования, должно разделить состояния, входящие в один блок разбиения $\pi_1\pi_{15}$, и иметь наименьший вес. Этим двум условиям удовлетворяет разбиение π_{23}, так как ни одно из разбиений с весами 4 и 5 не удовлетворяет первому условию. Легко проверить, что минимальный вес кода, которым можно закодировать внутренние состояния данного МА при выбранных микрокомандах, действительно равен 12. Кроме указанного имеется еще несколько подмножеств разбиений, которые также образуют коды весом 12.

Ясно, что при большом числе элементов памяти способ получения кода с минимальным весом, основанный на переборе всех двухблочных разбиений, может оказаться слишком трудоемким. Поэтому представляет интерес получение кода, который близок к минимальному и для получения которого потребовался бы меньший перебор. При этом, как и обычно при использовании приближенных

методов, желательно имен, оценки, позволяющие определить, насколько выбранным код отличается от кода с минимальным весом. Нетрудно понять, что сумма всех элементов матрицы μ дает вес кода $\rho_{K_{\text{мин}}}$, меньше которого невозможно получить для синтезируемого МА при выбранной структуре микрокоманд и минимальном числе элементов памяти (при этом учитываются и элементы $М_3$, образующие элементы ИЛИ с одним входом). Действительно, поскольку кодирование должно быть однозначным, любые два внутренних состояния a_i, a_j хотя бы в одном разбиении, используемом для кодирования, должны находиться в разных блоках. Если любые два внутренних состояния a_i, a_j, между которыми есть переход (соседние состояния), входят в разные блоки только одного разбиения, то полученный код будет иметь минимально возможный для данного графа переходов вес, равный сумме всех элементов матрицы μ. Очевидно, этот код будет соседним K_c, так как окажется, что всем соседним состояниям приписаны соседние кодовые комбинации, отличающиеся только одним разрядом. Однако известно, что соседнее кодирование возможно не для всех автоматов и простейшим критерием невозможности соседнего кодирования является, например, наличие нечетных контуров в графе переходов. Таким образом, если для данного графа переходов возможно получить соседний код, то этот код будет иметь минимальный вес, т. е. $\rho_{K_c} = \rho_{\text{мин}}$.

Заметим, что рассмотренный выше способ кодирования, основанный на выборе s разбиений из всех двухблочных допустимых разбиений, также даст соседний код, если он возможен для данного графа переходов. Во всех остальных случаях будет получен код, который не является соседним, но имеет наименьший возможный для данного графа вес.

Расстоянием t_{ij} между двумя кодовыми комбинациями, приписанными внутренним состояниям a_i, a_j, назовем число разрядов, значениями которых отличаются эти кодовые комбинации. Расстоянием между произвольным K и соседним K_c кодами будем называть величину $t = \Sigma (t_{ij} - 1)$ (сумма берется по всем i и j, соответствующим соседним состояниям). Например, код $K = \{000\ 011\ 111\ 110\ 101\ 100\}$ для графа переходов на рис. 5.20 находится на расстоянии 2 от соседнего кода.

Существенно, что если любой соседний код имеет один и тот же вес, то два кода K_n и K_m, находящиеся на равном расстоянии от соседнего кода, могут иметь разные веса. Это объясняется наличием ребер с разными весами и двусторонних переходов в графе переходов МА. Например, коды K_1 = = {000 100 011 101 001 111} и

K_2={000 011 110 101 111 100} для графа переходов на рис. 5.20 находятся на расстоянии 4 от соседнего кода, но имеют веса 14 и 17 соответственно. Поэтому при построении кода, близкого к минимальному, прежде всего желательно приписывать соседние кодовые комбинации тем состояниям, между которыми имеется наибольшее число переходов.

Для формирования разбиений, образующих такой код, более удобно пользоваться не матрицей, а просто списком, в котором указываются соседние состояния и число переходов между этими состояниями. Взяв пару соседних состояний x_i и x_j с наибольшим числом переходов, помещаем состояния x_i и x_j в один и тот же блок (s—1)-го разбиения и в разные блоки только одного разбиения. Затем по блокам этих разбиений распределяются состояния x_l и x_h, соседние с x_i и x_j, тоже начиная с состояний, имеющих наибольшее число переходов. Так процесс повторяется до тех пор, пока в каждое разбиение не войдут все состояния автомата. При этом на каждом шаге необходимо проверять, чтобы произведение получаемых разбиений было нулевым разбиением.

Если заданный граф переходов нельзя закодировать соседним кодом, некоторые соседние состояния придется помещать в разные блоки более чем одного разбиения. Оказывается более выгодным, чтобы эти соседние состояния, которым будут приписаны несоседние кодовые комбинации, были общими одновременно для нескольких контуров нечетной длины и имели наименьшее число связей.

С учетом этих рекомендаций выберем код для внутренних состояний МА, переходы между которыми заданы следующей матрицей (табл. 5 7):

Таблица 5.7

	1	2	3	4	5	6
1		2		1	1	
2	1		1			
3				1		
4		1	1			
5						1
6						

Составим таблицу соседних состояний с указанием числа переходов между ними:

Пары соседних состояний	1,2	1,4	1,5	2,3	2,4	3,4	5,6
Число связей	3	1	1	1	1	2	1

Сумма всех элементов матрицы равна 10, значит, соседний код имел бы вес, равный 10. Однако из таблицы соседних состояний видно, что в графе переходов имеются контуры нечетной длины 1, 2, 4 и 2, 3, 4. Состояниям хотя бы одной пары, входящей в такой контур, должны быть приписаны несоседние кодовые комбинации. Поскольку состояния 2 и 3 входят в оба указанных контура, желательно именно этим состояниям приписать несоседние комбинации, т. е. поместить эти состояния в разные блоки не одного, а двух разбиений. Учитывая это, а также число переходов между соседними состояниями, выберем код, постепенно распределяя состояния по блокам разбиений. Процесс выбора кода легко проследить по табл. 5.8.

Таблица 5.8

№ шага	Разбиения	Первый блок разбиения	Второй блок разбиения	Вес разбиения
1	π_1	1, 2		0
	π_2	1, 2		0
	π_3	1	2	3
2	π_1	1, 2, 4		0
	π_2	1, 2	4	2
	π_3	1, 4	2	4
3	π_1	1, 2, 3, 4		0
	π_2	1, 2	3, 4	3
	π_3	1, 4	2, 3	6
4	π_1	1, 2, 3, 4	5	1
	π_2	1, 2, 5	3, 4	3
	π_3	1, 4, 5	2, 3	6
5	π_1	1, 2, 3, 4	5, 6	1
	π_2	1, 2, 5, 6	4, 3	3
	π_3	1, 4, 5	2, 3, 6	7

Из последней таблицы видно, что на некоторых шагах процесс выбора разбиений не является однозначным. Например, состояние 6 можно было бы поместить во второй блок не π_3, а π_2, однако в данном случае это не повлияло бы на вес кода.

В результате получили код, имеющий вес 11I и находящийся на расстоянии $t=1$ от соседнего кода, который имеет вес, равный 10. Таким образом, полечен код, имеющий минимально возможный для данного графа вес.

Однако в общем случае можно не получить код с минимальным весом, так как возникающие на каждом шаге варианты распределения состояний по блокам разбиений могут приводить не только к перераспределению весов между разбиениями, но и к изменению веса получаемого кода. В этом случае просмотр нескольких вариантов может позволить нам приблизиться к коду с минимальным весом, определяемым по матрице μ.

Используя введенные понятия «вес разбиения» и «вес кода», можно поставить и задачу минимизации числа логических элементов в матрице M_3 с учетом ограничений на число входов в эти элементы.

До сих пор говорили о минимизации общего числа диодов или транзисторов в M_3 , т. е. элементов, на которых строятся логические элементы ИЛИ. Если число входов в элемент ИЛИ ограничено, то код с минимальным весом может не обеспечить минимального числа этих элементов. При этом нужно будет стремиться к получению таких двухблочных разбиений, вес каждого из которых не превосходит возможного числа входов в элемент ИЛИ для элементов памяти с одним входом. Для элементов памяти с раздельными входами вес разбиения нужно будет характеризовать двумя числами, каждое из которых определит число переходов в соответствующий блок разбиения.

Рассматривая задачу кодирования внутренних состояний МА с целью минимизации M_3, считали, что число и структура микрокоманд уже определены. При этом может оказаться, что полученный в процессе выбора микрокоманд граф переходов не может быть закодирован соседним кодом. Однако это еще не означает, что для данного автомата невозможно соседнее кодирование, так как от выбора структуры микрокоманд существенно зависит вид графа переходов. Именно поэтому оптимальные результаты при синтезе микропрограммных автоматов может дать совместное решение задач формирования микрокоманд и кодирования внутренних состояний. Это особенно важно, когда используется не схемная реализация управляющего алгоритма, а осуществляется его размещение в ПЗУ. При этом возникает целый ряд новых задач, относящихся к области микропрограммирования.

Итак рассмотрели основные этапы синтеза автономного микропрограммного автомата, условия работы которого задаются одной ЛСА или ПЛСА. Если автомат является неавтономным, то дополнительно должна быть решена задача оптимального кодирования

его состояний входа. Следует отметить также, что когда условия работы МА задаются набором параллельных алгоритмов, возникает ряд новых задач не только на этапе объединения частных ПЛСА, но и затем на этапе кодирования состояний, особенно если при синтезе учитываются свойства элементного базиса .

5.5. Блочный синтез управляющих автоматов

5.5.1. Общие положения

Создание систем управления различными технологическими процессами связано с решением ряда сложных задач, в связи с чем их проектирование обычно разбивают на отдельные этапы, на каждом из которых решается достаточно самостоятельная задача по синтезу системы управления.

Один из наиболее сложных этапов проектирования таких систем — этап блочного (системного) синтеза, когда на основе общего алгоритмического описания системы управления и взаимодействия последней с объектом управления (ОУ) определяется блочная структура системы управления, т. е. состав основных блоков и принцип их реализации.

Этап блочного синтеза структуры системы управления имеет большое значение в процессе проектирования, так как от правильного выбора структуры зависят надежностные, массо-габаритные, стоимостные и другие показатели проектируемой системы управления. При этом процесс выбора структуры системы управления еще более усложняется тем, что, как правило, в сложных системах выполнение части операций по управлению ОУ по тем или иным причинам возлагается па человека. В связи с этим выбор оптимальной структуры такой автоматизированной системы управления связан с решением многовариантной задачи, в которой оценочный функционал определяет зависимость выбираемого решения от многих параметров. В настоящее время для решения многовариантных задач управления используются методы теории решений, которые позволяют обеспечить принятие наилучшего решения с учетом всей априорной информации об условиях поставленной задачи. При этом под понятием наилучшего решения имеется в виду выбор определенного решения из некоторого множества альтернативных решений. В 5.5.2 излагается декомпозиционный метод блочного синтеза УА на основе алгоритмического автомата. Ряд других методов блочного синтеза иерархических и распределенных мпогомикропроцессорных систем достаточно подробно изложен в литературе.

Основой для выбора блочной структуры автоматизированной системы управления является алгоритм управления.
В связи с тем что в сложных УА каждая функция выходов f_{A_i} зависит от небольшого числа входных переменных при достаточно большом общем числе последних, алгоритм функционирования УА целесообразно разбить на отдельные, так называемые частные алгоритмы функционирования $\mathfrak{A}_1, \ldots, \mathfrak{A}_L$ и весь алгоритм функционирования УА представить в виде композиции таких частных алгоритмов функционирования: $\mathfrak{A}_1, \ldots, \mathfrak{A}_L$, т. е. $\mathfrak{A}=\mathfrak{A}(\mathfrak{A}_1, \ldots, \mathfrak{A}_L)$.
Каждый частный алгоритм функционирования (частный управляющий алгоритм) определяет последовательность воздействий на те исполнительные механизмы ОУ, которые образуют подмножество исполнительных механизмов, функции возбуждения которых зависят от одних и тех же (или почти одних и тех же) входных переменных. Таким образом, каждый частный управляющий алгоритм определяет совокупность операций, которые функционально связаны между собой значительно сильнее, чем операции, принадлежащие различным частным алгоритмам. В связи с этим при выборе структуры сложного иерархического УА целесообразно все операции, принадлежащие к одному и тому же частному управляющему алгоритму, реализовать одинаковым образом, включив их в один и тот же блок УА. В этом случае выбор оптимальной структуры УА будет состоять в выборе оптимальной реализации каждого такого блока и в определении целесообразности объединения различных блоков в один общий блок. При этом для упрощения процесса выбора структуры УА целесообразно иметь по возможности наименьшее число блоков, реализующих частные алгоритмы функционирования, что требует минимизации числа частных алгоритмов.

5.5.2. Метод декомпозиции алгоритмического автомата

Процесс построения декомпозиционной структуры алгоритмического автомата в виде совокупности блоков (Б), каждый из которых удовлетворяет требованиям по сложности своем реализации или по какому-либо другому параметру, рассмотрим на следующем примере. При этом будем считать, что разбиение УА на блоки необходимо осуществить так, чтобы в один блок вошли процедуры, имеющие наибольшее тяготение (например, использующие в процессе выполнения одни и те же параметры, взаимосвязанные по обработке данных и т. п.).
Пусть в УА реализуются восемь процедур $\pi_1, \ldots, \pi_8$. Их

взаимосвязь зададим в виде треугольной таблицы (табл. 5.9), а сложность реализации каждой из них представим в виде условных единиц:

$$C_{\pi_1} = 13,\ C_{\pi_2} = 13,\ C_{\pi_3} = 18,\ C_{\pi_4} = 9,$$

$$C_{\pi_5} = 14,\ C_{\pi_6} = 19,\ C_{\pi_7} = 9,\ C_{\pi_8} = 15.$$

Таблица 5.9

	π_1	π_2	π_3	π_4	π_5	π_6	π_7
π_2	5						
π_3	5	5					
π_4	—	1	—				
π_5	—	—	2	5			
π_6	2	—	5	1	5		
π_7	1	—	—	2	—	1	
π_8	—	2	1	—	2	5	5

При этом примем допустимую сложность $C^{\text{д}}_{b_i}$ реализации каждым i-м блоком $Б_i$ равную 70, т. е. $C^{\text{д}}_{Б} \leqslant 70$.

Две процедуры π_i и π_j назовем ε-связными, если тяготение между ними $y_{\pi_{ij}} \sim \varepsilon$. Группу процедур назовем ε-связной, если каждая пара процедур, вошедших в данную группу, является ε-связной. Допустимой назовем ε-связную группу $G^{\varepsilon}(m)$, включающую m процедур, если $\sum_{i=1}^{m} C_{\pi_i} \leqslant C^{\text{д}}_{Б}$, $\pi_i \in G^{*}(m)$.

Допустимую ε-связную группу назовем максимальной, если включение в эту группу любой процедуры из заданного множества процедур, выполняемых блоком, приводит к том, что она становится недопустимой ε-связной группой. Для получения максимальных допустимых ε-связных групп воспользуемся методом построения минимального покрытия максимальных ε-связных подмножеств частных алгоритмов, аналогичным методу минимизации числа внутренних состояний автомата. В результате при ε=1 получим

следующие допустимые максимальные ε-связные группы процедур. Для простоты вместо π_i в группах указывается только номер i:

$a = \overline{1, 2, 3}$ при $C_a = C_{\pi_1} + C_{\pi_2} + C_{\pi_3} = 13 + 13 + 18 = 44 < 70$;

$b = \overline{1, 6, 7}$ при $C_b = C_{\pi_1} + C_{\pi_6} + C_{\pi_7} = 13 + 19 + 19 = 41 < 70$;

$c = \overline{2, 4}$ при $C_c = C_{\pi_2} + C_{\pi_4} = 13 + 9 = 22 < 70$;

$d = \overline{2, 3, 8}$ при $C_d = C_{\pi_2} + C_{\pi_3} + C_{\pi_8} = 13 + 18 + 15 = 46 < 70$;

$e = \overline{3, 5, 6, 8}$ при $C_e = C_{\pi_3} + C_{\pi_5} + C_{\pi_6} + C_{\pi_8} = 18 + 14 + 19 + 15 = 66 < 70$;

$f = \overline{4, 5, 6}$ при $C_f = C_{\pi_4} + C_{\pi_5} + C_{\pi_6} = 9 + 14 + 19 = 42 < 70$;

$g = \overline{4, 6, 7}$ при $C_g = C_{\pi_4} + C_{\pi_6} + C_{\pi_7} = 9 + 19 + 9 = 37 < 70$;

$h = \overline{5, 7, 8}$ при $C_h = C_{\pi_5} + C_{\pi_7} + C_{\pi_8} = 19 + 9 + 15 = 43 < 70$;

$\eta = \overline{1, 3, 6}$ при $C_\eta = C_{\pi_1} + C_{\pi_3} + C_{\pi_6} = 13 + 18 + 19 = 50 < 70$.

Составим таблицу покрытий (табл. 5.10).

Таблица 5.10

	$\overline{1.2}$	$\overline{1.3}$	$\overline{1.6}$	$\overline{1.7}$	$\overline{2.3}$	$\overline{2.4}$	$\overline{2.8}$	$\overline{3.5}$	$\overline{3.6}$	$\overline{3.8}$	$\overline{4.5}$	$\overline{4.6}$	$\overline{4.7}$	$\overline{5.6}$	$\overline{5.8}$	$\overline{6.7}$	$\overline{6.8}$	$\overline{7.8}$
a	(V)	V			V													
b			V	(V)												V		
c						(V)												
d					V		(V)			V								
e								(V)	V	V				V	(V)		V	
f											(V)	V		V				
g												V	(V)			V		
h																V	V	(V)
η		V	V						V									

Из этой таблицы видно, что минимальное множество допустимых максимальных ε-связных групп включает группы a, b, c, d, e, f, g, h; группа η—избыточная. Легко понять, что они

являются пересекающимися, т. е. одна и та же процедура входит в несколько групп, но при этом связи между группами отсутствуют. Для того чтобы устранить такое пересечение, оставим каждую процедуру только в одной из групп, исключив ее из остальных так, чтобы возникающая при этом связь между группами была бы минимальной. Рассмотрим процедуру π_1, которая входит в две группы *(a* и *b).* Если процедуру π_1 удалить из группы *a,* то возникнет тяготение между группой *a* и *b:*

$$y^a_{a,b} = y_{1,2} + y_{1,3} = 5 + 5 = 10.$$

Если же процедуру π_1 удалить из группы *b,* то

$$y^b_{a,b} = y_{1,6} + y_{1,7} = 2 + 1 = 3.$$

Следовательно, процедуру π_1 целесообразнее удалить из группы *b,* так как при этом взаимосвязь между группами *a* и *b* будет меньше, чем при удалении процедуры π_1 из группы *a.* Возможность удаления процедуры π_1 из группы *b* запишем в виде $b = \widetilde{1, 6, 7}$. Аналогичным обэазом осуществим исключение пересечении из других групп. При этом получим $a = \overline{1, 2, 3}$; $b = \widetilde{1, 6, 7}$; $c = \widetilde{2, 4}$; $d = \widetilde{2, 3, 8}$; $e = \widetilde{3, 5, 6, 8}$; $f = \overline{4, 5}, \widetilde{6}$; $g = \widetilde{4, 6, 7}$; $h = \widetilde{6, 7}, \overline{8}$.

Для выбора минимального числа непересекающихся допустимых ε-связных групп процедур построим таблицу покрытий вида, приведенного в табл.5.11.

Таблица 5.11

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8
a	∨	∨	∨					
b	~					~	~	
c		~		~				
d		~	~					~
e			~		~	∨		
f				∨	∨	~		
g				~		~	~	
h						~	∨	∨

В табл. 5.11 знаком ~ показано, что процедура, сопоставленная со столбцом, может быть исключена из соответствующей группы.

В связи с тем что в группе *e* осталась лишь одна процедура π_6, она может быть включена или в группу *f*, или в группу *h*. Тогда, очевидно, в минимальное множество допустимых ε-связных групп войдут группы *a, f, h*. Сопоставив с каждой из этих групп один блок, получим три блока УА: $Б_a$, $Б_f$ и $Б_h$. При этом в зависимости от включения процедуры π_6 в группу *f* или в группу *h*, получим два варианта блоков $Б_f$ и $Б_h$.

В том случае, когда $C_{E_i} \leqslant C^{\text{п}}_{Б_i}$, можно осуществить объединение полученных блоков так, чтобы $C_{F_i}, \quad , i = \sum_{i=1}^{l} C_{E_i} \leqslant C^{\text{п}}_{Б_i}$.

В нашем случае с учетом исключенных процедур имеем

$C_a=44,\ C^1_f=42,\ C^1_h=28$ при $\pi_6 \in f$;

$C_a=44,\ C^2_f=23,\ C^2_h=43$ при $\pi_6 \in h$.

При $\pi_6 \in f$ можно объединить в один блок $Б_{f,h}$ группы *f* и *h*, так как $C_{f,h}=C_f+C_h=42+28=70=C^{\text{п}}_{Б}$. Тогда будем иметь два блока: $Б_a$, включающего процедуры π_1, π_2, π_3, и $Б_{f,h}$, включающего остальные процедуры π_4, π_5, π_6, π_7 и π_8.

При $\pi_6 \in h$ блок $Б_f$ можно объединить или с $Б_a$, или с $Б_h$, т. е. получим следующие два решения: 1) $Б_{a,f}$ и $Б_h$; 2) $Б_a$ и $Б_{f,h}$.

6. Метод логического синтеза автоматов на программируемых матрицах

На каждом этапе развития элементной базы и средств проектирования требуются новые постановки и новые методы решения задач логического синтеза, с которыми сталкиваются разработчики систем автоматики и вычислительной техники.

Математическую основу логического синтеза составляет прикладная теория функций алгебры логики (ее также называют теорией переключательных схем). Эта теория послужила основой для создания нескольких поколений методов логического синтеза, соответствующих поколениям синтезируемых объектов: 1) релейно-контактпых схем, 2) схем из функциональных элементов, 3) схем средней степени интеграции, 4) схем большой и сверхбольшой степени интеграции. Каждому поколению соответствуют свои постановки, свои размерности задач, свои критерии качества решений и свои вычислительные средства для решения.

В развитии методов логического синтеза четко прослеживаются две тенденции, которые условно назовем интеграцией и автоматизацией. В

первом поколении каждой элементарной операции алгебры логики соответствовал элемент схемы; задачи синтеза ставились так, чтобы выразить заданные функции в наиболее простом виде — наименьшим числом операций и вхождений переменных При этом в большинстве методов использовалась классическая постановка — поиск минимальной дизъюнктивной нормальной формы (ДНФ). Во втором и третьем поколениях заданная функция отображается композицией из стандартного набора более простых, составляющих базис схем малой и средней степени интеграции. Затем интеграция достигает такого уровня, когда физически неделимый элемент рассматривается как элемент структурного синтеза устройства, а объектом логического синтеза является структура большой интегральной схемы (БИС) Тенденции большой интеграции и автоматизации имеют определяющее значение. Первая формирует новый взгляд на результат синтеза — схему, вторая— на способ ее реализации. Прежде всего, большая интеграция изменяет критерии качества синтезируемых схем. Лучшей считается схема, занимающая меньшую площадь кристалла БИС. Этот критерий в сочетании с технологическим требованием регулярности структуры естественно побудил отображать на плоскость кристалла ДНФ функции или системы функций в виде матричной схемы, в которой столбцам соответствуют переменные, а строкам — элементарные конъюнкции или наоборот. Такие схемы, названные программируемыми логическими матрицами (ПЛМ), получили широкое распространение.

Потребность в методах синтеза ПЛМ вызвала волну интереса к преобразованиям функций в базисе ДНФ. Новые постановки задач отличаются в основном ориентацией на автоматизацию вычислений. Термин «методы синтеза» вытесняется термином «алгоритмы синтеза». Однако быстро растущую трудоемкость решения задач не удается «победить» с помощью ЭВМ. Приходится отказаться от поиска точных решений. Более того, часто невозможно или нецелесообразно затрачивать много усилий на поиск «хороших» приближений к точному решению. Отсюда возникает еще одно требование к методам логического синтеза — **алгоритмическая гибкость**. Это требование означает, что методы должны допускать построение алгоритмов, как простых для получения грубых решений, так и сложных для хороших приближений к точному решению.

Все перечисленные требования к методам синтеза ПЛМ были осознаны и сформулированы к началу 80-х годов XXв. Было создано несколько комплексов алгоритмов. Каждый комплекс имеет свое назначение, использует свой набор эвристических приемов и свои языки программирования. Однако все они основаны на одном и том же

математическом аппарате: **преобразование ДНФ логических функций достигается путем применения операций «склеивания» пар интервалов, на которых значения функций отличны от нуля.** Поиск склеиваемых интервалов осуществляется путем перебора, который с помощью **эвристических приемов** стараются сделать более направленным. Выработалось мнение, что такой подход единственный, что он исчерпывает возможности математического аппарата и поэтому усилия следует направлять на **поиск полезных эвристик, создание подходящих языков программирования и удобного сервиса**. Такое мнение вполне справедливо относительно необходимости разработки хорошего программного обеспечения, но не возможностей математического аппарата. Действительно, преобразования логических функций в известных алгоритмах производятся в булевом пространстве. На самом же деле как функции, так и все преобразования определены на множестве интервалов булева пространства. Свойства этого множества слабо изучены и мало используются. Однако известно, что чем глубже исследована область определения функции, чем больше на ней «порядка», тем больше шансов найти способы сокращения перебора. **Поэтому и стремяться упорядочить множество интервалов, исследовать его свойства и использовать их в алгоритмах преобразования функций.** В этом существо рассматриваемого ниже подхода к разработке алгоритмов синтеза ПЛМ.

Упорядочение множества интервалов достигается с помощью **теории разбиений,** основы которой изложены в книге Хартманиса и Стирнза . Понятие множества разбиений с определенными на нем отношениями порядка и соответствующими этим отношениям операциями используется для представления множества всех интервалов булева пространства. **При этом каждый интервал соответствует блоку (подмножеству) разбиения множества двоичных наборов.** Множество таких разбиений образует **алгебраическую структуру (решетку), обладающую свойствами булевой алгебры и изоморфную исходному множеству двоичных наборов**. Так осуществляется переход от множества двоичных наборов к множеству его равноблочных разбиений. В этой новой области определения преобразуемых функций отношения между импликантами выражаются явно, что естественно облегчает выбор тех из них, которые лучшим образом удовлетворяют критериям поставленной задачи. Отсюда простота и однотипность процедур, а также возможность сокращения перебора.

Более того, богатство свойств структуры и строгая симметричность отношений в ней позволили найти способ улучшения емкостной

сложности алгоритмов — введение так называемых **компактных представлений структуры**. **Структура множества разбиений представляется упорядоченным перечислением элементов множества, на котором строятся разбиения (множества двоичных наборов или состояний).** При этом любой блок любого разбиения извлекается из этого представления с помощью простых операций. **Компактные представления являются главным результатом в изложенной теории**.

Основные процедуры алгоритмов синтеза ПЛМ сведены к построению и анализу компактных предсгавлений структуры. При этом в алгоритмах приближенной минимизации при поиске максимальных импликант сокращение перебора достигается при использовании отношения порядка между разбиениями усеченной структуры — структуры, в которой отсутствуют блоки разбиений, пересекающиеся с областью нулевых значений исходных функций. В алгоритмах кодирования перебор вообще практически отсутствует - они состоят из простых операций упорядочения и переупорядочения множества состояний. Правила упорядочения обеспечивают получение структуры множества разбиений, удовлетворяющего заданному критерию — **либо наименьшему числу импликант в реализуемых функциях (экономичное кодирование), либо отсутствию гонок при переходах из состояния в состояние (противогоночное кодирование).**

Интересно проявляется дуальность алгоритмов решения этих двух задач. **Построение структуры осуществляется по алгоритму экономичного кодирования снизу вверх — от метких разбиений (на пары состояний) к крупным (разбиения на два блока), а по алгоритму противогоночного кодирования сверху вниз.** Такая стройность построений при решении разных по критериям задач, вообще говоря, не свойственная алгоритмам комбинаторного типа, свидетельствует о соответствии математического аппарата решаемым задачам. Приведенные алгоритмы, безусловно, не исчерпывают возможностей предлагаемого подхода, они только служат хорошей иллюстрацией его полезности

6.1. Задачи синтеза логических преобразователей на программируемых логических матрицах

6.1.1. Программируемые логические матрицы в цифровых устройствах

Тезис о том, что наиболее целесообразной организацией микроструктуры ЭВМ на базе больших интегральных схем (СБИС) является сеть из одинаковых автоматов, одинаковым образом соединенных со своими ближайшими соседями и имеющих программируемые функции, был выдвинут в 1962 г. Такие сети были названы однородными вычислительными средами и наделены конструктивной и технологической однородностью, универсальностью (в смысле выполнения любого алгоритма), а также возможностью наращивания и параллельного выполнения сразу нескольких алгоритмов.

Развитие микроэлектроники и вычислительной техники подтвердило тезис об однородных вычислительных средах. Действительно, основным принципом, позволяющим проектировать сложные устройства на основе БИС быстро и дешево, является принцип регулярности. Развитие этого принципа привело к унификации изделий в производстве, за исключением одной-днух последних операций. В результате возникли схемы-полуфабрикаты, имеющие однородную структуру, чаще всего матричную. Функции таких схем программируются с помощью последних технологических операций. Матричный подход к организации структуры БИС позволяег сочетать массовость производства с возможностью реализации произвольных логических функций.

На современном уровне развития микроэлектроники и вычислительной техники теоретическая концепция однородных вычислительных сред воплотилась в реальных объектах: сетях из микропроцессоров, параллельных микропрограммных структурах, вентильных, систолических, программируемых логических матрицах, матрицах макроэлементов.

Один из основных разделов теории однородных вычислительных сред составляют работы по синтезу автоматов в средах. Особенности вычислительных сред, а именно их матричная структура, изменили критерий сложности реализации автомата и постановки задач синтеза. Сложность реализации автомата определяется площадью участка среды, на котором размещен автомат, а не числом элементов логической сети. Возникли методы синтеза автоматов, позволяющие от

любых форм задания автомата перейти к матрице, являющейся программой настройки среды. Эти методы оказались применимыми без каких-либо изменений к программируемым логическим матрицам.
Программируемая логическая матрица представляет собой однородную сетку шин М, разделенную на две части: M_1 и M_2 (рис. 1).

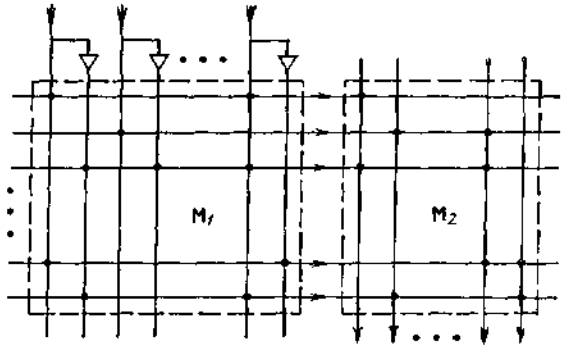


Рис. 1

Каждая вертикальная шина в $\mathbf{M}_1$ соответствует входному сигналу. Горизонтальная шина в **М** формирует логическое произведение от входов. Каждая вертикальная шина в $\mathbf{M}_2$ формирует логическую сумму от логических произведений, результат суммирования является выходом. Таким образом, ПЛМ предназначена для реализации системы булевых функций, выраженных в ДНФ. При этом число входов ПЛМ соответствует числу переменных, число строк — числу конъюнкций в ДНФ системы, число выходов — числу функций в системе. Как и среды, ПЛМ обладают конструктивной и технологической однородностью, возможностью наращивания, универсальностью (в смысле реализации любой логической функции). Основным общим критерием сложности реализации автоматов в среде и ПЛМ, с учетом которого разрабатываются методы синтеза, является минимум площади среды или кристалла БИС.

В узлах сетки шин находятся полупроводниковые элементы — диоды и транзисторы, которые могут быть включены в электрическую схему или выключены из нее на этапе программирования ПЛМ. Выключение элементов производится путем разрушения соответствующих связей. ПЛМ является универсальным видом БИС, на основе которого реализуются как управляющие, так и арифметическо-логические устройства. При этом ПЛМ могут выполнять самые разнообразные функции: памяти микропрограмм, преобразования кодов, поиска, ре-

шающих таблиц и т. п. ПЛМ может занимать кристалл или разделять его с регистрами, вентилями и другими схемами.
Часть входов и выходов могут быть внутренними и замыкаться через триггеры цепью обратной связи.
Опишем несколько конкретных примеров применения ПЛМ и способов организации устройств на их основе.
Например, в машине, выполняющей функции управления терминалом и режимом передачи сообщений, имеющей двустороннюю связь с центральным процессором, 86% всех логических схем реализовано на 7 ПЛМ с параметрами: 31 вход, 29 выходов, 70 строк, в том числе 13 выходов и 13 входов внутренние. Семь таких ПЛМ заменили 1731 логическую схему малой и средней степени интеграции.
Многоразрядный сумматор реализован в виде каскада ПЛМ. Входные коды разбиваются на группы по n разрядов. Для каждой группы используется свой сумматор — одна ПЛМ, имеющая 17 входов, 9 выходов, 73 строки. Для младшей группы разрядов применяется обычный сумматор, для остальных — сумматоры, организующие две суммы: одну для случая, когда перенос в данную группу разрядов равен единице, другую для случая, когда перенос равен нулю. Нужная сумма выбирается с помощью сигнала переноса. Если число групп разрядов больше двух, то одна дополнительная ПЛМ используется для организации сквозного переноса. Таким образом, для N -разрядного сумматора требуется $N/n+1$ ПЛМ, если $N/n>2$.
ПЛМ широко используются для наведения «мостов» между крупными функциональными блоками, реализуемыми в виде отдельных БИС. Такие «мосты» необходимы для получения законченных конструкций и представляют собой нетривиальные блоки произвольной логики. Для реализации таких блоков на основе ПЛМ организуется как комбинационная, так и последовательностная логика. Для организации комбинационной логики кроме двухуровневой ПЛМ используется упрощенный вариант — программируемая матричная логическая схема, представляющая собой первый уровень ПЛМ, т. е. матрицу $\mathbf{M}_1$.
Часто используется конкретный вид последовательностной схемы на основе ПЛМ — логический контроллер последовательности, представляющий собой автономный автомат, предназначенный для реализации логической последовательности как ряда переходов из одного состояния в другое.
Архитектура самой ПЛМ может быть более гибкой, чем ПЛМ, представленной на рис. 1. Гибкость достигается за счет разрезания шин (линий металлизации). Разрезы в вертикальных шинах матриц $\mathbf{M}_1$ и $\mathbf{M}_2$ позволяют подавать переменные и выводить информацию с двух сторон. Делают разрезы и в горизонтальных шинах ПЛМ, при этом

матрицу $\mathbf{M}_1$ разделяют на две части и между ними помещают матрицу $\mathbf{M}_2$ (рис. 2).

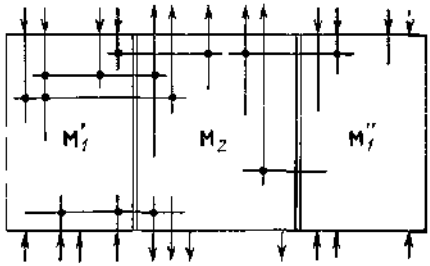


Рис. 2

Разрезание позволяет разделить оборудование на модули, повысить плотность записи информации в ПЛМ, более экономно и целесообразно использовать площадь кристалла. Места разрезов шин программируются пользователем.

Для увеличения гибкости структуры и расширения функциональных возможностей ПЛМ предлагается чередовать входные и выходные шины матрицы, между вертикальными шипами располагать ряды триггеров. Такая матрица путем разрезания шин может быть разбита на независимые подматрицы с отдельным доступом для выполнения независимых заданий.

6.1.2. Базовые понятия

Алгоритмы синтеза ПЛМ, описанные ниже, построены на основе теории конечных автоматов

Конечный автомат — это набор из пяти элементов $\{\Sigma, \Gamma, S, \delta, \lambda\}$, где $\Sigma = \{\sigma_1, \ldots \sigma_r\}$ — конечное множество входных состояний; $\Gamma=\{\gamma_1, \gamma_w\}$— конечное множество выходных состояний; $S = \{s_1, \ldots, s_v\}$— конечное множество внутренних состояний; $\delta:\Sigma\times S\rightarrow S$ — функция перехода; $\lambda:\Sigma\times S \rightarrow\Gamma$ — функция выхода. Входные и выходные состояния связывают автомат с внешней ородой

Автомат работает в дискретном времени. Опишем кратко работу автомата. Пусть в момент времени t автома имел внутреннее состояние s_i и входное σ_j. Функции перехода δ определяет состояние s_k автомата в моммент t+1 в ответ на полное состояние $\sigma_j s_i$. Функция выхода λ ставит выход γ_i в соответствие полному состоянию $\sigma_j s_i$ (в этом случае автомат называется автоматом Мили) или внутреннему состоянию s_i (в этом случае автомат называется автоматом Мура). Внутренние

состояния автомата позволяют определить состояние выхода в момент t в зависимости от состояния входа не только в момент t, но и в предшествующие моменты, т. е. в зависимости от предыстории работы автомата.

Для описания работы автомата чаще всего используют таблицы и графы переходов. Функции перехода и выхода задаются двумя таблицами (переходов и выходов) для автомата Мили $\mathbf{A}_1$

$$\begin{array}{c|ccc} & \sigma_1 & \sigma_2 & \sigma_3 \\ s_1 & s_1 & s_3 & s_2 \\ s_2 & s_2 & — & s_3 \\ s_3 & s_2 & s_1 & — \end{array}, \qquad \begin{array}{c|ccc} & \sigma_1 & \sigma_2 & \sigma_3 \\ s_1 & \gamma_1 & \gamma_3 & \gamma_3 \\ s_2 & \gamma_2 & — & \gamma_2 \\ s_3 & \gamma_2 & \gamma_1 & — \end{array}$$

И одной для автомата Мура $\mathbf{A}_2$

$$\begin{array}{c|ccc|c} & \sigma_1 & \sigma_2 & \sigma_3 & \\ s_1 & s_1 & s_3 & s_2 & \gamma_1 \\ s_2 & s_2 & — & s_3 & \gamma_2 \\ s_3 & s_2 & s_1 & — & \gamma_3 \end{array}.$$

Графы переходов автоматов $\mathbf{A}_1$ и $\mathbf{A}_2$ представлены на рис. 3, а и б соответственно.

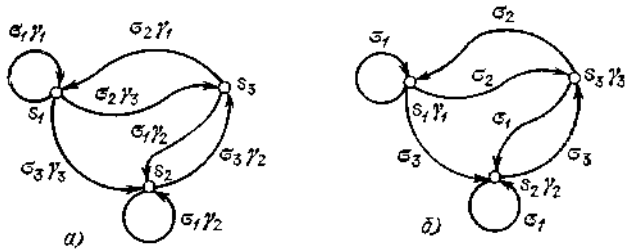


Рис. 3

Частным случаем автомата является *автомат без памяти,* выходное состояние которого зависит только от входного. На этапе построения логической схемы автомат принято считать состоящим из двух частей: логического преобразователя и памяти. Очевидно, что *логический преобразователь* — это автомат без памяти.

Функции логического преобразователя (функции перехода и выхода автомата) представляются в виде *булевых функций* $f(x_0, \ldots, x_{n-1})$, при этом все состояния (входные, выходные, внутренние) кодируются двоичными кодами. Булева функция задается на множестве $f^{(n)}$ двоичных наборов $(x_0, \ldots, x_{n-1})$. Общее число всех двоичных наборов длины n равно 2^n.

В общем случае на каждом двоичном наборе булева функция может принимать значения 0 и 1 или неопределенное значение $(*)$—либо 0, либо 1. Если функция на некоторых наборах имеет неопределенное

значение, то она называется не полностью определенной или частичной. Для задания полностью определенной булевой функции достаточно перечислить все двоичные наборы, на которых она равна единице. Множество таких наборов обозначим I^1. Множество наборов, на которых функция равна нулю, I^0 определяется как дополняющее I^1 до $I^{(n)}$, т. е. $I^0 = I^{(n)} \setminus I^1$. Если функция имеет неопределенные значения, то она задается двумя множествами I^1 и I^0, а множество наборов, на которых функция имеет неопределенное значение, I^* определяется как $I^{(n)} \setminus (I^1 \cup I^0)$.

Булева функция может быть задана таблицей своих значений на множестве наборов $I^{(n)}$. Например, функция $f_1(x_0, x_1, x_2)$ задана следующей таблицей:

x_0	x_1	x_2	f_1	x_0	x_1	x_2	f_1
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	1
0	1	1	0	1	1	1	0

Аналитически задать булеву функцию можно с помощью элементарных булевых функций. Среди них наиболее часто используют следующие: отрицание (НЕ) $f = \overline{x}$, конъюнкцию (И) $f = x_1 \& x_2$ ($f = 1$, если $x_1 = x_2 = 1$, $f = 0$ при всех других комбинациях значений x_1 и x_2, знак & иногда заменяется точкой или знаком $\wedge$, часто опускается), дизъюнкцию (ИЛИ) $f = x_1 \vee x_2$ ($f = 0$, если $x_1 = x_2 = 0$, $f = 1$ при всех других комбинациях значений x_1 и x_2). (Конъюнкцию и дизъюнкцию иногда называют логическими умножением и сложением соответственно).

Системы функций И-ИЛИ-НЕ, И-НЕ, ИЛИ-НЕ являются булевыми базисами. Это значит, что, используя одну из систем, можно аналитически выразить любую булеву функцию.

Наиболее часто для аналитического выражения булевой функции употребляется *дизъюнктивная нормальная форма* (ДНФ). Для ДНФ используется базис И-ИЛИ-НЕ. Конъюнкция множества попарно различных булевых переменных с отрицанием или без него называется элементарной, например $\overline{x}_0 x_1 \overline{x}_2 x_3$. Эта конъюнкция соответствует двоичному набору 0101. Дизъюнкция всех тех элементарных конъюнкций, которые соответствуют двоичным наборам из множества I^1 функции f, и есть ДНФ этой функции. Выражение $\overline{x}_0 \overline{x}_1 \overline{x}_2 \vee x_0 \overline{x}_1 \overline{x}_2 \vee$

$\vee x_0\bar{x}_1x_2$ представляет собой ДНФ функции f_1, таблица значений которой приведена выше. В общем случае булева функция представляется не единственной ДНФ. Так, функция f_1 может быть представлена еще ДНФ вида $x_0\bar{x}_2 \vee x_0\bar{x}_1$. Конъюнкция, содержащая не все переменные, соответствует троичному набору, например $x_0\bar{x}_2$ — набору $1*0$, $x_0\bar{x}_1$ —набору $10*$. Троичный набор $10*$ соответствует двум двоичным 100 и 101, набор $1*0$ — 100, 110. Это те самые двоичные наборы, на которых $f_1 = 1$. Отсюда следует, что обе приведенные выше ДНФ реализуют функцию f_1.

Среди всего множества ДНФ, реализующих функцию f, можно выбрать такую, которая содержит наименьшее общее число букв. Эту ДНФ называют *минимальной.* ДНФ функции f, содержащая наименьшее число конъюнкций по сравнению со всеми другими ДНФ, реализующими f, называется *кратчайшей.* Задача построения минимальной или кратчайшей ДНФ функции называется задачей минимизации f в классе дизъюнктивных нормальных форм.

Перейдем на язык **булевых интервалов**, дающий геометрическую интерпретацию понятиям, связанным с ДНФ булевой функции. Каждый двоичный набор из множества $f^{(n)}$ соответствует точке nмерного булева пространства. Наглядным представлением nмерного булева пространства является единичный nмерный куб, каждая вершина которого соответствует одному двоичному набору из $I^{(n)}$. Множество вершин куба, как и булево пространство, будем обозначать через $I^{(n)}$, множество вершин куба, на которых булева функция равна единице, — I^1, нулю — I^0, неопределенному значению— I^*.

Множество точек n-мерного булева пространств, соответствующее некоторой грани единичного n/-мерного куба, называется булевым интервалом. Например, множество {0001, 1001, 0101, 1101} есть булев интервал.

Существует более короткая запись булева интервала— троичный набор, i-й компонент которого имеет значение 0, если i-е компоненты всех двоичных наборов равны пулю, значение 1, если i-е компоненты всех двоичных наборов равны единице, неопределенное $(*)$, если i-е компоненты двоичных наборов различные. Так, интервал {0001,1001,0101,1101} представляется троичным набором $**01$.

Условимся троичные наборы обозначать через α_j. Если же интервал просто точка булева пространства (вершина n-мерного куба), то обозначим его a_j. В тех случаях, когда интервал как множество участвует в теоретико-множественных отношениях или операциях, будем обозначать его I_{α_j} или I_{a_j}. Для интервала существует понятие

ранга. **Ранг булева интервала** равен суммарному числу единиц и нулей в троичном наборе (или в частном случае — двоичном наборе). Так, ранг интервала $**01$ равен двум, ранг интервала a_j равен n. Интервалы ранга n в отличие от других булевых интервалов будем называть *n-интервалами.*

Булев интервал α_i, который принадлежит области I^1 функции $f: I_{\alpha_j} \subseteq I^1$, называется *допустимым* для этой функции. Очевидно, что допустимый интервал соответствует элементарной конъюнкции в ДНФ. ДНФ $f_1 = x_0\bar{x}_1\bar{x}_2 \vee x_0x_1\bar{x}_2 \vee x_0\bar{x}_1x_2$ соответствует множеству интервалов {100, 110, 101}, ДНФ $f_1 = x_0\bar{x}_2 \vee x_0\bar{x}_1$ —множеству интервалов $\{1* 0,\ 10*\}$. Для каждой ДНФ функции $f(x_0 ..., x_{n-1})$ выполняется соотношение $I^1 = \bigcup_j I_{\alpha_j}$, где $\{\alpha_j\}$ — множество интервалов, каждый из которых соответствует одной элементарной конъюнкции. Таким образом, каждой ДНФ функции f соответствует покрытие множества I^1 допустимыми интервалами. Покрытие, содержащее минимальное число интервалов, соответствует кратчайшей ДНФ и само называется кратчайшим. Обычно такое покрытие строится из максимальных интервалов. Допустимый интервал α_i называется максимальным для функции f, если не существует допустимого интервала α_j, такого, что $I_{\alpha_i} \subset I_{\alpha_j}$, т. е. внутри I^1 нет интервала α_j, объемлющего α_i.

Обобщим введенные понятия для частичных булевых функций. Интервал α_i называется допустимым для частичной функции f, если $I_{\alpha_i} \cap I^0 = \varnothing$ и $I_{\alpha_i} \cap I^1 \neq \varnothing$, т. е. интервал α_i принадлежит области $I^1 \cup I^*$ и содержит хотя бы один элемент из области I^1. Допустимый интервал α_i называется *максимальным* для функции f, если не существует допустимого интервала α_j-, такого, что $I_{\alpha_i} \subset I_{\alpha_j}$. ДНФ частичной функции f соответствует покрытию множества I^1 допустимыми интервалами. Кратчайшая ДНФ частичной функции f соответствует кратчайшему покрытию множества I^1 максимальными интервалами.

Система булевых функций $F(X) = \{f_0, \ldots, f_{m-1}\}, X = \{x_0, \ldots, x_{n-1}\}$, задается множеством $\{I^1_0, \ldots, I^1_{m-1}\}$ областей единичных значений функций $f_i \in F$, если все функции системы являются полностью определенными, и дополнительно множеством $\{I^0_0, \ldots, I^0_{m-1}\}$ областей нулевых значений функций $f_i \in$ F, если среди функций системы есть частичные. Система булевых функций также может быть представлена в виде ДНФ. При этом записываются все конъюнкции, соответствующие булевым интервалам, на каждом из которых хотя бы одна функция системы равна единице. Возле каждой

конъюнкции в скобках указываются те функции, которые равны единице на соответствующем интервале. Например, для системы функций $F_1 = \{f_0, f_1, f_2\}$, заданной таблицей

x_0	x_1	x_2	f_0	f_1	f_2	x_0	x_1	x_2	f_0	f_1	f_2
0	0	0	1	0	1	1	0	0	0	0	*
0	0	1	1	1	0	1	0	1	0	*	0
0	1	0	1	1	0	1	1	0	0	0	1
0	1	1	*	1	0	1	1	1	0	0	1

одна из ДНФ имеет вид $F_1 = \bar{x}_0\bar{x}_1\bar{x}_2(f_0, f_2) \vee \bar{x}_0\bar{x}_1x_2(f_0, f_1) \vee \vee \bar{x}_0x_1\bar{x}_2(f_0, f_1) \vee x_0\bar{x}_1x_2(f_1) \vee x_0x_1\bar{x}_2(f_2) \vee x_0x_1x_2(f_2)$.
ДНФ системы булевых функций соответствует покрытию множества областей $\{I^1_0, \ldots, I^1_{m-1}\}$ допустимыми для функций $f_0, \ldots, f_{m-1}$ интервалами. Кратчайшее покрытие соответствует кратчайшей ДНФ системы булевых функций. Так, для системы F_1 кратчайшая ДНФ имеет вид
$F_1 = \bar{x}_0\bar{x}_1\bar{x}_2(f_0, f_2) \vee \bar{x}_0x_2(f_0, f_1) \vee \bar{x}_0x_1(f_0, f_1) \vee \vee x_0x_1(f_2)$. Области $I^1_0 = \{000, 001, 010\}$, $I^1_1 = \{001, 010, 011\}$, $I^1_2 = \{000, 110, 111\}$ покрыты четырьмя допустимыми интервалами $000, 0*1, 01*, 11*$. Видно, что не все ишервалы покрытия являются максимальными для своих функций.

6.1.3. Матричная реализация системы булевых функций

Для разных технологий изготовления ПЛМ требуется представление систем булевых функций в разных базисах. Так, в ПЛМ, выполненной на МОП-транзисторах, система функций представляется в базисе ИЛИ-НЕ, в ПЛМ на биполярных элементах —в базисе И-ИЛИ-НЕ. Обозначим через $F(X) = \{f_0(X), \ldots, f_{m-1}(X)\}$ систему булевых функций, заданных на множестве переменных $X = \{x_0, \ldots, x_{n-1}\}$ и выраженных в ДНФ

$$f_j = \bigvee_{i=1}^{p_j} \varphi_i, \quad j = 0, \ldots, m-1.$$

Здесь φ_i — элементарная конъюнкция от переменных $\{x_0, \ldots, x_{n-1}\}$; p_j—число конъюнкций в ДНФ функции f_j. Обозначим $\Phi = \{\varphi_1, \ldots, \varphi_p\}$ множество элементарных конъюнкций, каждая из которых входит в ДНФ хотя бы одной из функций системы $F(X)$; $\tilde{X} = \{x_0, \bar{x}_0, \ldots, x_{n-1}, \bar{x}_{n-1}\}$ множество

всех переменных $x_i \in X$ и их отрицаний; $X(\varphi_i) = \{\tilde{x}_{i_1}, \ldots, \tilde{x}_{i_k}\}$ подмножество переменных $\tilde{x}_{i_j} \in X$, входящих в конъюнкцию φ_i ($\tilde{x}_{i_j}$ равен либо x_{i_j}, либо $\bar{x}_{i_j}$).

Введем в рассмотрение следующие матрицы:

$\mathbf{M}_1 = \|\mu_{ij}\|$ — матрица размера $p \times 2n$, элемент которой

$$\mu_{ij} = \begin{cases} 1, & \text{если } \tilde{x}_j \in X(\varphi_i), \\ 0 & \text{в противном случае;} \end{cases}$$

$\mathbf{M}_2 = \|\mu_{ij}\|$ — матрица размера $p \times m$, элемент которой

$$\mu_{ij} = \begin{cases} 1, & \text{если } \varphi_i \text{ входит в ДНФ функции } f_j, \\ 0 & \text{в противном случае,} \end{cases}$$

$\mathbf{X} = \|\tilde{x}_i\|$ — столбец из элементов $\tilde{x}_i \in \tilde{X}$;

$\mathbf{F} = \|f_j\|$ — строка из элементов $f_j \in F$.

Если для представления булевых функций используется базис ИЛИ-НЕ, система *F(X)* может быть записана в виде следующего матричного выражения:

$$\bar{\mathbf{F}} = \overline{(\mathbf{M}_1 \cdot \bar{\mathbf{X}})^{\mathrm{т}} \cdot \mathbf{M}_2} \quad \textbf{(1)}$$

Здесь умножение матриц выполняется по правилам матричного умножения с заменой арифметических операций одноименными логическими; знак «т» обозначает транспонирование матрицы; черта над матрицей обозначает поэлементное отрицание.

Выражение (1) является математической моделью двухуровневой матричной схемы, реализующей систему *F(X)*. При этом $\mathbf{M}_1$ соответствует первому уровню, реализующему множество конъюнкций Ф, а $\mathbf{M}_2$ — второму уровню, реализующему функции $f_j \in$ F как дизъюнкции некоторых $\varphi_i \in \Phi$.

Пример 1 Представим систему булевых функций

$$F(X) = \{f_0 = \bar{x}_0 x_1 x_2 \vee \bar{x}_0 \bar{x}_1 \vee x_0 x_1, \quad f_1 = x_0 x_1 \vee \bar{x}_0 \bar{x}_1 \bar{x}_2\}$$

ввиде матричного выражения. Множество конъюнкции, входящих в ДНФ функций f_0 и f_1:

$$\Phi = \{\varphi_1 = \bar{x}_0 x_1 x_2, \ \varphi_2 = \bar{x}_0 \bar{x}_1, \ \varphi_3 = x_0 x_1, \ \varphi_4 = \bar{x}_0 \bar{x}_1 \bar{x}_2\}.$$

Матрицы $\mathbf{M}_1$ и $\mathbf{M}_2$ имеют вид

$$\mathbf{M}_1 = \begin{matrix} x_0 & \bar{x}_0 & x_1 & \bar{x}_1 & x_2 & \bar{x}_2 \end{matrix}$$
$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{matrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \end{matrix};$$

$$\mathbf{M}_2 = \begin{matrix} f_0 & f_1 \end{matrix}$$
$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{matrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \end{matrix}.$$

Матричное выражение системы функций

$$\tilde{\mathbf{F}} = \overline{\left\{ \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \cdot \left[\begin{matrix} \bar{x}_0 \\ x_0 \\ \bar{x}_1 \\ x_1 \\ \bar{x}_2 \\ x_2 \end{matrix} \right]^T \right\}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} = \overline{\begin{bmatrix} \overline{x_0 \vee \bar{x}_1 \vee \bar{x}_2} \\ \overline{x_0 \vee x_1} \\ \overline{\bar{x}_0 \vee \bar{x}_1} \\ \overline{x_0 \vee x_1 \vee x_2} \end{bmatrix}^T} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} =$$

$$= \left[\overline{x_0 x_1 x_2 \vee \bar{x}_0 \bar{x}_1 \vee x_0 x_1} \;\middle|\; \overline{x_0 x_1 \vee \bar{x}_0 \bar{x}_1 \bar{x}_2} \right].$$

Операция матричного умножения с последующей инверсией соответствует представлению функций в базисе ИЛИ-НЕ. В выражении (1) умножение с инверсией используется дважды Это отвечает тому, что в ПЛМ на МОП-транзисторах (МОП-ПЛМ) и первый, и второй каскады являются наборами вентилей ИЛИ-НЕ Транспонирование в выражении (1) означает перемену ролями строк и столбцов во втором каскаде МОП-ПЛМ по сравнению с первым, инвертирование матриц **X** и **F** оответсгвует тому, что на МОП-ПЛМ подаются инвертированные переменные и значения функций на выходе ПЛМ также инвертированы. Все сказанное о соответствии матричного выражения (1) и принципиальной схемы ПЛМ можно проследить, сравнив пример 1 со схемой МОП-ПЛМ на рис. *4, а.*

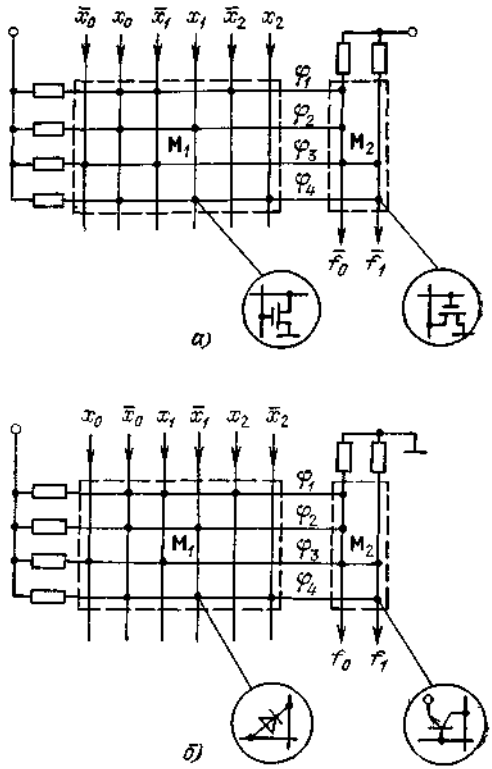


Рис. 4

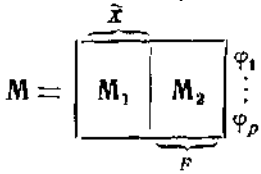
На рис. 4,б приведена схема ПЛМ на основе биполярных элементов, реализующая систему функций из примера 1. Для системы функций, реализованной в этой ПЛМ, матричное выражение имеет вид

$$\mathbf{F} = (\mathbf{M}_1 \odot \mathbf{X})^T \cdot \mathbf{M}_2. \qquad (2)$$

Матричная операция $\odot$ похожа на умножение матриц с заменой арифметических операций одноименными логическими, только вместо дизъюнкции (сложения) поэлементных произведений строки и столбца здесь конъюнкция произведений, не равных нулю:

$$(\mu_{i1}\mu_{i2} \ \dots \ \mu_{i2n}) \odot \begin{bmatrix} x_0 \\ \bar{x}_0 \\ \\ x_{n-1} \\ \bar{x}_{n-1} \end{bmatrix} = \bigwedge_{\mu_{ij} \wedge \tilde{x}_j \neq 0} (\mu_{ij} \wedge \tilde{x}_j).$$

На первом уровне биполярной ПЛМ производятся конъюнкции от входных переменных (операция $\odot$ в (2)), на втором — дизъюнкция логических произведений, чему соответствует операция матричного умножения. Матричное представление системы функций



(3а)

может рассматриваться как отображение принципиальной схемы и топологического рисунка ее на кристалле. Одной и той же булевой функции соответствует множество реализующих ее ДНФ. Следовательно, любой системе *F* булевых функций соответствует класс матриц **M**, каждая из которых реализует систему *F*. Задача синтеза ПЛМ — найти в классе матриц, реализующих систему *F*, одну матрицу, такую, которая удовлетворяет заданным условиям, например имеет минимальное число строк по сравнению со всеми другими матрицами в этом классе

6.1.4. Матричная реализация логического преобразователя конечного автомата

Пусть конечный автомат *A* задан множествами входных $\Sigma = \{\sigma_1, \ldots, \sigma_r\}$, внутренних $S = \{s_1, \ldots, s_v\}$ и выходных $\Gamma = \{\gamma_1, \ldots, \gamma_w\}$ состояний и функциями переходов $\delta : \Sigma \times S \to S$ и выходов $\lambda : \Sigma \times S \to \Gamma$. Кодирующие матрицы вида

$$C_\Sigma = \begin{matrix} g^0 & \cdots & g^{d-1} \end{matrix}\begin{bmatrix} g_1^0 & \cdots & g_1^{d-1} \\ g_2^0 & \cdots & g_2^{d-1} \\ \cdots & \cdots & \cdots \\ g_r^0 & \cdots & g_r^{d-1} \end{bmatrix}\begin{matrix} \sigma_1 \\ \sigma_2 \\ \cdots \\ \sigma_r \end{matrix};\qquad C_S = \begin{matrix} q^0 & \cdots & q^{h-1} \end{matrix}\begin{bmatrix} q_1^0 & \cdots & q_1^{h-1} \\ q_2^0 & \cdots & q_2^{h-1} \\ \cdots & \cdots & \cdots \\ q_v^0 & \cdots & q_v^{h-1} \end{bmatrix}\begin{matrix} s_1 \\ s_2 \\ \cdots \\ s_v \end{matrix};$$

$$C_\Gamma = \begin{matrix} z^0 & \cdots & z^{t-1} \end{matrix}\begin{bmatrix} z_1^0 & \cdots & z_1^{t-1} \\ z_2^0 & \cdots & z_2^{t-1} \\ \cdots & \cdots & \cdots \\ z_w^0 & \cdots & z_w^{t-1} \end{bmatrix}\begin{matrix} \gamma_1 \\ \gamma_2 \\ \cdots \\ \gamma_w \end{matrix},$$

где *g, q, z* — булевы переменные, ставят в соответствие каждому состоянию σ_i, s_j, γ_k набор значений переменных, а каждой

переменной $g^b \in G$, $q^m \in Q$, $z^n \in Z$ — два подмножества $\{\Sigma_b^1, \Sigma_b^0\}$, $\{S_m^1, S_m^0\}$, $\{\Gamma_u^1, \Gamma_u^0\}$ соответствующего множества состояний. При этом

$$\Sigma_b^1 = \{\sigma_i : g_i^b = 1\}; \quad \Sigma_b^0 = \{\sigma_i : g_i^b = 0\}; \quad S_m^1 = \{s_j : q_j^m = 1\};$$
$$S_m^0 = \{s_j : q_j^m = 0\}; \quad \Gamma_u^1 = \{\gamma_k : z_k^n = 1\}; \quad \Gamma_u^0 = \{\gamma_k : z_k^n = 0\}.$$

Будем использовать обозначения, подобные тем, которые были введены для систем булевых функций. Пусть $\Phi = \{\phi_1, \ldots, \phi_p\}$— множество полных состояний, при которых происходят переходы автомата, т. е. $\Phi \subseteq S \times \Sigma$. Полное состояние автомата $\phi_i = \{\sigma_j s_k\}$ может быгь выражено конъюнкцией

$$\varphi_i = \tilde{g}^0 \ldots \tilde{g}^{d-1} \tilde{q}^0 \ldots \tilde{q}^{h-1},$$

где $\tilde{g}^b = \begin{cases} g^b, & \text{если } g_j^b = 1, \\ \bar{g}^b, & \text{если } g_j^b = 0; \end{cases}$ $\tilde{q}^m = \begin{cases} q^m, & \text{если } q_k^m = 1, \\ \bar{q}^m, & \text{если } q_k^m = 0. \end{cases}$

Условимся через $Y = \{y^0, \ldots, y^{h-1}\}$ обозначать множество внутренних переменных, определяющих состояние перехода $s_l = \delta(\sigma_j s_k)$ $(l = 1, \ldots, v)$. Функции переходов автомата и функции выходов:

$$y^j = \bigvee_{\varphi_i : \delta(\phi_i) \in S_j^1} \varphi_i, \quad j = 0, \ldots, h-1;$$

$$z^j = \bigvee_{\varphi_i : \lambda(\phi_i) \in \Gamma_j^1} \varphi_i, \quad j = 0, \ldots, t-1,$$

или, аналогично (1), в матричной форме

$$\bar{\mathbf{Y}} = \overline{\left[\overline{\mathbf{M}_1 \cdot \begin{bmatrix} \bar{\mathbf{G}} \\ \bar{\mathbf{Q}} \end{bmatrix}}\right]^{\mathrm{T}} \cdot \mathbf{M}_2'};$$

$$\bar{\mathbf{Z}} = \overline{\left[\overline{\mathbf{M}_1 \cdot \begin{bmatrix} \bar{\mathbf{G}} \\ \bar{\mathbf{Q}} \end{bmatrix}}\right]^{\mathrm{T}} \mathbf{M}_2''}.$$

Матрица $\mathbf{M}_1 = \|\mu_{ij}\|$ размера $p \times 2(d+h)$, строки ее взаимно однозначно соответствуют множеству Φ, столбцы с номерами 1, 2, ... $2d$ — множеству $\tilde{G}$, с номерами $2d+1, \ldots, 2(d+h)$ — множеству $\tilde{Q}$;

$$\mu_{ij} = \begin{cases} 1, & \text{если } \tilde{g}^j \in G(\phi_i) \text{ при } j < 2d \\ & \text{или } \tilde{g}^m \in Q(\phi_i) \text{ при } j \geqslant 2d, \ m = j - 2d, \\ 0 & \text{в противном случае} \end{cases}$$

Матрица $\mathbf{M}_2 = \|\mu_{ij}\|$ размера $p \times h$, строки ее соответствуют элементам множества Φ, столбцы — элементам множества Y,

$$\mu_{ij} = \begin{cases} 1, & \text{если } \delta(\phi_i) \in S_j^1, \\ 0 & \text{в противном случае.} \end{cases}$$

Матрица $\mathbf{M}_2' = \|\mu_{ij}\|$ размера $p \times t$, строки ее соответствуют элементам множества Φ, столбцы — элементам множества Z;

$$\mu_{ij} = \begin{cases} 1, & \text{если } \lambda(\phi_i) \in \Gamma_j^1, \\ 0 & \text{в противном случае} \end{cases}$$

G, Q — столбцовые матрицы из элементов множеств $\tilde{G}$ и $\tilde{Q}$ соответственно.

Y, Z — строчные матрицы, содержащие функции переходов и выходов автомата.

Переход к матричному поедставтению автомата выполняется по следующим правилам

1. Если автомат задан таблицей или графом переходов, то построение матрицы $\mathbf{M}_1$ заключается в расстановке единиц в строках в соответствии с кодами входного и внутреннего состояний для каждой непустой клетки таблицы или дуги графа. Порядок расположения строк соответствует порядку возрастания номеров внутренних состояний, в которые автомат переходит при сооветствующих ϕ_i. Внутри подмножеств, приводящих автомат в одно и то же состояние, порядок строк произвольный. Порядок расположения столбцов такой же, как порядок нумерации элементов в множествах G и Q.

2. Матрица $\mathbf{M}_2$ получается из кодирующей матрицы C_S путем повторения каждой k-й строки ν_k раз, где ν_k —число полных состояний, приводящих автомат и состояние s_k.

3. Матрица $\mathbf{M}''_2$ получается из матрицы C_Γ, порядок строк определяется состоянием выходов для данного полного (для автомата Мили) или внутреннего (для автомата Мура) состояния.

Матрицы $\mathbf{M}_1$, $\mathbf{M}'_2$ и $\mathbf{M}''_2$ могут рассматриваться как одна матрица

$$\mathbf{M} = \left[\begin{array}{cccc} \overbrace{\quad}^{\tilde{G}} & \overbrace{\quad}^{\tilde{Q}} & & \\ \mathbf{M}_1 & \mathbf{M}_2' & \mathbf{M}_2 & \mathbf{M}_2 \\ \underbrace{\quad}_{Y} & & \underbrace{\quad}_{Z} & \end{array}\right] \begin{array}{l} \varphi_1 \\ \vdots \\ \varphi_p \end{array} \qquad \textbf{(3б)}$$

Построить эту матрицу можно, исходя из описания алгоритма функционирования автомата. Каждой строке соответствует один переход. Если это переход из состояния S_i в состояние s_j при

состояниях входов σ_i и выходов γ_k , то в строке проставляются единицы в соответствии с кодами σ_i, s_l, s_j γ_k в столбцах $\hat{G}$, $\tilde{Q}$, Y, Z сответственно. Столбцы $\tilde{G}$ и Z являются внешними (связанными с другими устройствами) входами и выходами ПЛМ, столбцы $\tilde{Q}$ и У — внутренними входами и выходами, которые замыкаются в цепи обратной связи через элементы памяти

6.1.5. Постановка задач матричного синтеза программируемых логических матриц

Здесь рассматриваются те задачи логического синтеза, которые вытекают из требований компактной реализации логического преобразователя в ПЛМ. Под компактной будем понимать реализацию логического преобразователя в ПЛМ, имеющей минимальную площадь. Площадь ПЛМ определяется размером матрицы **М**, представляющей логический преобразователь На практике чаще всего внешними связями определено заранее число входов и выходов проектируемой ПЛМ, т. е. задано число столбцов в матрице **М** (3а). В этих условиях поиски компактного матричного представления логического преобразователя сводятся к построению матрицы **М**, реализующей данный логический преобразователь и имеющей минимальное число строк по сравнению с другими его матричными представлениями. Здесь возможны два случая

1. Заданы коды входных и выходных состояний логического преобразователя, т. е. исходное матричное представление **М** как система булевых функций. В этом случае задача минимизации числа строк в матрице **М** является задачей построения кратчайшей ДНФ системы булевых функций.
2. Коды входных состояний не заданы заранее (хотя заранее определена разрядность кода, чаще всего минимальная). В этом случае имеется возможность таким образом закодировать входные состояния, чтобы система булевых функций, порожденных кодом, имела бы кратчайшую ДНФ, наиболее короткую по сравнению с ДНФ системы, порожденной любым другим кодом. Задачу поиска такого кода будем называть **экономичным кодированием**.

Для матричного представления (3б) логического преобразователя конечного автомата решаются аналогичные задачи. Если заданы коды входных, выходных и внутренних состояний автомата, то соответствующая матрица **М** в виде (3б) представляет систему булевых функций в ДНФ (конъюнкции реализуются в $\mathbf{M}_1$, дизъюнкции конъюнкций — в M_2) и решается задача приведения ДНФ системы

булевых функций к кратчайшему виду. Если не заданы коды внутренних состояний автомата (считаем, что коды входных и выходных состояний заданы), решается задача их экономичного кодирования. При проектировании асинхронного автомата задача кодирования усложняется, так как в этом случае коду необходимо обеспечить и экономичность, и противогоночность.

Таким образом, мы определили **три задачи синтеза компактной ПЛМ- приведение ДНФ системы булевых функций к кратчайшему виду, экономичное кодирование состояний логического преобразователя, противогоночное кодирование внутренних состояний автомата**. Далее даны точные постановки этих задач.

Задача 1. Приведение ДНФ системы булевых функций к кратчайшему виду. Обозначим I_i^0, I_i^1, I_i^* подмножества элементов n-мерного булева пространства $I^{(n)} = \{a_0, \ldots, a_{2^n-1}\}$, на которых булева функция $f_i(X) \in F$, $X = \{x_0, \ldots, x_{n-1}\}$, имеет соответственно нулевое, единичное и неопределенное значения. Систему булевых функций $F(X) = \{f_0, \ldots, f_{m-1}\}$, среди которых могут быть частичные, будем задавать в виде множества $\Psi_0 = \{(\alpha_j, \beta_j)\}$ $(j = 1, \ldots, p_0)$ пар в общем случае троичных векторов. Пару (α_j, β_j) будем называть *обобщенным* (на множество функций) *интервалом.* Часть α_j обобщенного интервала назовем *входной,* а часть β_j — *выходной.* Входная часть $\alpha_j = \alpha_j^{n-1} \ldots \alpha_j^0$ обобщенного интервала является интерналом в пространстве булевых переменных $X = \{x_0, \ldots x_{n-1}\}$. Выходная часть $\beta_j = \beta_j^{m-1} \ldots \beta_j^i \ldots \beta_j^0$ обобщенного интервала является троичным вектором значений функций $f_0, \ldots, f_i, \ldots, f_{m-1}$ на интервале α_j. При этом $\beta_j^i = 1$, если $I_{\alpha_j} \subseteq I_i^1$, $\beta_j^i = 0$, если $I_{\alpha_j} \subseteq I_i^0$, $\beta^i{}_j = *$, если $I_{\alpha_j} \subseteq I_i^*$.

Обозначим через $\tilde{I}_i^0$ объединение интервалов α_j, таких, для которых

$$\beta_j^i = 0 : \tilde{I}_i^0 = \bigcup_{\substack{j=1 \\ \alpha_j : \beta_j^i = 0}}^{p} I_{\alpha_j}.$$

Аналогично

$$\tilde{I}_i^1 = \bigcup_{\substack{j=1 \\ \alpha_j : \beta_j^i = 1}}^{p} I_{\alpha_j}.$$

Множество $\Psi_0 = \{(\alpha_j, \beta_j)\}$ представляет систему *F*, если для каждой функции $f_i \in F$ выполнены условия: 1) $\tilde{I}^1_i = I^1_i$; 2) $I^0_i = I^0_i$. Другими словами, множество обобщенных интервалов Ψ_0 представляет области единичных и нулевых значений всех функций системы. Заметим, что входные и выходные части обобщенных интервалов из множества Ψ_0 могут быть двоичными векторами, если входные части являются элементами булева пространства (*n*-интервалами) и либо среди функций системы нет частичных, либо области неопределенных значений всех функций совпадают.

Поясним, почему разряды векторов α_j и β_j перенумерованы справа налево. Далее будет использовано десятичное представление двоичного вектора, при этом естественно разряды двоичного вектора нумеровать справа налево. Поскольку входной частью обобщенного интервала может быть двоичный вектор, то принята нумерация ее разрядов справа налево, для единообразия и разряды выходной части обобщенного интервала нумеруются справа налево.

Множество обобщенных интервалов $\Psi = \{(\alpha_j, \beta_j)\}$ реализует ДНФ системы *F* (или просто систему *F)*, если для каждой функции $f_i \in F$ выполнены условия:

1) $\tilde{I}^1_i \cap I^0_i = \varnothing$; 2) $I^1_i \subseteq \tilde{I}^1_i$; 3) $I_{\alpha_j} \cap I^1_i \neq \varnothing$ при $\beta^i_j = 1$.

Эти условия означают, что входные части α_j всех обобщенных интервалов являются допустимыми для своих функций (т. е. для тех функций, которые помечены единицами в β_j и область единичных значений I^1_i каждой функции $f_i \in F$ покрыта допустимыми для этой функции интервалами.

Заметим, что если из множества обобщенных интервалов Ψ_0 удалить все те интервалы, в выходных частях которых нет единиц, то полученное множество реализует некоторую ДНФ системы *F*.

Из множества Ψ_0 обобщенных интервалов можно восстановить ДНФ системы *F*. Для этого каждый обобщенный интервал надо представить в виде конъюнкции переменных *X*, соответствующей входной его части, и эту конъюнкцию пометить теми функциями, которым соответствуют единицы в выходной части обобщенного интервала.

Пример 2. Пусть система $F_2(x_0, x_1, x_2) = \{f_0, f_1, f_2\}$, функции которой имеют следующие области определенных значений $I^1_0 = \{010, 101\}$, $I^0_0 = \{000, 001, 011, 100\}$, $I^1_1 = \{000, 010\}$, $I^0_1 = \{001, 100\ 101\}$, $I^1_2 = \{000, 001, 010, 011\}$, $I^0_2 = \{100\}$, представлена множеством обобщенных интервалов

$$\Psi_0 = \begin{array}{ll} (\alpha_i,\ \beta_i) & i \\ (000,\ 110) & 1 \\ (001,\ 100) & 2 \\ (010,\ 111) & 3 \\ (011,\ 1*0) & 4 \\ (100,\ 000) & 5 \\ (101,\ *01) & 6 \end{array}$$

На наборах 110, 111 все функции системы не определены, поэтому в множестве Ψ_0 нет обобщенных интервалов, входные части которых содержали бы эти наборы. Очевидно, что множество Ψ_0 после удаления из него интервала (α_5, β_5) реализует следующую ДНФ данной системы: $F_2 = \bar{x}_0\bar{x}_1\bar{x}_2(f_1,\ f_2) \vee x_0\bar{x}_1\bar{x}_2(f_2) \vee \bar{x}_0x_1\bar{x}_2(f_0,\ f_1,\ f_2) \vee \vee x_0x_1\bar{x}_2(f_2) \vee x_0\bar{x}_1x_2(f_0)$ Множество обобщенных интервалов

$$\Psi_1 = \begin{cases} (0*0,110), \\ (0*1,100), \\ (*10,001), \\ (1*1,*01) \end{cases}$$

реализует другую ДНФ заданной системы $F_2 = x_0'x_2'(f_1,\ f_2) \vee \vee x_0\bar{x}_2(f_2) \vee \bar{x}_0x_1(f_0) \vee x_0x_2(f_0)$. Действительно, входные части всех интерваловиз Ψ_1 являются допустимыми для своих функций и $I_0^1 \subset \tilde{I}_0^1$ ($\tilde{I}_0^1 = \{010,\ 101,\ 110,\ 111\}$), $\tilde{I}_1^1 = I_1^1$, $\tilde{I}_2^1 = I_2^1$

Обобщенный интервал (α_j, β_j) в выходной части которого есть хотя бы одна единица, будет соответствовать строке матрицы $\mathbf{M} = [\mathbf{M}_1 \mid \mathbf{M}_2]$, если значения 0,1,$*$ во входной части его закодировать как 01, 10, 00 соответственно, а неопределенным значениям в выходной части присвоить нули. Тогда множество обобщенных интервалов будет представляться матрицей **M** и становится понятным смысл названий «входной», «выходной» частей обобщенного интервала: входные части составят входную часть ПЛМ (подматрицу $\mathbf{M}_1$), выходные — выходную часть ПЛМ (подматрицу $\mathbf{M}_2$).

Пример 3. Матрица **M**, представляющая множество обобщенных интервалов Ψ_1 из примера 2, имеет вид

$$\mathbf{M} = \left[\begin{array}{cccccc|ccc} x_2 & \bar{x}_2 & x_1 & \bar{x}_1 & x_0 & \bar{x}_0 & f_2 & f_1 & f_0 \\ \hline 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Обобщенный интервал (α_j, β_j), в выходной части которого есть хотя бы одна единица, называется *максимальным,* если интервал α_j нельзя расширить без того, чтобы расширенный интервал не пересекся с областью нулевых значений хотя бы одной из функций системы, помеченных в β_j единицами. *Множество максимальных обобщенных интервалов,* реализующее систему *F,* называется *кратчайшим,* если оно имеет наименьшую мощность по сравнению со всеми другими множествами максимальных обобщенных интервалов, реализующими систему *F*. Очевидно, что кратчайшее множество обобщенных интервалов Ψ соответствует кратчайшей ДНФ системы *F* и вкладывается в ГТЛМ с наименьшим числом строк.

Таким образом, первая задача состоит в преобразовании исходного множества обобщенных интервалов Ψ_0, представляющего систему булевых функций *F,* к кратчайшему множеству обобщенных интервалов. Эта задача классическая. Существуют методы точного ее решения для систем от малого числа (5,6) переменных и приближенного решения этой задачи для систем с большими значениями параметров: несколько десятков переменных и функций, несколько сотен конъюнкций в ДНФ системы. С появлением ПЛМ возрос угасший было интерес к этой задаче, что привело к интенсивному развитию методов решения задачи минимизации, использующих идею направленного поиска, при этом направление поиска выбирается по тем или иным эвристическим критериям.

Задача 2. Экономичное кодирование. Пусть задано множество состояний $S = \{s_1, \ldots, s_p\}$. Каждому состоянию s_i поставлен в соответствие вектор β_i —выходная часть гипотетического обобщенного интервала (s_i, β_i). Задача экономичного кодирования состояний $s_i \in S$ при минимальной длине кодирующего слова состоит в присвоении каждому состоянию s_i двоичного набора a_i длины $n =]\log_2 p * (]\log_2 p[$ — ближайшее к $\log_2 p$ целое число, большее или равное $\log_2 p$), таким образом, чтобы множество обобщенных интервалов $\Psi_0 = \{(a_i, \beta_i)\}$ моглобыть приведено к кратчайшему виду, наиболее короткому по сравнению с кратчайшим множеством, полученным при любом другом кодировании состояний S. Иными словами, задается матрица $\mathbf{M}_2$ размера $p \times tn$, требуется построить матрицу $\mathbf{M}_1$ размера $p \times 2n$, такую, чтобы система булевых функций, записанная в матрице $\mathbf{M} = [\mathbf{M}_1 \mid \mathbf{M}_2]$, имела бы кратчайшую ДНФ, наиболее короткую по сравнению с тем, что могла бы дать любая другая $\mathbf{M}_1$.

Множеству строк матрицы $\mathbf{M}_2$ поставим во взаимно однозначное соответствие множество состояний $S = \{s_1, \ldots, s_p\}$. В матрице $\mathbf{M}_2$ можно

выделить множество W подматриц w_i, не содержащих ни одного нуля. Наименьшее число подматриц w_i, таких, что каждая единица из $\mathbf{M}_2$ входит хотя бы в одну w_i, составляет минимальное покрытие W_{min} матрицы $\mathbf{M}_2$. Если матрица $\mathbf{M}_2$ имеет не единственное минимальное покрытие, то мы будем иметь в виду любое из таких покрытий.

Пример 4. Пусть

$$\mathbf{M}_2 = \begin{array}{c} \begin{matrix} 3 & 2 & 1 & 0 \end{matrix} \\ \left[\begin{array}{cccc} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & * & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & * & 1 \end{array}\right] \end{array} \begin{matrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{matrix}$$

Минимальное покрытие составляют следующие ее подматрицы:

$$w_1 = \begin{array}{c} \begin{matrix} 3 & 2 \end{matrix} \\ \left[\begin{array}{cc} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{array}\right] \end{array} \begin{matrix} s_1 \\ s_2 \\ s_4 \end{matrix}, \quad w_2 = \begin{array}{c} \begin{matrix} 2 & 1 \end{matrix} \\ \left[\begin{array}{cc} 1 & 1 \\ * & 1 \\ 1 & 1 \end{array}\right] \end{array} \begin{matrix} s_2 \\ s_3 \\ s_5 \end{matrix},$$

$$w_3 = \begin{array}{c} \begin{matrix} 3 & 1 \end{matrix} \\ \left[\begin{array}{cc} 1 & 1 \\ 1 & * \end{array}\right] \end{array} \begin{matrix} s_6 \\ s_7 \end{matrix}, \quad w_4 = \begin{array}{c} \begin{matrix} 0 \end{matrix} \\ \left[\begin{array}{c} 1 \\ 1 \end{array}\right] \end{array} \begin{matrix} s_1 \\ s_7 \end{matrix}.$$

Каждой подматрице $w_i \in W_{\min}$ поставим в соответствие подмножество $K_i \subset S$, определяемое теми строками матрицы $\mathbf{M}_2$, на которых образована w_i. Подмножество K_i будем называть *K-множеством.* Подматрицы $w_1, ..., w_4$ из примера 4 порождают следующие *K*-множества: $K_1 = \{s_1, s_2, s_4\}$; $K_2 = \{s_2, s_3, s_5\}$; $K_3 = \{s_6, s_7\}$; $K_4 = \{s_1, s_7\}$. Если длина n кодирующего слова может быть выбрана произвольно, то множество состояний S может быть закодировано таким образом, что множество двоичных наборов, соответствующее каждому подмножеству K_i, войдет целиком в некоторый интервал α_j n-мерного булева пространства, не содержащий ни одного набора, присвоенного любому $s_k \notin K_i$. Тогда множество обобщенных интервалов $\Psi = \{(\alpha_i, \beta_i)\}$ (i=1, ... $|W_{min}|$), где β_i определяется подматрицей $w_i \in W_{\min}$ (при этом $\beta^k_i = 1$, если на k-*м* столбце матрицы $\mathbf{M}_2$ определена w_i, $\beta^k_i = 0$ в противном случае), будет являться кратчайшим для системы $F(X)$, порожденной принятым кодом C_S. Эта система может быть представлена матрицей $\mathbf{M}$ с числом строк, равным $|W_{\min}|$. Очевидно, что ни один код для заданной $\mathbf{M}_2$ не может дать матричное представление соответствующей ему системы с меньшим числом строк, чем $|W_{\min}|$.

Пример 5. К матрице $\mathbf{M}_2$ из примера 4 присоединим матрицу $\mathbf{M}_1$, соответствующую следующему коду состояний $s_1, \ldots, s_7$:

$$\mathbf{M}_1 = \left[\begin{array}{ccc|c} 0 & 1 & 0 & s_1 \\ 1 & 1 & 0 & s_2 \\ 1 & 1 & 1 & s_3 \\ 0 & 0 & 0 & s_4 \\ 1 & 0 & 1 & s_5 \\ 0 & 0 & 1 & s_6 \\ 0 & 1 & 1 & s_7 \end{array}\right].$$

Такое кодирование дает следующее кратчайшее множество обобщенных интервалов

$$(**0,\ 1100),\quad (1**,\ 0110),\quad (0*\ 1,\ 1010),\quad (01*,\ 0001).$$

Следовательно, ПЛМ, реализующая полученную систему функций, имеет четыре строки, и ни один другой код не может дать ПЛМ с меньшим числом строк.

При минимальной длине кодирующего слова $n =]\log_2 p[$ может не найтись такого кода C_S, который позволил бы реализовать каждое K-множество одним интервалом. При определенном коде C_S каждое K_i может быть реализовано некоторым подмножеством булевых интервалов, не содержащих кодов, присвоенных состояниям $s_k \notin K_i$. Множество интервалов, наименее мощное из всех реализующих совокупность K-множеств, порождаемую покрытием W_{min}, обозначим через $R_{\min}^{C_S}$. *Наилучшим в смысле экономичности кодом* множества S для $\mathbf{M}_2$ при длине $n =]\log_2 p[$ будем называть такой C'_S, что $|R_{\min}^{C'_S}| < |R_{\min}^{C_S}|$ для любого другого кода C_S с тем же n.

Точное решение задачи экономичного кодирования, требующее перебора кодов, неприемлемо громоздко. Поэтому будем с помощью эвристических приемов строить «хороший» код.

Допустим следующее отступление от точной постановки задачи. В практических случаях в матрице $\mathbf{M}_2$ число столбцов существенно меньше, чем число строк (матрица сильно вытянута по вертикали). Минимальным покрытием такой матрицы (определяющим K-множества) или близким к нему является покрытие столбцовыми подматрицами, не содержащими нулей. Поэтому в качестве исходной информации при экономичном кодировании мы будем брать совокупность K-множеств, соответствующих столбцовым подматрицам, а в алгоритме кодирования учитывать пересечения таких K-множеств, имея в виду, что любое K-множество, соответствующее $w_i \in W_{\min}$, является пересечением «столбцовых» K-множеств.

При кодировании внутренних состояний автомата *К* -множества выделяются для каждой пары (σ_i, s_j) (входное состояние, внутреннее состояние). В одно *К*-множество входят те внутренние состояния, из которых при входе σ_i автомат переходит в состояние s_j (для автоматов Мура) и имеет выходное состояние γ_k (для автомата Мили).

Задача 3. Противогоночное кодирование. При функционировании асинхронного автомата могут возникнуть так называемые состязания или гонки между элементами памяти. Гонки возникают, если при переходе автомата и из одного состояния в другое должны переключиться два или более элементов памяти. Из-за небольшого разброса во времени срабатывания элементов памяти, неодинаковых длин логических цепей, случайных задержек один или несколько элементов могут переключиться раньше, чем остальные, т. е. выиграть гонки. Это может привести к переходу автомата в состояние, не предусмотренное его функцией переходов, т. е. к неправильному срабатыванию. В таком случае гонки называются опасными или критическими. Если автомат под влиянием гонок не отклоняется от заданного поведения, гонки называются неопасными или некритическими.

Пример 6. В автомате при одном и том же входном сигнале имеются два перехода $s_1 \to s_2$, $s_3 \to s_4$ —из состояния $s_1(s_3)$ автомат переходит в состояние $s_2(s_4)$. Состояниям присвоены коды: $s_1 = 000$, $s_2 = 110$, $s_3 = 010$, $s_4 = 100$. При переходе $s_1 \to s_2$ возникает гонка между первым и вторым элементами памяти (разряды кода нумеруются справа налево, начиная с нуля, номера элементов памяти совпадают с номерами разрядов кода). Если гонку выиграет, например, первый элемент, то автомат вместо состояния s_2 попадает в состояние s_4, это значит, состязание опасно. Состязание было бы неопасным, если состояния s_3, и s_4 были бы закодированы соответственно 011 и 101. Опасные гонки в асинхронном автомате устраняются с помощью специального кодирования внутренних состояний, которое называют *противогоночным*. Сформулировано необходимое и достаточное условие отсутствия опасных состязаний в асинхронном автомате. Суть его в следующем. Два перехода $s_l \to s_k$ и $s_q \to s_r$, происходящие под действием одного и того же входного сигнала и такие, что $s_k \neq s_r$, называют *состязающимися.* Два перехода в автомате, состояния которого закодированы двоичным кодом, называют *развязанными,* если хотя бы один (пусть *i*-й) разряд кода имеет одно значение для пары состояний (s_l, s_k) и противоположное — для пары (s_q, s_r). Например, при $s_l = 000$, $s_k = 110$, $s_q = 011$, $s_r = 101$ переходы $(s_q \to s_r)$, $(s_l \to s_k)$ развязаны по нулевому разряду кода. Итак, в

автомате, состояния которого закодированы двоичным кодом, опасные состязания отсутствуют тогда и только тогда, когда развязаны любые два состязающихся перехода.

Кроме этого условия при противогоночном кодировании часто используют достаточное условие отсутствия опасных состязаний в автомате. Оно формулируется для *k*-множеств внутренних состояний — множеств, из элементов которых при фиксированном входе автомат переходит в одно и то же состояние. Покажем на примере, как из таблицы переходов автомата выделить *k*-множества.

Пример 7. Из таблицы переходов автомата

	x_1	x_2	x_3
s_1	s_1	s_4	s_2
s_2	s_1	s_3	s_2
s_3	s_1	s_5	s_3
s_4	s_4	s_4	s_3
s_5	s_4	s_6	s_5
s_6	s_4	s_6	s_5

можно выделить следующие *k*-множества: по первому столбцу $k_1 = \{s_1, s_2, s,\}$, $k_2 = \{s_4, s_5, s_6\}$; по второму $k_3 = \{s_1, s_4\}$, $k_4 = \{s_2, s_3\}$, $k_5 = \{s_5, s_6\}$; по третьему $k_6 = \{s_1, s_2\}$, $k_7 = \{s_3, s_4\}$, $k_8 = \{s_5, s_6\}$.

Заметим, что *k*-множества внутренних состояний автомата, используемые при противогоночном кодировании, совпадают с *K*-множествами, выделяемыми по иходным состояниям и состояниям перехода без учета иыхода и используемыми при экономичном кодировании.

Пары состязающихся *k*-множеств рассматривают подобно парам состязающихся переходов. Так, в примере 7 состязаются k_1 и k_2, попарно состязаются k_3, k_4, k_5, так же, как и k_6, k_7, k_8. Два *k*-множества (k_q, k_r) считаются развязанными, если хотя бы один, пусть *i*-й, разряд кода имеет одно значение для всех состояний множества k_q и противоположное — для всех состояний множества k_r. Достаточное условие отсутствия опасных состязаний в автомате — все пары состязающихся *k*-множеств развязаны.

При противогоночном кодировании требуется минимизировать длину кода и, следовательно, общее число столбцов матричного представления автомата.

6.2. Теоретические основы алгоритмов матричного синтеза

6.2.1. Теоретико-структурные свойства совокупности разбиения булева пространства

Разбиения булева пространства. *Разбиением* множества S называется множество подмножеств $S_k \subset S$, таких, что:
1) $S_i \cap S_j = \varnothing$ при $i \neq j$ (подмножества S_k попарно не пересекаются),
2) $\bigcup S_k = S$ (все вместе подмножества S_k образуют множество S).
Множество $I^{(n)} = \{a_0, \ldots, a_{2^n-1}\}$ элементов (*n*-интервалов) *n*-мерного булева пространства разобьем на подмножества мощности 2^v, $v \in \{0, \ldots, n\}$, таким образом, чтобы эти подмножества являлись булевыми интервалами ранга *n*—*v*, определенными на одной совокупности координат $\{x_{i_1}, \ldots, x_{i_v}\} \subset \{x_0, \ldots, x_{n-1}\}$. В совокупность включаются те координаты, по которым *n*-интервалы одного подмножества склеиваются в интервал ранга *n*—*v*, иначе говоря, те координаты, которым в троичном векторе, представляющем интервал, соответствуют неопределенные значения $(*)$. Например, интервал {0000, 0001, 0010, 0011}, представленный троичным вектором $00{*}{*}$, определен на совокупности $\{x_0, x_1\}$. Подмножества разбиения будем называть блоками, элементы каждого блока будем объединять чертой сверху.

Пример 8. Разбиение множества $I^{(3)}$ ={000, 001, 010, 011, 100, 101, 110, 111} для *v*=2 и определяющей совокупности координат $\{x_0, x_2\}$ имеет вид

$$\{\overline{000,\ 001,\ 100,\ 101};\ \overline{010,\ 011,\ 110,\ 111}\}.$$

Имеется C^v_n (число сочетаний из *n* элементов по *v*) различных разбиений множества $I^{(n)}$ на блоки мощности 2^v так же, как имеется C^v_n различных совокупностей координат мощности *v*. Множество таких разбиений будем обозначать $\Theta^v = \{\theta^v_i\}$ $(i = 0, \ldots, C^v_n - 1)$, разбиение в множестве — $\theta^v_i = \{B_j\}$, $|B_j| = 2^v$, $j = 0, \ldots, 2^{n-v} - 1$, B_j — блок в разбиении.

Пример 9 Запишем все разбиения множества $I^{(3)}$ на блоки мощности 2^2 (*v*=2). Рядом с каждым разбиением укажем определяющую совокупность координат

$$\theta_0^2 = \{\overline{000,\ 001,\ 010,\ 011};\quad \overline{100,\ 101,\ 110,\ 111}\},\ \{x_0,\ x_1\},$$
$$\theta_1^2 = \{\overline{000,\ 001,\ 100,\ 101};\quad \overline{010,\ 011,\ 110,\ 111}\},\ \{x_0,\ x_2\},$$
$$\theta_2^2 = \{\overline{000,\ 010,\ 100,\ 110};\quad \overline{001,\ 011,\ 101,\ 111}\},\ \{x_1,\ x_2\}.$$

Заставив ν «пробежать» все значения из множества $\{0, ..., n\}$, мы получим совокупность $\Theta = \cup\Theta^{\nu}$ множеств разбиений. Чтобы сделать запись разбиений менее громоздкой, вместо двоичных наборов будем использовать их десятичные представления.

Пример 10. Совокупность Θ разбиений множества $I^{(3)}$ (см, пример 8):

$$\Theta^0 = \{\overline{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7}\},\ \{\varnothing\},$$

$$\Theta^1 = \begin{cases} \theta_0^1 = \{\overline{0,1};\ \overline{2,3};\ \overline{4,5};\ \overline{6,7}\},\ \{x_0\}, \\ \theta_1^1 = \{\overline{0,2};\ \overline{1,3};\ \overline{4,6};\ \overline{5,7}\},\ \{x_1\}, \\ \theta_2^1 = \{\overline{0,4};\ \overline{1,5};\ \overline{2,6};\ \overline{3,7}\},\ \{x_2\}; \end{cases}$$

$$\Theta^2 = \begin{cases} \theta_0^2 = \{\overline{0,\ 1,\ 2,\ 3};\quad \overline{4,\ 5,\ 6,\ 7}\},\ \{x_0,\ x_1\}, \\ \theta_1^2 = \{\overline{0,\ 1,\ 4,\ 5};\quad \overline{2,\ 3,\ 6,\ 7}\},\ \{x_0,\ x_2\}, \\ \theta_2^2 = \{\overline{0,\ 2,\ 4,\ 6};\quad \overline{1,\ 3,\ 5,\ 7}\},\ \{x_1,\ x_2\}; \end{cases}$$

$$\Theta^3 = \{\overline{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7}\},\ \{x_0,\ x_1,\ x_2\}.$$

Совокупность разбиений как структура. Будем рассматривать отношение эквивалентности $[\theta^{\nu}_i]$ на множестве $I^{(n)}$: $a_g\,[\theta^{\nu}_i]\,a_h$ означает, что a_q, a_h находятся в одном и том же блоке разбиения θ^{ν}_i. Разбиения можно складывать и умножать по следующим правилам .

1. Сложение. $\theta^{\mu}_k = \theta^{\nu}_i + \theta^{\gamma}_j$, a_g, a_h попадут в один блок разбиения θ^{μ}_k, если найдутся $a_{g_1}, ..., a_{g_m}$, такие, что

$$a_g\,[\theta^{\nu}_i]\,a_{g_1},\ a_{g_1}\,[\theta^{\gamma}_j]\,a_{g_2},\ a_{g_2}\,[\theta^{\nu}_i]\,a_{g_3},\ ..,\ a_{g_m}\,[\theta^{\gamma}_j]\,a_h.$$

Иными словами, каждый блок разбиения θ^{μ}_k является объединением всех блоков из θ^{ν}_i и θ^{γ}_j, таких, отношению пересечения которых соответствует связный граф.

2. Умножение. $\theta^{\xi}_r = \theta^{\nu}_i \cdot \theta^{\gamma}_j$, a_g, a_h попадут в один блок разбиения θ^{ξ}_r, если $a_g\,[\theta^{\nu}_i]\,a_h$ и $a_g\,[\theta^{\gamma}_j]\,a_h$. Иными словами, каждый блок разбиения θ^{ξ}_r равен пересечению блоков из θ^{ν}_i и θ^{γ}_j, если пересечение не пусто.

Пример 11. Пусть имеются два разбиения из совокупности Θ при *n=4:*

$\theta^2_0 = \{\overline{0, 1, 2, 3};\ \overline{4, 5, 6, 7};\ \overline{8, 9, 10, 11};\ \overline{12, 13, 14, 15}\},$

$\theta^2_1 = \{\overline{0, 1, 4, 5};\ \overline{2, 3, 6, 7};\ \overline{8, 9, 12, 13};\ \overline{10, 11, 14, 15}\}.$

$\theta^2_0 + \theta^2_1 = \{\overline{0, 1, 2, 3, 4, 5, 6, 7};\ \overline{8, 9, 10, 11, 12, 13, 14, 15}\},$

$\theta^2_0 \cdot \theta^2_1 = \{\overline{0,1};\ \overline{2,3};\ \overline{4,5};\ \overline{6,7};\ \overline{8,9};\ \overline{10,11};\ \overline{12,13};\ \overline{14,15}\}.$

На множестве $\Theta = \{\theta^\nu_i\}$ $(\nu = 0, \ldots, n;\ i = 0, \ldots, C^\nu_n - 1)$ будем рассматривать отношение частичного порядка:

$\theta^\nu_i \leqslant \theta^\gamma_j$ означает, что из $a_g [\theta^\nu_i] a_h$ следует $a_g [\theta^\gamma_j] a_h$.

Другими словами, для каждого блока $B_l \in \theta^\nu_i$ в разбиении θ^γ_j найдется блок B_q, целиком включающий блок B_l. Например, нетрудно убедиться, что $\theta^2_0 > \theta^1_0$, $\theta^2_0 > \theta^1_1$, а θ^2_0 и θ^1_2 не сравнимы (θ^1_0, θ^1_1, θ^1_2, θ^2_0 см. в примере 10).

В частично упорядоченном множестве в *нижней границей* для двух разбиений θ^ν_i и θ^γ_j является разбиение θ^μ_k, такое, что $\theta^\mu_k \leqslant \theta^\nu_i$ и $\theta^\mu_k \leqslant \theta^\gamma_j$. Разбиение θ^μ_k является *максимальной нижней границей* для θ^ν_i, θ^γ_j, если любая нижняя граница θ^ξ_r для θ^ν_i, θ^γ_j удовлетворяет отношению $\theta^\xi_r \leqslant \theta^\mu_k$. Аналогично *верхней границей* для θ^ν_i, θ^γ_j является разбиение θ^π_m, такое, что $\theta^\pi_m \geqslant \theta^\nu_i$ и $\theta^\pi_m \geqslant \theta^\gamma_j$.

Разбиение θ^π_m является *минимальной верхнейграницей* для θ^ν_i, θ^γ_j, если любая верхняя граница θ^τ_n для θ^ν_i, θ^γ_j удовлетворяет соотношению $\theta^\pi_m \leqslant \theta^\tau_n$.

Например, разбиение θ^3 (см. пример 10) является верхней границей, а разбиение θ^2_2— минимальной верхней границей для θ^1_1 и θ^1_3.

Любые два разбиения θ^ν_i, θ^γ_j имеют максимальную нижнюю и минимальную верхнюю границы. В самом деле, в разбиениях, входящих в Θ, содержатся все интервалы n-мерного булева пространства, а сравнение разбиений осуществляется поблочно. Отсюда следует, что для любой пары пересекающихся блоков $B_t \in \theta^\nu_i$ и $B_n \in \theta^\gamma_j$, $B_l \cap B_q \neq \varnothing$, найдется минимальный объемлющий блок B_p в некотором разбиении θ^π_m, $B_l \subset B_p$, $B_q \subset B_p$, а это значит, что разбиение θ^π_m является минимальной верхней границей для θ^ν_i и θ^γ_j. При этом $\theta^\pi_m = \theta^\nu_i + \theta^\gamma_j$.

Аналогично для любой пары пересекаю щихся блоков $B_l \in \theta_i^{\nu}$ и $B_q \in \theta_j^{\tau}$ найдется блок $B_p = B_l \cap B_q$ в некотором разбиении $\theta^{\mu}{}_k$, которое и является максимальной нижней границей для θ_i^{ν} и θ_j^{τ}. При этом $\theta_k^{\mu} = \theta_i^{\nu} \cdot \theta_j^{\tau}$.

Известно, что **частично упорядоченное множество является *структурой,* если в нем каждые два элемента имеют максимальную нижнюю и минимальную верхнюю границы.** На этом основании считаем, что частично упорядоченное множество разбиений $\hat{\Theta} = <\Theta, \leqslant>$ является структурой. В структуре $\hat{\Theta}$ имеются общие для всех разбиений универсальные верхняя и нижняя границы:

$$\Theta^n = \{\overline{a_0, a_1, \ldots, a_{2^n-1}}\}; \quad \Theta^0 = \{\overline{a}_0, \overline{a}_1, \ldots, \overline{a}_{2^n-1}\}.$$

Универсальные верхняя и нижняя границы называются *единицей* и *нулем структуры* соответственно.

Без доказательства скажем, что структура $\hat{\Theta}$, обладающая нулем и единицей, является *дистрибутивной, замкнутой относительно дополнения* или *булевой структурой* (в этом легко убедиться, рассмотрев все понятия на конкретной структуре из примера 10). Дистрибутивность структуры следует из того, что в ней справедливы аксиомы дистрибутивности:

$$\theta_i^{\nu} \cdot (\theta_j^{\tau} + \theta_k^{\mu}) = (\theta_i^{\nu} \cdot \theta_j^{\tau}) + (\theta_i^{\nu} \cdot \theta_k^{\mu});$$

$$\theta_i^{\nu} + (\theta_j^{\tau} \cdot \theta_k^{\mu}) = (\theta_i^{\nu} + \theta_j^{\tau}) \cdot (\theta_i^{\nu} + \theta_k^{\mu}).$$

Пример 12. Проверим второй закон дистрибутивности. Рассмотрим три разбиения множества $I^{(4)}$, входящие в структуру в $\hat{\Theta}$:

$$\theta_i^{\nu} = \{\overline{0,1};\ \overline{2,3};\ \overline{4,5};\ \overline{6,7};\ \overline{8,9};\ \overline{10,11};\ \overline{12,13};\ \overline{14,15}\};$$

$$\theta_j^{\tau} = \{\overline{0, 2, 4, 6};\ \overline{1, 3, 5, 7};\ \overline{8, 10, 12, 14};\ \overline{9, 11, 13, 15}\};$$

$$\theta_k^{\mu} = \{\overline{0, 1, 2, 3, 8, 9, 10, 11};\ \overline{4, 5, 6, 7, 12, 13, 14, 15}\}.$$

Произведем над разбиениями следующие действия:

$$\theta_i^{\nu} \cdot \theta_k^{\mu} = \{\overline{0,2};\ \overline{4,6};\ \overline{1,3};\ \overline{5,7};\ \overline{8,10};\ \overline{12,14};\ \overline{9,11};\ \overline{13,15}\};$$

$$\theta_i^{\nu} + (\theta_j^{\tau} \cdot \theta_k^{\mu}) = \{\overline{0, 1, 2, 3};\ \overline{4, 5, 6, 7};\ \overline{8, 9, 10, 11};\ \overline{12, 13, 14, 15}\};$$

$$\theta^{\nu}{}_i + \theta_j^{\tau} = \{\overline{0, 1, 2, 3, 4, 5, 6, 7};\ \overline{8, 9, 10, 11, 12, 13, 14, 15}\};$$

$$\theta^{\nu}{}_i + \theta_k^{\mu} = \{\overline{0, 1, 2, 3, 8, 9, 10, 11};\ \overline{4, 5, 6, 7, 12, 13, 14, 15}\},$$

$$(\theta^{\nu}{}_i + \theta_j^{\tau}) \cdot (\theta_i^{\nu} + \theta_k^{\mu}) = \{\overline{0,1,2,3};\ \overline{4,5,6,7};\ \overline{8,9,10,11};\ \overline{12,13,14,15}\}.$$

Таким образом, $\theta_i^{\nu} + (\theta_j^{\tau} \cdot \theta_k^{\mu}) = (\theta_i^{\nu} + \theta_j^{\tau}) \cdot (\theta_i^{\nu} + \theta_k^{\mu})$.

Дополнением разбиения θ^{ν}_i называется разбиение θ^{γ}_j, такое, что $\theta^{\nu}_i + \theta^{\gamma}_j = \Theta^n$, $\theta^{\nu}_i \cdot \theta^{\gamma}_j = \Theta^0$. В $\hat{\Theta}$ для каждого θ^{ν}_i существует единственное дополнение θ^{γ}_j.

Пример 13. Разбиения из примера 10

$\theta^1_0 = \{\overline{0,1};\ \overline{2,3};\ \overline{4,5};\ \overline{6,7}\}$ и $\theta^2_2 = \{\overline{0,\ 2,\ 4,\ 6};\ \ \overline{1,\ 3,\ 5,\ 7}\}$

дополняют друг друга в $\hat{\Theta}$. Действительно,

$\theta^1_0 + \theta^2_2 = \{\overline{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7}\};\quad \theta^1_0 \cdot \theta^2_2 = \{\overline{0},\ \overline{1},\ \overline{2},\ \overline{3},\ \overline{4},\ \overline{5},\ \overline{6},\ \overline{7}\}$

Графом булевой структуры является *n*-мерный куб . Вершинам куба соответствуют разбиения θ^{ν}_i, а дугам —пары разбиений θ^{ν}_i, θ^{γ}_j, таких, что $\theta^{\nu}_i < \theta^{\gamma}_j$, и не существует такого θ^{μ}_k в $\hat{\Theta}$, что $\theta^{\nu}_i < \theta^{\mu}_k < \theta^{\gamma}_j$. Подмножество разбиений $\Theta^{\nu} \subset \Theta$ $(\nu = 0, \ldots, n)$, содержащее C^{ν}_n разбиений на $2^{n-\nu}$ блоков мощности 2^{ν} каждый, условимся называть *ν*-м *уровнем структуры* $\hat{\Theta}$. Номер уровня, которому принадлежит некоторое разбиение, определяется верхним индексом в обозначении разбиения. Граф структуры в для *n* = 3 изображен на рис. 5

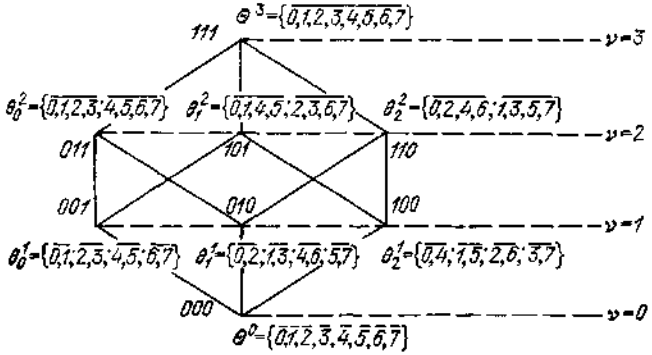


Рис. 5

Вершины графа, соответвующие разбиениям одного уровня, расположены на одной горизонтали.

Разбиения можно нумеровать произвольно, в произвольном порядке можно располагать и блоки в разбиениях. Условимся разбиениям первого уровня всегда присваивать номера, совпадающие с номерами координат, по которым *n*-интервалы в блоках этих разбиений являются соседями. В примере 10 так и сделано: разбиению θ^1_0 соответствует $\{x_0\}$, $\theta^1_1 - \{x_1\}$, $\theta^1_2 - \{x_2\}$. Далее удобно будет вместо определяющего каждое разбиение подмножества координат рассматривать двоичный вектор $g = g^{n-1} \ldots g^0$ с весом *ν* (вес равен числу единиц в *g*). Если координата x_j входит в определяющую

некоторое разбиение совокупность координат, то $g^j = 1$. Так, каждому разбиению первого уровня соответствует g-вектор с весом 1, единственная единица стоит в разряде с тем же номером, который присвоен самому разбиению. Каждому разбиению второго уровня соответствует свой g-вектор с весом 2 и т. д.

Пример 14. Запишем соответствие между разбиениями из примера 10 и g-векторами:

$$\theta^0 = 000,\ \theta_0^1 = 001,\ \theta_1^1 = 010,\ \theta_2^1 = 100,\ \theta_0^2 = 011,$$

$$\theta_1^2 = 101,\ \theta_2^2 = 110,\ \theta^3 = 111.$$

Структура $\hat{\Theta}$ изоморфна булевой структуре $\hat{G} = \langle G, \leqslant \rangle$, которую образует множество G g-векторов. Операции сложения и умножения над разбиениями соответствуют операциям дизъюнкции и конъюнкции над g-векторами. На рис. 5 показаны изоморфные структуры $\hat{\Theta}$ и $\hat{G}$ для $n = 3$.

Чаше всего мы будем иметь дело с разбиениями первого и $(n-1)$-го уровней структуры $\hat{\Theta}$. Условимся для краткости иногда называть разбиения первого уровня *разбиениями на пары,* разбиения $(n-1)$-го уровня *двублочными.*

Свойства совокупности разбиений. Нам будут полезны следующие свойства структуры $\hat{\Theta}$.

1. Из разбиений одного уровня, исключая n-й и нулевой, с помощью операций сложения и умножения можно получить разбиения любого другого уровня. Так, любое разбиение $(\nu+1)$-го уровня можно получить, сложив два разбиения ν-го уровня, а любое разбиение $(\nu-1)$-го уровня, умножив два разбиения ν-го уровня:

$$\theta_q^{\nu+1} = \theta_i^{\nu} + \theta_j^{\nu};\quad \theta_p^{\nu+1} = \theta_i^{\nu} \cdot \theta_j^{\nu}.$$

Пример 15. Разбиения первого уровня получают из разбиений второго уровня при $n=3$, записанных в примере 10, следующим образом:

$$\theta_0^1 = \theta_0^2 \cdot \theta_1^2 = \{\overline{0,1};\ \overline{2,3};\ \overline{4,5};\ \overline{6,7}\};$$

$$\theta_1^1 = \theta_0^2 \cdot \theta_2^2 = \{\overline{0,2};\ \overline{1,3};\ \overline{4,6};\ \overline{5,7}\};$$

$$\theta_2^1 = \theta_1^1 \cdot \theta_2^1 = \{\overline{0,4};\ \overline{1,5};\ \overline{2,6};\ \overline{3,7}\}.$$

Третий уровень при $n=3$ содержит одно тривиальное разбиение

$$\theta^3 = \theta_0^2 + \theta_1^2 = \theta_0^2 + \theta_2^2 = \theta_1^2 + \theta_2^2 = \{\overline{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7}\},$$

где $\theta_0^2 = \{\overline{0,\ 1,\ 2,\ 3};\quad \overline{4,\ 5,\ 6,\ 7}\};\quad \theta_1^2 = \{\overline{0,\ 1,\ 4,\ 5};\quad \overline{2,\ 3,\ 6,\ 7}\};$

$\theta_2^2 = \{\overline{0,\ 2,\ 4,\ 6};\quad \overline{1,\ 3,\ 5,\ 7}\}.$

2. Разбиение, являющееся дополнением для разбиения ν-го уровня, принадлежит $(n-\nu)$-му уровню структуры, т. е.

$\theta_i^\nu + \theta_j^{n-\nu} = \Theta^n$, $0_i^\nu \cdot \theta_j^{n-\nu} = \Theta^0$, и g-векторы, соответствующие разбиениям θ_i^ν и $\theta_j^{n-\nu}$:

$$g_i \vee g_j = \underbrace{1 \ldots 1}_{n}; \quad g_i \wedge g_j = \underbrace{0 \ldots 0}_{n}. \quad \textbf{(4)}$$

Дополнениями для разбиений первого уровня являются разбиения $(n-1)$-го уровня. Так, разбиения θ_0^1 и θ_2^2 из примера 10 дополняют друг друга, их g-векторы 001 и 110 также являются дополнениями друг для друга в смысле выражений (4).

В следующих трех пунктах даны соотношения, связывающие разбиение первого уровня с разбиениями любых других уровней, позволяющие «взглянуть» на структуру, находясь на первом уровне.

3. Любое разбиение ν-го уровня $(\nu \geqslant 0)$ равно сумме ν разбиений первого уровня:

$$0_i^\nu = \sum_{k=1}^{\nu} \theta_{i_k}^1, \quad \textbf{(5)}$$

где $i_1, \ldots, i_\nu$ — одно из сочетаний из $\{0, \ldots, n-1\}$ по ν чисел.

Пример 16. Разбиения первого уровня для n=4:

$\theta_0^1 = \{\overline{0,1};\ \overline{2,3};\ \overline{4,5};\ \overline{6,7};\ \overline{8,9};\ \overline{10,11};\ \overline{12,13};\ \overline{14,15}\}$;

$\theta_1^1 = \{\overline{0,2};\ \overline{1,3};\ \overline{4,6};\ \overline{5,7};\ \overline{8,10};\ \overline{9,11};\ \overline{12,14};\ \overline{13,15}\}$;

$\theta_2^1 = \{\overline{0,4};\ \overline{1,5};\ \overline{2,6};\ \overline{3,7};\ \overline{8,12};\ \overline{9,13};\ \overline{10,14};\ \overline{11,15}\}$;

$\theta_3^1 = \{\overline{0,8};\ \overline{1,9};\ \overline{2,10};\ \overline{3,11};\ \overline{4,12};\ \overline{5,13};\ \overline{6,14};\ \overline{7,15}\}$.

Разбиения третьего уровня выражаются через разбиения первого уровня следующим образом:

$\Theta_0^3 = \theta_0^1 + \theta_1^1 + \theta_2^1 = \{\overline{0, 1, 2, 3, 4, 5, 6, 7};\ \overline{8, 9, 10, 11, 12, 13, 14, 15}\}$;

$\Theta_1^3 = \theta_0^1 + \theta_1^1 + \theta_3^1 = \{\overline{0, 1, 2, 3, 8, 9, 10, 11};\ \overline{4, 5, 6, 7, 12, 13, 14, 15}\}$;

$\Theta_2^3 = \theta_0^1 + \theta_2^1 + \theta_3^1 = \{\overline{0, 1, 4, 5, 8, 9, 12, 13};\ \overline{2, 3, 6, 7, 10, 11, 14, 15}\}$;

$\Theta_3^3 = \theta_1^1 + \theta_2^1 + \theta_3^1 = \{\overline{0, 2, 4, 6, 8, 10, 12, 14};\ \overline{1, 3, 5, 7, 9, 11, 13, 15}\}$.

Заметим, что номера разбиений первого уровня, сумма которых равна 0_i^ν, определяются по g-вектору, соответствующему 0_i^ν, и равны номерам единичных компонентов в g-векторе. И действительно, из изоморфизма структур $\hat{\Theta}$ и $\hat{G}$ следует, что поскольку разбиение 0_i^ν равно сумме разбиений $\theta_{i_1}^1, \ldots, \theta_{i_\nu}^1$, постольку g-вектор разбиения θ_i^ν равен дизъюнкции g-векторов разбиений $\theta_{i_1}^1, \ldots, \theta_{i_\nu}^1$. Это значит, что в g-векторе разбиения θ_i^ν единицы стоят в компонентах с номерами $i_1, \ldots, i_\nu$. Так, в примере 16 $\theta_0^3 = \theta_0^1 + \theta_1^1 + \theta_2^1$ и

$g=0111$, $\theta_1^3=\theta_0^1+\theta_1^1+\theta_3^1$ и $g=1011$, $\theta_2^3=\theta_0^1+\theta_2^1+\theta_3^1$ и $g=1101$, $\theta_3^3=\theta_1^1+\theta_2^1+\theta_3^1$ и $g=1110$.

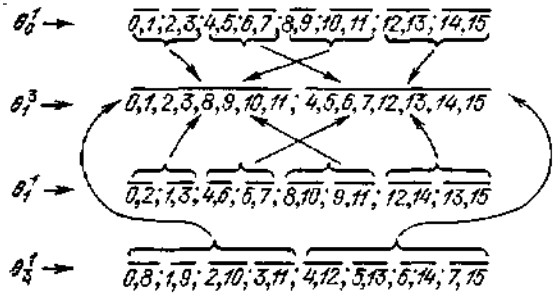
Каждое разбиение *(n*—1)-го уровня θ_i^{n-1} равно сумме всех разбиений первого уровня, за исключением одного θ_j^1, являющегося дополнением для θ_i^{n-1}:

$$\theta_i^{n-1}=\sum_{\substack{k=0,\\ k\neq j}}^{n-1}\theta_k^1.$$

Очевидно, что тривиальное разбиение Θ^n равно сумме всех разбиений первого уровня:

$$\Theta^n=\sum_{k=0}^{n-1}\theta_k^1. \quad \textbf{(6)}$$

4. Из (5) следует, что каждый блок $B_j\in\theta_i^v$ ($|B_j|=2^v$) может быть получен путем объединения 2^{v-1} блоков из любого разбиения $\theta_{i_k}^1$, вошедшего в сумму $\theta_i^v=\theta_{i_1}^1+\ldots+\theta_{i_v}^1$. Обратимся к примеру 16. Блок $\overline{0,1,2,3,8,9,10,11}\in\theta_1^3$ получается как объединение четырех блоков одного из разбиений θ_0^1, θ_1^1, θ_3^1. Изобразим это в виде диаграммы



Подмножество блоков разбиения $\theta_{i_k}^1$, таких, что их объединение равно блоку $B_j\in\theta_i^v$, назовем *коллекцией* $C_j^i{}_k$, порождаемой в разбиении $\theta_{i_k}^1$ блоком B_j. Так, блок $B_0=\overline{0,1,2,3,8,9,10,11}$ из разбиения $\theta^3{}_1$ порождает в разбиениях θ_1^0, θ_1^1, θ_3^1 следующие коллекции: $C_0^0=\{\overline{0,1};\ \overline{2,3};\ \overline{8,9};\ \overline{10,11}\}$; $C_0^1=\{\overline{0,2};\ \overline{1,3};\ \overline{8,10};\ \overline{9,11}\}$; $C_0^3=\{\overline{0,8};\ \overline{1,9};\ \overline{2,10};\ \overline{3,11}\}$, блок $B_1=\overline{4,5,6,7,12,13,14,15}$ из разбиения $\theta^3{}_1$ — следующие коллекции: $C_1^0=\{\overline{4,5};\ \overline{6,7};\ \overline{12,13};\ \overline{14,15}\}$; $C_1^1=\{\overline{4,6};\ \overline{5,7};\ \overline{12,14};\ \overline{13,15}\}$; $C_1^3=$

$= \{\overline{4,12};\ \overline{5,13};\ \overline{6,14};\ \overline{7,15}\}$ (см. диаграмму). Очевидно, что в каждом разбиении первого уровня, входящем в сумму $\theta^{\nu}_{i} = \theta^{1}_{i_1} + \ldots + \theta^{1}_{i_\nu}$, коллекций ровно столько же, сколько блоков в разбиении θ^{ν}_{i}, а именно 2^{n-1}. Каждое разбиение из $\{\theta^{1}_{i_1}, \ldots, \theta^{1}_{i_\nu}\}$ можно представить как совокупность коллекций

$$\theta^{1}_{i_k} = \{C^{i}_{0k},\ C^{i}_{1k},\ \ldots,\ C^{i}_{2^{n-\nu}-1k}\},\quad k = 1,\ \ldots,\ \nu.$$

Так, разбиения θ^{1}_{1}, θ^{1}_{2}, θ^{1}_{3} (см. пример 16) состоят из следующих коллекций, порожденных в этих разбиениях двублочным разбиением $\theta^{3}_{3} = \theta^{1}_{1} + \theta^{1}_{2} + \theta^{1}_{3}$:

$$\theta^{1}_{1} = \{C^{1}_{0} = \{\overline{0,2};\ \overline{4,6};\ \overline{8,10};\ \overline{12,14}\},\ C^{1}_{1} = \{\overline{1,3};\ \overline{5,7};\ \overline{9,11};\ \overline{13,15}\}\};$$

$$\theta^{1}_{2} = \{C^{2}_{0} = \{\overline{0,4};\ \overline{2,6};\ \overline{8,12};\ \overline{10,14}\},\ C^{2}_{1} = \{\overline{1,5};\ \overline{3,7};\ \overline{9,13};\ \overline{11,15}\}\};$$

$$\theta^{1}_{3} = \{C^{3}_{0} = \{\overline{0,8};\ \overline{2,10};\ \overline{4,12};\ \overline{6,14}\},\ C^{3}_{1} = \{\overline{1,9};\ \overline{3,11};\ \overline{5,13};\ \overline{7,15}\}\}.$$

5. Каждый блок $B_j = \theta^{\nu}_{i}$ можно считать единицей некоторой структуры $\hat{\Theta}(B_j)$ с числом уровней $\nu+1$, если рассматривать совокупность разбиений булева подпространства B_j. Тогда коллекция $C^{i}_{jk} \subset \theta^{1}_{i_k}$ является разбиением первого уровня структуры $\hat{\Theta}(B_j)$, а множество коллекций $\{C^{i_1}_{j},\ C^{i_2}_{j},\ \ldots,\ C^{i_k}_{j},\ \ldots,\ C^{i_\nu}_{j}\}$, порождаемых блоком B_j в каждом $\theta^{1}_{i_k}$ $(k = 1,\ \ldots,\ \nu)$,—множеством разбиений первого уровня структуры $\hat{\Theta}(B_j)$. Так, пусть блок $B_0 = \overline{0,1,4,5,8,9,12,13}$ разбиения θ^{3}_{2} (см. пример 16) является единицей структуры $\hat{\Theta}(B_0)$ с четырьмя уровнями (размерность булева подпространства B_0 равна трем). Первым уровнем структуры $\hat{\Theta}(B_0)$ является совокупность коллекций $\{C^{1}_{0},\ C^{2}_{0},\ C^{3}_{0}\}$, приведенных в п. 4.

Граф структуры $\hat{\Theta}(B_0)$ приведен на рис. 6.

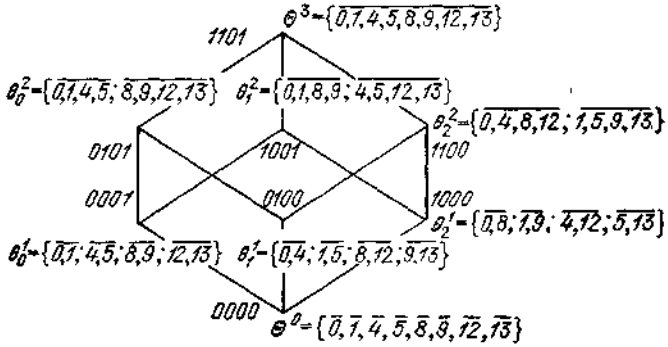


Рис. 6

Все, что сказано в пп. 3—5 о разбиениях первого уровня, можно двойственно сформулировать относительно разбиений *(n*—1)-го уровня. При этом следует заменить разбиения первого уровня их дополнениями, т. е. разбиениями *(n*—1)-го уровня, операцию сложения операцией умножения, число *v* дополнением до *n,* т. е. на *n- v,* и т. д. Сделаем это.

6. Каждое разбиение *v*-го уровня, за исключением *n,* равно произведению *n*—*v* разбиений *(n*—1)-го уровня:

$$\theta_i^v = \prod_{k=1}^{n-v} \theta_{i_k}^{n-1}. \qquad (7)$$

В произведение входятвсе разбиения (*n*—1)-го уровня, кроме тех, которые являются дополнениями для разбиений первого уровня, входящих в сумму (5).

Пример 17. Возьмем одно из разбиений второго уровня структуры $\hat{\Theta}$ при *n* =4, например

$$\theta_0^2 = \{\overline{0,1,2,3};\ \overline{4,5,6,7};\ \overline{8,9,10,11};\ \overline{12,13,14,15}\}.$$

Оно равно сумме разбиений θ_0^1 и θ_1^1 (см. пример 16): $\theta_0^2 = \theta_0^1 + \theta_1^1$, а его вектор $g = 0001 \vee 0010 = 0011$. Дополнением для $\theta^1{}_0$ является разбиение $\theta^3{}_3$ с g=1110, для $\theta^1{}_1$ — разбиение θ_2^3 с g= 1101 (см. пример 16) Вычислим произведение разбиений третьего уровня, исключив из него θ_3^3 и θ_2^3:

$$\theta_0^3 \cdot \theta_1^3 = \{\overline{0,1,2,3};\ \overline{4,5,6,7};\ \overline{8\ 9,10,11};\ \overline{12,13,14,15}\},$$

для соответствующих *g*-векторов $0111 \wedge 1011 = 0011$. Вычислим произведение отброшенных разбиений третьего уровня

$$\theta_2^3 \cdot \theta_3^3 = \{\overline{0,4,8,12};\ \overline{1,5,9,13};\ \overline{2,6,10,14};\ \overline{3,7,11,15}\},$$

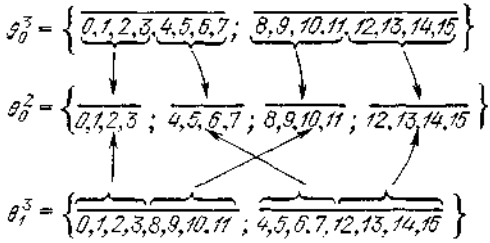
для соответствующих g-векторов $1101 \wedge 1110 = 1100$. θ_2^3 θ_3^3 есть разбиение второго уровня, дополняющее $\theta^2{}_0$. Действительно, g-вектор полученного разбиения 1100 и g-вектор разбиения $\theta^2{}_0$ 0011 дополняют друг друга: $1100 \vee 0011 = 1111$, $1100 \wedge 0011 = 0000$. Заметим, что дополняющие друг друга разбиения оказались на одном уровне, это не противоречит п. 2, так как в данном случае $n = 4$, а $v = 2$, поэтому $n—v=2$

В заключение этого пункта приведем выражение, двойственное (6):

$$\Theta^0 = \prod_{k=0}^{n-1} \theta_k^{n-1}. \quad (8)$$

7. Из (7) следует, что каждый блок разбиения $\theta^v{}_i$, равного произведению двублочных разбиений $\{\theta_{i_1}^{n-1}, \ldots, \theta_{i_{n-v}}^{n-1}\}$, целиком входит в один из блоков каждого разбиения $\theta_{i_k}^{n-1}$ как подблок. Такие подблоки блоков $B_l \in \theta_{i_k}^{n-1}$ $(l=0,\ 1;\ k=1,\ \ldots,\ n—v)$ будем называть *секциями* и обозначать $E_j^{i_k}$ $(j = 0, \ldots, 2^{n-v} - 1)$, Хотя секция есть не что иное, как блок $B_j \in \theta_i^v$, будем использовать обозначение $E_j^{i_k}$, чтобы подчеркнуть, что имеется в виду часть блока разбиения (n—1)-го уровня, а не блок разбиения v-го уровня. Понятие «секция» двойственно понятию «коллекция». Действительно, в коллекции мы собираем определенные блоки разбиений первого уровня, на секции расчленяем блоки разбиений двойственного ему (n—1)-го уровня.

Пример 18. Возьмем разбиение $\theta^2{}_0$ из примера 17, которое равно произведению двух двублочных разбиений (при $n = 4$ (n—1)-й уровень — это третий) θ_0^3 и θ_1^3. Секции в блоках разбиений θ_0^3 и θ_1^3 показаны на диаграмме



8. Разбиение θ_i^v можно считать нулем подструктуры $\hat{\Theta}' \subset \hat{\Theta}$.

Единицей $\hat{\Theta}'$ является единица структуры $\Theta—\Theta^n$.

Структура $\hat{\Theta}'$ имеет $r = n$—v+1 уровней. Множество двублочных разбиений $\{\theta_{i_1}^{n-1}, \ldots, \theta_{i_{n-v}}^{n-1}\}$ является (r—1)-м уровнем $\hat{\Theta}'$. На рис. 5 в

качестве $\hat{\Theta}'$ можно рассматривать, например, подструктуру $\{\theta_1^1, \theta_0^2, \theta_2^2 \theta^3\}$, имеющую три уровня, в которой разбиение θ_1^1 является нулем, θ^3 — единицей.

6.2.2. Анализ структуры

Усеченная структура, порожденная булевой функцией. Пусть в пространстве $I^{(n)} = \{a_0, \ldots, a_{2^n-1}\}$ задана полностью определенная булева функция $f(x_0, \ldots, x_{n-1})$, при этом пусть I^0— множество точек в $I^{(n)}$, на которых $f=0$, $I^1 = I^{(n)} \setminus I^0$ —множество точек в $I^{(n)}$, на которых *f=1*. Задание функции делит множество всех интервалов булева пространства или множество блоков всех разбиений структуры $\hat{\Theta}$ на два непересекающихся подмножества $Б^0$ и $Б^1$. Подмножество $Б^0$ составляют блоки, и каждом из которых есть хотя бы один *n*-интервал из области I^0, $Б^1$ — блоки, целиком содержащиеся в области I^1. Проанализируем, что происходит со структурой, если из ее разбиений удалить блоки, принадлежащие $Б^0$, и посмотрим, какими средствами можно представить «осколки» структуры. Удалим из разбиений структуры $\hat{\Theta}$ блоки, входящие в $Б^0$. При этом некоторая часть разбиений отбрасывается, а от других остаются «осколки» в виде разбиений некоторых подмножеств множества I^1. Обозначим такие разбиения через $\Lambda = \{\Lambda_1, \ldots, \Lambda_u\}$.

Пример 19. Пусть функция $f(x_0, x_1, x_2)$ имеет такие области значений: $I^1 = \{0,1,4,5,6,7\}$, $I^0=\{2,3\}$. Удалив из разбиений, приведенных в примере 10, блоки, содержащие *n*-интервалы 2 и 3, получим.

$$\Lambda=\{\Lambda_1, \ldots, \Lambda_6\}, \quad \Lambda_1=\{\overline{0,1}, \overline{4,5}, \overline{6,7}\}, \quad \Lambda_2=\{\overline{0,1}; \ \overline{4,5}; \ \overline{6,7}\},$$

$$\Lambda_3=\{\overline{4,6}; \ \overline{5,7}\}, \quad \Lambda_4=\{\overline{0,4}; \ \overline{1,5}\}, \quad \Lambda_5=\{\overline{4,5,6,7}\}, \quad \Lambda_6=\{\overline{0,1,4,5}\}.$$

Из восьми разбиений структуры $\hat{\Theta}$ при *n*=3 осталось только шесть разбиений подмножеств множества I^1.

Множество $\Lambda = \{\Lambda_j\}$ назовем *усеченной структурой,* порожденной булевой функцией *f*. Каждому $\Lambda_j \in \Lambda$ соответствует g-вектор, тот самый, что был поставлен в соответствие разбиению θ_t^s, из которого было получено Λ_j. Частичный порядок, имевший место в множестве $\{\theta_i^k\}$, в усеченной структуре $\{\Lambda_j\}$ не выполняется, но сохраняется для g-векторов, соответствующих разбиениям Λ_j ($g_i < g_j$ $(i \neq j)$, если $g_i^k \leqslant g_j^k$ для всех *k* заметим, что среди g-векторов нет равных, поэтому хотя бы для одного разряда обязательно выполняется неравенство $g_i^k < g_j^k$).

Графически усеченную структуру Λ будем изображать в виде усеченного куба, а частичный порядок, который символизируется кубом, относить только к g-векторам. На рис. 7 изображен граф усеченной структуры, порожденной $f(x_0, x_1, x_2)$ с $I^0=\{2,3\}$ и $I^1 = I^{(3)} \setminus I^0$ (см. пример 19).

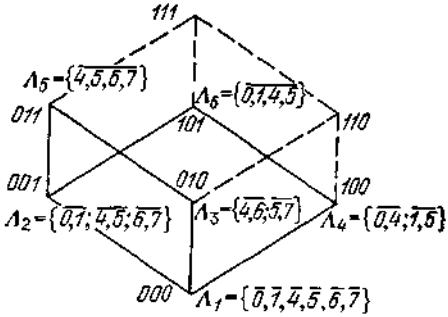


Рис. 7

Из множества Б[1] можно выделить множество $Б^1_{max}$ максимальных блоков: $B_j \in Б^1_{max}$, если в Б[1] не найдется блока B_k, такого, что $B_j \subset B_k$. Блоки $B_j \in Б^1_{max}$ являются максимальными интервалами в области I^1. Выделим из $Б^1_{max}$ множество блоков, содержащих некоторый *n*-интервал $a_h \in I^1$, выделенное множество обозначим через $Б^1_{max}(a_h)$. Обозначим через $I^1(a_h)$ подмножество *n*-интервалов, равное объединению всех блоков в $Б^1_{max}(a_h): I^1(a_h) = \bigcup_{B_j \in Б^1_{max}(a_h)} B_j$. Так, множество $Б^1_{max}$ (6) для усеченной структуры из примера 19 состоит из одного блока $\overline{4,5,6,7} \in \Lambda_5$, $I^1(6) = \{4,5,6,7\}$.

Усеченная структура, порожденная *n*-интервалом. Рассмотрим усеченную структуру $\Lambda(a_h) = \{\Lambda_j(a_h)\}$, порождаемую *n*-интервалом a_h. Каждое разбиение $\Lambda_j(a_h)$ получено из Λ_j путем удаления тех блоков, которые не содержатся в множестве $I^1(a_h)$. Так, усеченная структура Λ(4), образованная из Λ (см. пример 19), совпадает с самой усеченной структурой Λ. Граф усеченной пруктуры Λ (6), образованной из той же Λ, изображен на рис. 8.

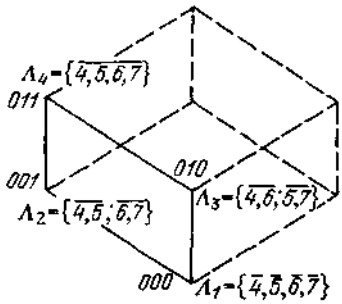


Рис. 8

По отношению к $a_h \in I^1$ все множество g-векторов G делится на два непересекающихся подмножества: одно из них $G'(a_h)$ содержит g-векторы, соответствующие разбиениям $\Lambda_j(a_h)$, другое есть $G''(a_h)$ множество всех остальных g-векторов. Так, усеченной структуре Λ (5) (см. рис. 7) соответствует множество *G'* (5) = {000,001,010, 100,011,101}, при этом G"(5) = {110,111}. Для усеченной структуры Λ(6) (см. рис. 8) G'(6) = {000,001, 010,011}, G"(6) = {100,101,110,111}. В множестве $G'(a_h)$ можно выделить подмножество $G'_{\max}(a_h)$ максимальных g-векторов: g_i- является максимальным в $G'(a_h)$, если в $G'(a_h)$ не найдется g_j, такого что $g_i < g_j$. Так, в G'(5) имеется два максимальных вектора: $G'_{\max}$ (5) = {011,101}, в G'(6)— только один: $G'_{\max}$ (6) = {011}. Каждый вектор $g_t \in G'_{\max}(a_h)$ соответствует максимальному в области I^1 интервалу, принадлежащему множеству $B^1_{\max}(a_h)$. Векторы 011 и 101 из $G'_{\max}$ (5) соответствуют интервалам $\overline{\{4,5,6,7\}}$ и $\overline{\{0,1,4,5\}}$ (см. пример 19 и рис. 7). Поскольку каждый вектор $g_t \in G'_{\max}(a_h)$ определяет совокупность координат $\{x_{j_1}, \ldots, x_{j_k}\}$ $(g^{j_1}_t = \ldots = g^{j_k}_t = 1)$, по которым *n*-интервалы объединяются в блок $B_t \in B^1_{\max}(a_h)$, то, зная только множество $G'_{\max}(a_h)$, можно записать все интервалы, максимальные в области I^1. Для этого в двоичном наборе a_h компоненты, соответствующие единицам в $g_t \in G'_{\max}(a_h)$, нужно заменить на звездочки $(*)$. Так, в примере 19 $G'_{\max}$ (4) = {011,101}, интервалы, максимальные в I^1(4) = {0,1,4,5,6,7}, представляются троичными векторами $1{*}{*}$ и ${*}0{*}$.

В множестве $G(a_h)$ можно выделить подмножество $G''_{\min}(a_h)$ минимальных g-векторов: $g_t \in G''_{\min}(a_h)$, если в $G''(a_h)$ не найдется g_j. такого, что $g_t > g_j$.

Заметим, что вершины *n*-мерного куба, соответствующие минимальным g-векторам, являются соседними вершинам, представляющим g-векторы из $G'(a_h)$. Это значит, что стоит в векторе $g_i \in G''_{\min}(a_h)$ заменить одну единицу нулем, как получится g-вектор из множества $G'(a_h)$. Действительно, в примере 19 $G''_{\min}(4) = \{110\}$, заменив одну из единиц нулем, получим g-векторы 100 или 010 из G'(4). Таким образом, каждый $g_i \in G''_{\min}(a_h)$ определяет *запретную для a_h совокупность координат,* Причем минимальную совокупность. Это значит, что стоит заменить в g_i одну единицу нулем — и совокупность перестанет быть запретной. А запретна она в том смысле, что интервал, определенный на этой совокупности координат и содержащий a_h, принадлежит множеству $Б^0$. Такой же запретгой для a_h является совокупность координат, отвечающая любому $g_k > g_i$ $g_i \in G'_{\min}$ (a_h). Для a_h =4 из примера 19 минимальной запретной спинкупностью координат является $\{x_1, x_2\}$, определяемая g-вектором110. Интервал {0,2,4,6}, или в троичном представлении $* * 0$, принадлежит множеству $Б^0$, так кик включает элемент 2, входящий в I^0.

Множества g-векторов $G'_{\max}(a_h)$ и $G''_{\min}(a_h)$ связаны следующим образом. Из множества $G''_{\min}(a_h)$ образуем булеву матрицу Q, записав каждый g-вектор в строку. Для такой матрицы можно найти множество столбцовых безызбыточных покрыпий. Каждое такое покрытие может быть представлено в виде двоичного вектора *y* той же длины, что и g-вектор. Единицы в *y* соответствуют столбцам матрицы Q, вошедшим в покрытие. Таким образом, все множество безызбыточных столбцовых покрытий матрицы *Q* будет представлено множеством двоичных, векторов $\{y_i\}$. Оказывается, что двоичный вектор $\bar{g}_i$, полученный из $g_i \in G'_{\max}(a_h)$ путем инвертирования всех его компонентов, является одним из векторов множества $\{y_i\}$. Действительно, в соответствии с содержательным смыслом *g*-векторов из $G'_{\max}(a_h)$ вектор $\bar{g}_i$ дает такую совокупность координат, которая содержит хотя бы по одной координате из каждой запретной совокупности, т. е. для каждого $g_j \in G''_{\min}(a_h)$ вектор $\bar{g}_i \wedge g_j$ имеет хотя бы одну единицу. Отсюда следует, что $\bar{g}_i$ соответствует одному из покрытий матрицы *Q* столбцами. То, что это покрытие безызбыточно, легко показать от противного. Предположим, что $\bar{g}_i$ соответствует небезызбыточному покрытию матрицы *Q*. Это значит, что в $\bar{g}_i$ можно

заменить нулем хотя бы одну единицу. После такой замены и инвертирования компонентов мы получим вектор $g_i' > g_i$. Но вектор g'_i не может принадлежать множеству $G_{max}(a_h)$. Отсюда следует, что $\overline{g_i}$ соответствует безызбыточному покрытию.

Таким образом, двоичные векторы, соответствующие всем безызбыточным столбцовым покрытиям матрицы *Q*, полученной из $G''_{min}(a_h)$, будучи инвертированы, составят множество $G'_{max}(a_h)$. Для примера 19 мы получили такие подмножества g-векторов: $G'_{max}(4) = \{011, 101\}$ и $G''_{min}(4) = \{110\}$. В этом случае матрица Q имеет одну строку ||110|| и два безызбыточных покрытия, соответствующие векторам 100 и 010. Инвертируя эти векторы, получаем {011,101}.

6.2.3. Синтез структур

Задачи кодирования как задачи синтеза структуры.
Пусть множество состояний $S = \{s_0, \ldots, s_{2^n-1}\}$ закодировано *n*-разрядным двоичным кодом $C_S = \{s_j \to x_{n-1} \ldots x_0\}$, $x_i \in \{0,1\}$. Каждый *i*-й разряд кода определяет разбиение множества S на два блока: $T_i = \{S^0_i, S^1_i\}$. В блок S^0_i входят те элементы s_j, которым присвоены кодирующие слова с нулем в *i*-м разряде, в блок S^1_i —такие s_j, которым присвоены кодирующие слова с единицей в *i*-м разряде. Таким образом, код C_S порождает множество двублочных разбиений множества $S: T = \{T_0, \ldots, T_{n-1}\}$.

Пример 20. Пусть состояния {*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*} закодированы таким образом:

$$C = \begin{array}{c} \begin{matrix} x_2 & x_1 & x_0 \end{matrix} \\ \left[\begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{array}\right] \end{array} \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{matrix}$$

Трехразрядный код порождает три двублочкых разбиения множества состояний:

$$T_0 = \{\overline{c,\ d,\ g,\ h};\ \ \overline{a,\ b,\ e,\ f}\};$$
$$T_1 = \{\overline{a,\ c,\ d,\ e};\ \ \overline{b,\ f,\ g,\ h}\};$$
$$T_2 = \{\overline{b,\ c,\ e,\ g};\ \ \overline{a,\ d,\ f,\ h}\}.$$

Множество T разбиений можно рассматривать как (n—1)-й уровень булевой структуры $\hat{\Theta}$, которая соответствует коду C_S и образована разбиениями множества состояний S, равными всевозможным произведениям двублочных разбиений $\{T_0, \ldots, T_{n-1}\}$.

Пример 21. Запишем все разбиения булевой структуры $\hat{\Theta}$ для n= 3, разбиения второго уровня которой даны в примере 20

$$\Theta^0 = \{\overline{a},\ \overline{b},\ \overline{c},\ \overline{d},\ \overline{e},\ \overline{f},\ \overline{g},\ \overline{h}\};$$
$$\theta_0^1 = T_0 \cdot T_1 = \{\overline{c,d};\ \overline{a,e};\ \overline{g,h};\ \overline{b,f}\},$$
$$\theta_1^1 = T_0 \cdot T_2 = \{\overline{c,g};\ \overline{b,e};\ \overline{d,h};\ \overline{a,f}\},$$
$$\theta_2^1 = T_1 \cdot T_2 = \{\overline{c,e};\ \overline{b,g};\ \overline{a,d};\ \overline{f,h}\};$$
$$\theta_0^2 = T_0 = \{\overline{c,\ d,\ g,\ h};\ \ \overline{a,\ b,\ e,\ f}\},$$
$$\theta_1^2 = T_1 = \{\overline{a,\ c,\ d,\ e};\ \ \overline{b,\ f,\ g,\ h}\},$$
$$\theta_2^2 = T_2 = \{\overline{b,\ c,\ e,\ g};\ \ \overline{a,\ d,\ f,\ h}\};$$
$$\Theta^3 = \{\overline{a,\ b,\ c,\ d,\ e,\ f,\ g,\ h}\}.$$

Таким образом, задачу двоичного кодирования множества состояний можно сформулировать как задачу синтеза булевой структуры на множестве состояний с учетом тех требований, которые предъявляются к коду. Как следует из свойства 1 § 6.2.1, булеву структуру можно считать построенной, если построены разбиения одного ее уровня (нетривиального). Это значит, что задачу синтеза структуры $\hat{\Theta}$ можно свести к построению разбиений одного уровня. Естественно для этого выбрать уровень простой по числу разбиений и удобный для того, чтобы учесть требования к коду. В булевой структуре $\hat{\Theta}$ два простых нетривиальных уровня: первый — множество разбиений на блоки по два элемента в каждом (такие блоки удобно называть парами и разбиения на такие блоки — разбиениями на пары) и (n—1)-й — множество двублочных разбиений. Эти уровни содержат по n разбиений. В зависимости от того, что требуется от кода C_S — склеивать определенные подмножества состояний в интервалы (экономичное кодирование) или развязывать пары подмножеств (противогоночное кодирование)—можно строить первый уровень, наглядно отображающий соседство состояний в булевом пространстве,

или (*n*—1)-й, т. е. двублочные разбиения, непосредственно развязывающие пары подмножеств.

Поясним, почему удобно при экономичном кодировании строить первый уровень, а при прогивогоночном *(n*—1)-й. Задача экономичного кодирования состоит в построении на множестве состояний такой структуры $\hat{\Theta}$, которая дает минимальное покрытие совокупности *K*-множеств блоками разбиений структуры, являющееся наименее мощным по сравнению с минимальным покрытием блоками разбиений любой другой структуры. Чтобы получить точное решение задачи, необходимо построение разбиений разных уровней и перебор различных вариантов разбиений. Не ставя цели найти наилучший по экономичности код, предлагаем использовать следующий эвристический прием, естественный для решения близкой задачи кодирования: соседние коды присваиваются прежде всего парам состояний, которые встречаются в большем числе *K*-множеств . Поскольку такой прием связан с некоторым списком пар состояний, то для синтеза структуры $\hat{\Theta}$ целесообразно взять первый уровень, т. е. строить разбиения на пары, учитывая при этом исходный список пар состояний. Очевидно, что если разбиения первого уровня составлены так, чтобы в них входило как можно больше пар состояний из исходного списка, то эти пары, объединенные на более высоких уровнях в четверки, восьмерки и т. д., образуют крупные блоки, целиком входящие в *K*-множества.

Противогоночное кодирование непосредственным образом связано с построением *(n*—1)-го уровня. Действительно, необходимое и достаточное условие отсутствия опасных состязаний между элементами памяти в асинхронном автомате в терминах теоретико-структурного подхода формулируется так.

Для любых двух состязающихся переходов $(s_l \to s_k,\ s_q \to s_r)$ среди двублочных разбиений $\{\theta_0^{n-1}, \ldots, \theta_{n-1}^{n-1}\}$, соответствующих *n*-разрядному двоичному коду C_s, есть хотя бы одно такое, в котором состязающиеся переходы находятся в разных блоках, т. е $\{\overline{s_l, s_k};\ \overline{s_q, s_r}\} \leqslant \theta_j^{n-1}$. Это значит, что имеется хотя бы один разряд кода (например, *j*-й), который принимает одно значение для состояний s_l и s_k и противоположное для состояний s_q и s_r. Иначе говоря, переходы $s_l \to s_k$ и $s_q \to s_r$ развязаны по *j*-му разряду кода как развязаны блоки разбиения θ_j^{n-1}. Например пусть s_l = 001, s_k=*100*, s_q=*111*, s_r =011, переходы 001→100 и 111→011 развязаны по первому разряду (напомним что разряды кода нумеруются справа налево начиная с

нуля), в состояниях первого перехода x_1=0. второго x_1 =1, в разбиении $0_2^2 = \overline{\{000,010,100,110;\ 001,011,\ 101,111\}}$ (см. пример 9) переходы находятся в разных блоках: s_l, s_k в B_0 s_r, s_r в B_1.

Достаточное условие отсутствия опасных состязаний , накладываемое на пары состязающихся *k*-множеств: $\{k_t \quad k_l\} \leqslant \theta_l^{n-1}$. Это значит, что два подмножества состояний k_i и k_l развязаны по *j*-му разряду. Таким образом, задачи экономичного и противогоночного кодирования в терминах теоретико-структурного подхода сформулированы нами как задачи построения двойственных уровней булевой структуры первого и (*n*-1)-го, и естественно, что способы решения этих задач двойственны.

Теоремы о синтезе струкгуры. Для обоснования процедур синтеза структуры на множестве состоянии докажем несколько утверждений

Два блока разбиения первого уровня $B_i,\ B_j \in \theta_p^1$ сопряжены по разбиению θ^1_q 0;, если

$$B_i \cup B_j \in \theta_p^1 + \theta_q^1, \quad (9)$$

т.е. блоки $B_i = \overline{a,\ b}$ и $B_j = \overline{c,\ d}$, принадлежащие θ^1_p, сопряжены по θ^1_q, если в θ^1_q есть блоки $B_k = \overline{a,\ c}$, $B_l = \overline{b,\ d}$ или $B_k = \overline{a,\ d}$, $B_l = \overline{b,\ c}$. Для сопряженных блоков справедливо

$$\{B_i,\ B_j\} \cdot \{B_k,\ B_l\} \subset \theta^0. \quad (10)$$

Так, $\{\overline{a,b};\ \overline{c,d}\} \cdot \{\overline{a,d};\ \overline{b,c}\} = \{\bar{a},\ \bar{b},\ \bar{c},\ \bar{d}\}$. Сопряженные блоки обозначим $B_j = B_i^{\#}(\theta_q^1)$; это означает, блок B_j сопряжен с блоком B_i, по разбиению θ^1_q. Очевидно, что то же самое означает и запись $B_i = B_j^{\#}(\theta_q^1)$. Заметим, что каждый блок $B_i \in \theta_q^1$ сопряжен с самим собой по своему же разбиению:

$$B_i = B_i^{\#}(\theta_q^1). \quad (11)$$

Можно говорить о совокупности $\{B_{i_1}, \ldots, B_{i_\mu}\}$ блоков разбиения $\theta^1_{p_m}$, сопряженных по подмножеству разбиений $\{\theta^1_{p_1}, \ldots, \theta^1_{p_\nu}\}$. Это значит, что в совокупности для каждого B_{i_k} найдется блок B_{i_l}, сопряженный с

$$B_{i_k} \text{ по } \theta^1_{p_m} \quad (m = 1, \ldots, \nu).$$

Пример 22. Пусть имеется подмножество $\{\theta_0^1,\ \theta_1^1,\ \theta_2^1\}$ разбиений первого уровня (*n* = 4):

$$\theta_0^1 = \{\overline{a,b};\ \overline{c,d};\ \overline{e,f};\ \overline{g,h};\ \overline{i,j};\ \overline{k,l};\ \overline{m,n};\ \overline{o,p}\},$$

$$\theta_1^1 = \{\overline{a,c};\ \overline{b,d};\ \overline{e,g};\ \overline{f,h};\ \overline{i,k};\ \overline{j,l};\ \overline{m,o};\ \overline{n,p}\},$$

$$\theta_2^1 = \{\overline{a,e};\ \overline{b,f};\ \overline{c,g};\ \overline{d,h};\ \overline{i,m};\ \overline{j,n};\ \overline{k,o};\ \overline{l,p}\}.$$

Блоки совокупности $\{\overline{a,e};\ \overline{b,f};\ \overline{c,g};\ \overline{d,h}\} \subset \theta_2^1$ сопряжены по $\{\theta_0^1, \theta_1^1, \theta_2^1\}$. Действительно, $(a,e) = (b,f)^{\#}(\theta_0^1)$, $(a,e) = (c,g)^{\#}(\theta_1^1)$, $(c,g) = (d,h)^{\#}(\theta_0^1)$, $(d,h) = (b,f)^{\#}(\theta_1^1)$, и каждый блок сопряжен с самим собой по θ_2^1 (см. (11)).

Как следует из определения понятия коллекции, блоки $B_i,\ B_j \in \theta_p^1$, сопряженные по θ_q^1, составляют коллекцию, порожденную в θ_p^1 разбиением второго уровня $\theta_p^1 + \theta_q^1$. Действительно, как следует из (9), $B_i \cup B_j$ есть блок разбиения $\theta_p^1 + \theta_q^1$. Любая более крупная коллекция $C_i^{p_i} \subset \theta_{p_i}^1$, порожденная разбиением $\theta_{p_1}^1 + \ldots + \theta_{p_v}^1 = \theta_k^v$, состоит из блоков, сопряженных по подмножеству разбиений $\{\theta_{p_1}^1, \ldots, \theta_{p_i}^1, \ldots, \theta_{p_v}^1\}$.

Не всякая совокупность из *n* разбиений множества $S(|S| = 2n)$ на пары элементов является первым уровнем булевой структуры.

Теорема 1. Множество $\{\theta_0, \ldots, \theta_{n-1}\}$ разбиений множества $S(|S| = 2^n)$ на пары элементов является первым уровнем булевой структуры, если и, только если выполнены два условия:

1) для любого блока $B_i \in \theta_p$ $(p = 0, \ldots, n-1)$ в разбиении θ_p найдется блок B_j, сопряженный с блоком B_i по θ_q $(q = 0, \ldots, n-1)$;

2) мощность совокупности блоков разбиения θ_q $(q=0,\ldots,n\text{-}1)$, сопряженных по любому подмножеству разбиений $H = \{\theta_{p_1}, \ldots, \theta_{p_v}\}$, $\theta_q \in H$, равна 2^{v-1}.

Содержательный смысл теоремы состоит в том, что из блоков разбиений $\theta_0, \ldots, \theta_{n-1}$ можно составить все необходимые коллекции. И это обеспечит возможность построения из разбиения на пары всех остальных разбиений.

Доказательство. Докажем прямое утверждение. Предположим, что $\{\theta_0, \ldots, \theta_{n-1}\}$ действительно первый уровень булевой структуры, но пара $(a, b) \in \theta_p$ не имеет в разбиении θ_p сопряженной с ней пары по θ_q. Это значит, что если $\{\overline{a, c};\ \overline{b, d}\} \subset \theta_q$, то элементы *c, d* в разбиении θ_p находятся в разных парах, пусть (*c, f)* и *(d, h).* Тогда разбиение $\theta_p + \theta_q$ будет содержать блок $\overline{a, b, c, d, f, h}$.

Но по определению структуры блока такой мощностине может быть ни в одном ее разбиении. Следовательно, $\{\theta_0, \ldots, \theta_{n-1}\}$ при принятом предположении относительно $(a, b) \in \theta_p$ не может быть первым уровнем структуры. Невыполнение второго условия означает, что в разбиении *v*-го уровня $\theta_{p_1} + \ldots + \theta_{p_v}$ имеется блок мощности, не равной 2^v. Следовательно, и предположение о невыполнении второго условия вступает в противоречие с тем, что $\{\theta_0, \ldots, \theta_{n-1}\}^*$ — первый уровень булевой структуры.

Для доказательства обратного утверждения достаточно показать, что условия теоремы обеспечивают выполнение всех аксиом булевой алгебры для множества в разбиений, равных попарным произведениям $\theta_p \cdot \theta_q$ *(p, q = 0, ... n—1, p≠q)* и всем возможным суммам разбиений из $\{\theta_0, \ldots, \theta_{n-1}\}$

$$\theta_t^v = \sum_{k=1}^{v} \theta_{i_k}, \quad v = 1, \ldots, n; \quad t = 0, \ldots, c_n^v - 1,$$

где $i_1, \ldots, i_v$ — одно из сочетаний из $\{0, \ldots, n-1\}$ по *v* чисел. Из (10) следует наличие нулевого разбиения в множестве Θ

$$\Theta^0 = \theta_p \cdot \theta_q = \{\overline{s}_0, \overline{s}_1, \ldots, \overline{s}_{2^n-1}\}.$$

Из условий теоремы и выражения (9) следует наличие единичного разбиения в множестве Θ

$$\Theta^n = \sum_{k=0}^{n-1} \theta_k = \{\overline{s_0, s_1, \ldots, s_{2^n-1}}\},$$

а также дополнения $\theta_t^{n-v} = \theta_{i_{v+1}} + \ldots + \theta_{i_n}$ для каждого разбиения $\theta_k^v = \theta_{i_1} + \ldots + \theta_{i_v}$. С очевидностью выполняются все остальные аксиомы булевой алгебры. Теорема доказана.

Условимся вместо длинной фразы „коллекция, порожденная в $\theta_{p_k}^1$ разбиением *v*-го уровня, равным сумме разбиений $\theta_{p_1}^1, \ldots, \theta_{p_k}^1, \ldots, \theta_{p_v}^1$", говорить „коллекция, порожденная в $\theta_{p_k}^1$ множеством разбиений

$$\theta_{p_1}^1, \ldots, \theta_{p_k}^1, \ldots, \theta_{p_v}^1.$$

Следствие. Пусть имеются некоторое подмножество разбиений $H = \{\theta_{p_1}^1, \ldots, \theta_{p_v}^1\}$ и разбиение $\theta_t^1 \notin H$. Элементы любого блока $B_j \in \theta_t^1$ не могут входить в блоки одной и той же коллекции

$C_{l}^{p_k} \subset \theta_{p_k}^{1}$ $(k=1, \ldots, v;$ $l=0, \ldots, 2^{n-v}-1)$, порожденной в $\theta_{p_k}^{1}$ множеством разбиений *H.*

Доказательство. Предположим обратное. Пусть пары *(a, b)* и *(c, d)* составляют коллекцию в разбиении $\theta_l^1 \in H,\ H=\{\theta_l^1,\ \theta_q^1\}$, а в разбиении $\theta_i^1 \notin H$ есть пара *(a, d).* Но это означает, что в разбиении θ_i имеется пара *(b, c),* а в разбиении θ_q^1—коллекция $\{\overline{a,\ c};\ \overline{b,\ d}\}$ (по первому условию теоремы). Тогда оказывается, что в каждом из разбиений $\theta_i^1,\ \theta_l^1,\ \theta_q^1$ разбиение третьего уровня $\theta_i^1+\theta_l^1+\theta_q^1$ порождает коллекции $\{\overline{a,\ d};\ \overline{b,\ c}\}$, $\{\overline{a,\ b};\ \overline{c,\ d}\}$ и $\{\overline{a,\ c};\ \overline{b,\ d}\}$ соответственно. Мощность этих коллекций равна 2 вместо 2^2, как того требует второе условие теоремы. Очевидно, при принятом предположении такое противоречие будет иметь место для подмножества *H* любой мощности. Следствие доказано.

Сформулируем без доказательств аналогичные теореме 1 и следствию 1 утверждения для *(n*— 1)-го уровня структуры. Будем говорить, что *блок* $B_i \in \theta_p^{n-1}$ *развязывается* разбиением θ_q^{n-1} на подблоки B'_{i_q} и B''_{i_q}, если $\{B'_{i_q},\ B''_{i_q}\} \subset \theta_p^{n-1}\cdot\theta_q^{n-1}$, и записывать это в виде $\{B'_{i_q},\ B''_{i_q}\} = B_i^{\#}(\theta_q^{n-1})$. Другими словами, блоки $B_0,\ B_1 \in \theta_q^{n-1}$ включают в себя по половине блока $B_i \in \theta_p^{n-1}$.

Подблоки B'_{i_q}, B''_{i_q} являются секциями, порожденными в θ_p^{n-1} разбиением $\theta_i^{n-2} = \theta_q^{n-1}\cdot\theta_p^{n-1}$.

Пример 23. Пусть имеются разбиения

$\theta_0^2=\{\overline{a,b,c,d};\ \overline{e,f,g,h}\}$; $\theta_1^2=\{\overline{a,b,e,f};\ \overline{c,d,g,h}\}$.

Легко заметить, что любой блок $B_j \in \theta_0^2$ развязывается разбиением θ_1^2 и любой блок $B_j \in \theta_1^2$ — разбиением θ_0^2. Так, блок $\overline{a,\ b,\ c,\ d}$ развязывается в $\theta^2{}_1$ на $\overline{a,\ b}$ и $\overline{c,\ d}$, а блок $\overline{e,\ f,\ g,\ h}$—на $\overline{e,\ f}$ и $\overline{g,\ h}$, и, наоборот, блок $\overline{a,\ b,\ e,\ f} \in \theta_1^2$ развязывается в θ_0^2 на $\overline{a,\ b}$ и $\overline{e,\ f}$, блок $\overline{c,\ d,\ g,\ h} \in \theta_1^2$ на $\overline{c,\ d}$ и $\overline{g,\ h}$,

Теорема 2. Множество $\{\theta_0,\ \ldots,\ \theta_{n-1}\}$ двублочных разбиений множества $S(|S|=2^n)$ является *(n*—1)-м уровнем булевой структуры, если, и только если выполнены два условия:

1) каждый блок B_i $(i = 0, 1)$ любого разбиения θ_p $(p = 0, \ldots, n—1)$ развязывается каждым разбиением θ_q $(q = 0, n\text{-}1)$;

2) мощность пересечения подблоков $B_{i_q}^{i}$, каждый из которых является результатом развязывания блока $B_i \in \theta_r$ $(r=0, \ldots, n-1)$ разбиением θ_q из подмножества $H=\{\theta_{p_1}, \ldots, \theta_{p_v}\}$, $\theta_r \in H (q=p_1, \ldots, p_v)$, равна 2^{n-v}.

Следствие 1. Пусть имеются некоторое подмножество разбиений $H=\{\theta_{p_1}^{n-1}, \ldots, \theta_{p_v}^{n-1}\}$ и разбиение $\theta_l^{n-1} \notin H$. Ни один блок $B_j \in \theta_i^{n-1}$ не может включать в себя ни одну секцию, порождаемую в $\theta_{p_k}^{n-1}$ $(k=1, \ldots, v)$ множеством разбиений *H*.

Это утверждение можно усилить, чтобы сделать его более конструктивным.

Следствие 2. Если $\theta_i^{n-1} \notin H$, то блоки разбиения θ_i^{n-1} включают ровно ПОЛОВИНУ любой секции, порождаемой множеством разбиений *H* в каждом разбиении $\theta_{p_k}^{n-1} \in H$.

6.2.4. Алгоритмы синтеза структуры

Изложим процедуры формирования разбиений первого и (*n*—1)-го уровней структуры, которые основаны на теоремах и следствиях § 6.2.3. Основным действием в этих процедурах является построение коллекции или секции. Исходной информацией, содержащей требования к коду, при построении первого уровня является список P_1 взвешенных пар состояний, которым желательно было бы присвоить соседние коды, при построении *(n*—1) -го уровня — список P_2 пар подмножеств состояний, которые необходимо развязать хотя бы по одному разряду кода, т. е. поместить эти подмножества в разные блоки хотя бы одного разбиения. Список P_1 формируется из *K*-множеств следующим образом. Пара состояний (s_i, s_j) включается в список P_1, если она входит хотя бы в одно *K*-множество. Вес пары (s_i, s_j) определяется числом *K*-множеств, включающих ее в себя. Список P_2 составляется из пар состязающихся переходов или *k*-множеств в зависимости от того, какое условие отсутствия в автомате опасных состязаний используется.

Построение разбиений первого уровня структуры. Введем определенную нумерацию пар в разбиениях первого уровня. Будем считать, что разбиение $\theta^1{}_i$ состоит из 2^{n-i-1} коллекций, порождаемых подмножеством $\{\theta_0^1, \ldots, \theta_i^1\}$, а каждая коллекция — из 2^i блоков:

$$C_j^i = \{B_{j_1}, B_{j_1+1}, \ldots, B_{j_1+2^i-1}\}, \quad j=0, \ldots, 2^{n-i-1}.$$

Первый блок в *j*-й коллекции имеет номер

$$j_1 = j \cdot 2^i. \quad (12)$$

Остальные блоки коллекции упорядочены следующим образом:

$$B_{j_1+k} = B^{\#}_{j_1+k-2^p}(\theta^1_p), \quad k = 1, \ldots, 2^i - 1, \quad (13)$$

где p — ближайшее к $\log_2 k$ целое число, меньшее $\log_2 k$.

Пример 24. Пусть имеется подмножество разбиений первого уровня (n=4)

$$\theta^1_0 = \{\overline{a,b};\ \overline{c,d};\ \overline{e,f};\ \overline{g,h};\ \overline{i,j};\ \overline{k,l};\ \overline{m,n};\ \overline{o,p}\};$$

$$\theta^1_1 = \{\overline{a,e};\ \overline{b,f};\ \overline{c,h};\ \overline{d,g};\ \overline{i,k};\ \overline{j,l};\ \overline{m,p};\ \overline{o,n}\};$$

$$\theta^1_2 = \{\overline{a,h};\ \overline{b,g};\ \overline{c,e};\ \overline{d,f};\ \overline{i,m};\ \overline{j,n};\ \overline{k,p};\ \overline{l,o}\}.$$

Разбиение θ^1_2 состоит из двух коллекций

$$C^2_0 = \{\overline{a,h};\ \overline{b,g};\ \overline{c,e};\ \overline{d,f}\};\quad C^2_1 = \{\overline{i,m};\ \overline{j,n};\ \overline{k,p};\ \overline{l,o}\}.$$

В соответствии с (13) блоки в коллекциях C^2_0 и C^2_1 пронумерованы и сопряжены следующим образом:

$$C^2_0 = \begin{cases} B_0 = \overline{a,h}, & i=2,\ j=0,\ j_1=0, \\ B_1 = B^{\#}_0(\theta^1_0) = \overline{b,g}, & k=1,\ p=0, \\ B_2 = B^{\#}_0(\theta^1_1) = \overline{c,e}, & k=2,\ p=1, \\ B_3 = B^{\#}_1(\theta^1_1) = \overline{d,f}, & k=3,\ p=1; \end{cases}$$

$$C^2_1 = \begin{cases} B_4 = \overline{i,m}, & i=2,\ j=1,\ j_1=4, \\ B_5 = B^{\#}_4(\theta^1_0) = \overline{j,n}, & k=1,\ p=0, \\ B_6 = B^{\#}_4(\theta^1_1) = \overline{k,p}, & k=2,\ p=1, \\ B_7 = B^{\#}_5(\theta^1_1) = \overline{l,o}, & k=3,\ p=1. \end{cases}$$

Каждое разбиение θ^1_i $(i=0, \ldots, n-1)$ строится коллекциями, порождаемыми в нем множеством уже построенных разбиений $\{\theta^1_0, \ldots, \theta^1_{i-1}\}$, с учетом списка P_1.

Процедура построения i-го разбиения следующая:

1. Строится разбиение θ^1_i $(i=0, \ldots, n-1)$, т. е. коллекции C^i_j $(j=0, \ldots, 2^{n-i-1}-1)$.

а) первая пара (блок) коллекции C^i_j, имеющая в θ^i_j номер j_1, вычисляемый по формуле (12), выбирается из списка P_1 так, чтобы было выполнено условие следствия 1, т. е. оба элемента выбранной пары не входили бы в пары ни одной уже построенной коллекции C^l_k $(l=0, \ldots, i-1,\ k=0, \ldots, 2^{n-l-1}-1)$; если в списке P_1 не оказывается подходящей пары, то она составляется произвольно из элементов множества S с учетом условия следствия 1;

б) остальные пары (блоки) $B_{j_1+1}, \ldots, B_{j_1+2^{i-1}-1}$ коллекции C^2_j вычисляются по формуле (13).

2. Пары, принадлежащие списку P_1 и попавшие в построенные коллекции, удаляются из P_1. Построенные коллекции объединяются в разбиение $\theta^1_l = \{C^l_0, \ldots, C^l_{2^{n-i-1}-1}\}$.

Полученное в результате выполнения описанной процедуры множество разбиений по способу построения удовлетворяет условиям теоремы 1 и, следовательно, является первым уровнем булевой структуры.

Пример 25. Построим множество разбиений $\Theta^1 = \{\theta^1_0, \theta^1_1, \theta^1_2\}$ наазе множества $S=\{a, b, c, d, e, f, g, h\}$ учетом списка $P_1 = \{(a,b), (c,d), (e, f), (g,h), (a,c), (a,d), (e, h), (a,g), (b, f)\}$. Считаем, что все пары списка имеют одинаковые веса.

1. Разбиение θ^1_0, каждая *i*-я пара (блок) которого является коллекцией C^0_i, образуем из попарно не пересекающихся четырех пар списка P_1:

$$\theta^1_0 = \{\overline{a,b};\ \overline{c,d};\ \overline{e,f};\ \overline{g,h}\}.$$

2. Разбиение θ^1_1 состоит из двух коллекций и $C^1_0 = \{B_0, B_1\}$ и $C^1_1 = \{B_2, B_3\}$. В качестве B_0 выбираем пару из списка P_1 — это (**a, c**) и находим $B_1 = B^{\#}_0(\theta^1_0) = \overline{b,\ d}$. Таким образом, $C^1_0 = \{\overline{a,c};\ \overline{b,d}\}$. Аналогично $C^1_1 = \{\overline{e,h};\ \overline{f,g}\}$. И наконец,

$$\theta^1_1 = \{\overline{a,c};\ \overline{b,d};\ \overline{e,h};\ \overline{g,f}\}.$$

3. Разбиение θ^1_2— одна коллекция C^2_0. Первая неиспользованная пара в списке P_1 — это (a,d), но согласно следствию 1 она не может быть взята в качестве $B_0 \in C^2_0$, так как *a* и *d* входят в пары коллекции C^1_0. Берем следующую пару $B_0 = \overline{a,g}$ и находим $B_1 = B^{\#}_0(\theta^1_0) = \overline{b,h}$, $B_2 = B^{\#}_0(\theta^1_1) = \overline{c,f}$, $B_3 = B^{\#}_1(\theta^1_1) = \overline{d,e}$.

И наконец,

$$\theta^1_2 = \{\overline{a,g};\ \overline{b,h};\ \overline{c,f};\ \overline{d,e}\}.$$

Построение разбиений *(n*—1)-го уровня структуры.

Перейдем к изложению процедур формирования разбиений *(n*—1)-го уровня. Обозначим пары подмножеств, составляющие список P_2, через $\pi_l : P_2 = \{\pi_l\}, \pi_l = (S_{l1}, S_{l2})$, $S_{l1}, S_{l2} \subset S$, $S_{l1} \cap S_{l2} = \varnothing$. Для пар списка P_2 введем следующие понятия. *Две пары* $\pi_i = (S_{i1}, S_{i2})$ *и* $\pi_j = (S_{j1}, S_{j2})$ будем называть *согласованными,* если каждое подмножество одной пары пересекается не более чем с одним

подмножеством другой пары. Для определенности будем считать, что для согласованных пар

$$S_{i1} \cap S_{j2} = \varnothing \quad \text{и} \quad S_{i2} \cap S_{j1} = \varnothing.$$

Пример 26. Пусть $\pi_1 = (a, b, c;\ d, e)$, $\pi_2 = (a, c;\ b, d, h)$, $\pi_3 = (a, c, f;\ d, e, g)$, $\pi_4 = (a, b;\ i, j, k)$, $\pi_5 = (i, l;\ j, m, n)$. Пары π_1 и π_3, π_2 и π_3, π_1 и π_4, π_1 и π_5, π_2 и π_5 и т. д. являются согласованными. Пары π_1 и π_2, π_2 и π_4 не являются согласованными.

Объединение двух согласованных пар подмножеств будем выполнять следующим образом:

$$\pi_i \cup \pi_j = (S_{i1} \cup S_{j1},\ S_{i2} \cup S_{j2}).$$

Так, объединение пар π_1 и π_3 из примера 25 будет

$$\pi_1 \cup \pi_3 = (a, b, c, f;\ d, e, g).$$

Разбиение θ_i^{n-1} формируется при условии, что разбиения $\theta_0^{n-1}, \ldots, \theta_{i-1}^{n-1}$ построены. Считаем, что разбиение θ_i^{n-1} состоит из 2^{i+1} секций мощности 2^{n-i-1}. Так, разюиие θ_0^{n-1} состоит из двух секций, которыми являются его собственные блоки: $E_0^0 = B_0$, $E_1^0 = B_1$. В θ_1^{n-1} входят четыре секции мощности 2^{n-2} каждая, в этом случае секция — это половина блока разбиения и т. д.

Построение разбиения θ_i^{n-1} при условии, что разбиение θ_{i-1}^{n-1} построено, состоит в расчленении каждой секции $E_k^{i-1} \in \theta_{i-1}^{n-1}$ пополам с учетом списка P_2 и формировании из этих половин блоков разбиения θ_i^{n-1}.

Процедура построения разбиений $\theta_0^{n-1}, \ldots, \theta_{n-1}^{n-1}$ при первоначально взятом $n =]\log_2 |S|[$ следующая (считаем, что перед выполнением процедуры все секции разбиений пусты).

1. Строится разбиение θ_i^{n-1} $(i = 0, \ldots, n-1)$, т. е. секции $E_0^i, \ldots, E_{2^i+1-1}^i$. Если список P_2 не пуст, то выполняются п. а—в. Иначе, секции $E_0^{i-1}, \ldots, E_{2^{i-1}-1}^{i-1}$ разбиения θ_{i-1}^{n-1} расчленяются произвольно пополам, т. е. из каждой E_j^{i-1} образуются две секции разбиения θ_i^{n-1}: E_j^i и $E_{j+2^i}^i$

(одна секция попадает в блок $B_0 \in \theta_i^{n-1}$, другая — в блок $B_1 \in \theta_i^{n-1}$).

Далее выполняется п. 2.

а) Берется некоторая пара из списка P_2, будем называть ее базовой для разбиения θ_i^{n-1} и обозначать через $\tilde{\pi}$. В списке P_2 отыскиваются все пары, согласованные с базовой, множество таких пар, включающее и

базовую, обозначается через $\tilde{P}$. Если такое множество не единственно, то берем любое.

б) Берется пара $\pi_j \in \tilde{P}$ $(j = 1, \ldots, |\tilde{P}|)$. Отыскиваются все секции $E^{i-1}_{j_1}, \ldots, E^{i-1}_{j_k}$, которые пересекаются хотя бы с одним из подмножеств S_{j1}, $S_{j2} \in \pi_j$. Определяются пересечения

$$\tilde{E}_{j_r} = E^{i-1}_{j_r} \cap S_{j1}, \quad \tilde{\tilde{E}}_{j_r} = E^{i-1}_{j_r} \cap S_{j2},$$

$(r = 1, \ldots, k)$. Из найденных пересечений $\tilde{E}_{j_r}$, $\tilde{\tilde{E}}_{j_r}$ строятся секции $E^i_{j_r}$, $E^i_{j_r+2^i}$ $(r = 1, \ldots, k)$ разбиения θ^{n-1}_i.

При этом $\tilde{E}_{j_r}$ включается в секцию $E^i_{j_r}$, $\tilde{\tilde{E}}_{j_r}$ — в секцию $E^i_{j_r+2^i}$. Если $|E^i_{j_r}| = 2^{n-i-1}$ или $|E^i_{j_r+2^i}| = 2^{n-i-1}$ (это означает, что секция $E^i_{j_r}$ или $E^i_{j_r+2^i}$ сформирована), что значит сформирована и секция $E^i_{j_r+2^i} = E^{i-1}_{j_r} \setminus E^i_{j_r}$ или $E^i_{j_r} = E^{i-1}_{j_r} \setminus E^i_{j_r+2^i}$. Если $|E^i_{j_r}| > 2^{n-i-1}$ или $|E^i_{j_r+2^i}| > 2^{n-i-1}$, то значит, что соответствующая пара списка P_2 не может быть развязана в θ^{n-1}_i.

в) Если все $\pi_j \in P$ испытаны, но не все секции E^i_k имеют мощность 2^{n-i-1} (а это значит, что не все секции разбиения θ^{n-1}_{i-1} расчленены), то нерасчлененные секции E^{i-1}_k расчленяются произвольно.

2. Пары списка P_2, развязанные в θ^{n-1}_i, удаляются из P_2. Из построенных секций образуется разбиение

$$\theta^{n-1}_i = \{\overline{E^i_0, \ldots, E^i_{2^i-1}}; \overline{E^i_{2^i}, \ldots, E^i_{2^{i+1}-1}}\}.$$

В результате выполнения описанной процедуры может оказаться, что построены n разбиений, а список P_2 не исчерпан, т. е. не все пары подмножеств развязаны. В этом случае следует увеличить n на единицу, множество S дополнить фиктивными состояниями до $|S| = 2^{n+1}$, блоки уже построенных разбиений расширить до 2^n за счет фиктивных состояний с учетом следствия 2. Затем нужно построить еще одно разбиение и развязать оставшиеся пары. Если же и этого окажется недостаточно для исчерпания списка P_2, то следует проделать то же самое с еще большим n, и т. д.

Задача поиска минимального по мощности множества двублочных разбиений, развязывающих все пары списка P_2, формулируется как задача поиска кратчайшего покрытия списка P_2 двублочными разбиениями. Вопросы, связанные с эвристическим решением этой задачи, обсуждаются дальше. Здесь заметим, что число двублочных разбиений, развязывающих все пары списка P_2, зависит от того, какие

пары взяты в качестве базовых для θ_i^{n-1} и какие из подмножеств пар, согласованных с некоторой базовой (если такое подмножество не единственно), используются для построения разбиения θ_i^{n-1}.

Пример 27. Работу алгоритма покажем на множестве состояний $S=\{a, b, c, d, e, f, g, h\}$ и списке P_2:

$$\pi_1=(b,d;\ c,e);\quad \pi_2=(a,d;\ c,e);\quad \pi_3=(a,b;\ e,g);$$
$$\pi_4=(a,b;\ c,d);\quad \pi_5=(b,e;\ c,d);\quad \pi_6=(a,d;\ b,c);$$
$$\pi_7=(a,h;\ b,c);\quad \pi_8=(a,f;\ g,h);\quad \pi_9=(a,g;\ h,\ f).$$

Возьмем $n=3$

1. Строим θ_0^2. В качестве базовой берем пару π_4, т. е. $\tilde{\pi}=(b,d;c,e)$ Тогда $\tilde{P}=\{\pi_1, \pi_2, \pi_3, \pi_8\}$. Разбиение θ_0^{n-1} получим, объединив все пары из списка $\tilde{P}$:

$$\theta_0^2=\{\overline{a, b, d, f};\ \overline{c, e, g, h}\}.$$

2. В качестве базовой для θ_0^2 берем пару π_4, т. е. $\tilde{\pi}=(a, b;\ c, d)$. Тогда $\tilde{P}=\{\pi_4, \pi_5, \pi_9\}$. Найдем пересечения пары π_4 с E_0^0 и E_1^0:

$$\tilde{E}_0=E_0^0\cap S_{41}=\{a, b, d, f\}\cap\{a, b\}=\{a, b\};$$
$$\tilde{\tilde{E}}_0=E_0^0\cap S_{42}=\{a, b, d, f\}\cap\{c, d\}=\{d\};$$
$$\tilde{\tilde{E}}_1=E_1^0\cap S_{42}=\{c, e, g, h\}\cap\{c, d\}=\{c\}.$$

Таким образом, $E_0^1=\{a, b\}$, а это значит, что $E_2^1=\{d, f\}$. Кроме того, $E^1{}_3=\{c\}$. Пересечения пары π_5 с E_0^0 и E_1^0:

$$\tilde{E}_0=E_0^0\cap S_{51}=\{b\};\quad \tilde{\tilde{E}}_0=E_0^0\cap S_{52}=\{d\};$$
$$\tilde{E}_1=E_1^0\cap S_{51}=\{e\};\quad \tilde{\tilde{E}}_1=E_1^0\cap S_{52}=\{c\}.$$

Тогда $E^1{}_1=\{c\}$. И наконец, пересечения пары π_9 с E_0^0 и E_1^0:

$$\tilde{E}_0=\{a\};\ \tilde{\tilde{E}}_0=\{f\};\ \tilde{E}_1=\{g\};\ \tilde{\tilde{E}}_1=\{h\};\ E_1^1=\{e, g\};\ E_1^3=\{c, h\}.$$

Таким образом,

$$\theta_1^2=\{\overline{a, b, e, g};\ \overline{d, f, c, h}\}.$$

3. Базовая пара для разбиения θ_2^1 $\tilde{\pi}=\pi_6$, $\tilde{P}=\{\pi_6, \pi_7\}$. Пересечения пары π_6 с E_0^1, E_2^1, E_3^1:

$$\tilde{E}_0=\{a\};\ \tilde{E}_2=\{d\};\ \tilde{\tilde{E}}_0=\{b\};\ \tilde{\tilde{E}}_3=\{c\}.$$
$$E_0^2=\{a\};\ E_2^2=\{d\};\ E_3^2=\{h\};\ E_4^2=\{b\};\ E_6^2=\{f\};\ E_7^2=\{c\}.$$

Пересечение пары π_7 с $E^1{}_1$ дает $E_1^2=\{g\}$, $E_5^2=\{e\}$. Таким образом,

$$\theta_2^2=\{\overline{a, g, d, h};\ \overline{b, e, f, c}\}.$$

6.2.5. Компактное представление структуры

Линейное упорядочение состояний. Введем компактное представление булевой структуры, образованной разбиениями множества состояний $S = \{s_0, \ldots, s_{2^n-1}\}$.

Вместо громоздкой системы разбиений мы будем иметь дело с матрицей-строкой $D = \|d_i\|$ $(i = 0, \ldots, 2^n-1)$. Состояния будем представлять *n*-интервалами или соответствующими десятичными числами от 0 до 2^n—1. Разбиения структуры извлекаются из *D* с помощью простых правил. Порядок расположения чисел в строке *D* и вытекающие из него правила извлечения разбиений сначала проиллюстрируем на следующем простом построении.

Запишем в строке *D* числа 0, ..., 2^n—1 в лексикографическом порядке, возьмем $n = 4$:

$$D = (0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15). \quad (14)$$

Разобьем строку *D* на пары элементов, не нарушая порядка следования чисел. Полученное разбиение

$\{\overline{0,1};\ \overline{2,3};\ \overline{4,5};\ \overline{6,7};\ \overline{8,9};\ \overline{10,11};\ \overline{12,13};\ \overline{14,15}\}$ является разбиением θ_0^1 первого уровня структуры, *n*-интервалы, составляющие любую пару разбиения, являются соседними по координате x_0. Следующее разбиение θ_1^1 первого уровня можно получить, разбив строку *D* на четверки элементов и затем объединив в пары элементы 0- и 2-й, 1- и 3-й каждой четверки:

$$\theta_1^1 = \{\overline{0,2};\ \overline{1,3};\ \overline{4,6};\ \overline{5,7};\ \overline{8,10};\ \overline{9,11};\ \overline{12,14};\ \overline{13,15}\}.$$

Двоичные аналоги чисел каждой пары различаются только в первом разряде. Разбиение θ_3^1 получим, разбив строку *D* на восьмерки и объединив в пары элементы 0- и 4-й, 1- и 5-й, 2- и 6-й, 3- и 7-й каждой восьмерки:

$$\theta_2^1 = \{\overline{0,4};\ \overline{1,5};\ \overline{2,6};\ \overline{3,7};\ \overline{8,12};\ \overline{9,13};\ \overline{10,14};\ \overline{11,15}\}.$$

И наконец, разбиение θ_3^1 получим, объединив в пары элементы строки *D* 0- и 8-й, 1- и 9-й, ..., 7- и 15-й, воспринимая строку как одну группу из 16 чисел:

$$\theta_3^1 = \{\overline{0,8};\ \overline{1,9};\ \overline{2,10};\ \overline{3,11};\ \overline{4,12};\ \overline{5,13};\ \overline{6,14};\ \overline{7,15}\}.$$

Существование алгоритма извлечения из матрицы-строки *D* всех разбиений первого уровня убеждает нас в том, что строка *D* с лексикографическим порядком расположения в ней чисел является представлением булевой структуры $\tilde{\Theta}$. Однако лексикографический порядок вовсе не обязателен для чисел в строке *D*, а для состояний,

обозначенных буквами, он просто не определен. Ключом порядка является следующее положение: если *a* и *b* являются блоком разбиения θ_k^1, т. е. $\overline{a,\ b} \in \theta_k^1$, то в строке *D* они должны попасть на такие места, двоичные номера которых различаются только в *k*-м разряде, т. е. $d_i = a$, $d_j = b$, и

$$i_{(2)} \oplus j_{(2)} = \underbrace{0\ \ldots\ 010\ \ldots\ 0}_{k}, \quad (15)$$

где $i_{(2)}$ — двоичное число; $\bigoplus$ —операция сложения по mod 2. Для *D* с лексикографическим порядком чисел это положение является тривиальным, так как номера элементов совпадают с самими элементами.

Пример 28. В строке

$$D' = (10,\ 11,\ 8,\ 9,\ 14,\ 15,\ 12,\ 13,\ 2,\ 3,\ 0,\ 1,\ 6,\ 7,\ 4,\ 5)$$

числа расположены не в лексикографическом порядке, но в соответствии с (15). Разбиения первого уровня, извлеченные из строки *D'* с помощью приемов, описанных выше, имеют вид

$$\theta_0^1 = \{\overline{10,11};\ \overline{8,9};\ \overline{14,15};\ \overline{12,13};\ \overline{2,3};\ \overline{0,1};\ \overline{6,7};\ \overline{4,5}\};$$

$$\theta_1^1 = \{\overline{10,8};\ \overline{11,9};\ \overline{14,12};\ \overline{15,13};\ \overline{2,0};\ \overline{3,1};\ \overline{6,4};\ \overline{7,5}\};$$

$$\theta_2^1 = \{\overline{10,14};\ \overline{11,15};\ \overline{8,12};\ \overline{9,13};\ \overline{2,6};\ \overline{3,7};\ \overline{0,4};\ \overline{1,5}\};$$

$$\theta_3^1 = \{\overline{10,2};\ \overline{11,3};\ \overline{8,0};\ \overline{9,1};\ \overline{14,6};\ \overline{15,7};\ \overline{12,4};\ \overline{13,5}\}.$$

Сравнивая эти разбиения с полученными из (14), убеждаемся, что это те же самые разбиения и, следовательно, *D'*, как и *D*, представляет структуру $\hat{\Theta}$.

Дадим формальное описание порядка чисел в строке *D*, которую далее будем называть *D-представлением структуры* $\hat{\Theta}$. Будем считать, что каждому разбиению θ_i^1 соответствует разбиение строки *D* на 2^{n-i-1} подстрок $D^i = (D_0^i,\ D_1^i,\ \ldots,\ D_{2^{n-i-1}-1}^i)$ по 2^{i+1} элементов в каждой. Состояния (или числа) в каждой подстроке D_j^i, объединенные в пары, образуют блоки коллекции C_j^i. Блоки коллекции C_j^i составляются из следующих элементов строки:

$$B_{j_1+k} = (d_{j_1+k},\ d_{j_1+2^i+k}), \quad j_1 = j \cdot 2^{i+1};\ k = 0, \ldots, 2^i - 1. \quad (16)$$

Формула (16) соответствует нумерации (12), (13) блоков в коллекциях, введенной по признаку сопряженности. В (16) эта нумерация относится не только к блокам, но и к расположению элементов блока в строке *D*. Выражения (15) и (16) являются разными формами описания порядка чисел в *D* -представлении структуры.

Из (15) и (16) следует, что если некоторое число *a* назначено быть начальным в *D*, т. е. $d_0 = a$, то все остальные числа расположатся в *D*

единственным образом. Строку D с начальным элементом a будем обозначать через $D(a)$. Чтобы получить простую формулу для построения $D(a)$, преобразуем (15) к виду

$$d_{i(2)} \oplus d_{j(2)} = i_{(2)} \oplus j_{(2)} = 0 \ldots \underset{k_v}{010} \ldots 1 \ldots \underset{k_1}{010}, \quad \textbf{(17)}$$

что означает: если n-интервалы d_i и d_j различаются в разрядах $k_1, \ldots, k_v$, т. е. d_i и d_j входят в один блок разбиения $\theta_k^v = \theta_{k_1}^1 + \ldots + \theta_{k_v}^1$, то в тех же самых разрядах различаются и двоичные номера места в D этих n-интервалов. Положив в (17) i=0, d_0=a, выразим d_j через j и a:

$$d_{j(2)} = j_{(2)} \oplus a_{(2)}. \quad \textbf{(18)}$$

Придавая j значения $1, \ldots, 2^n - 1$ и вычисляя d_j по (18), получаем $D(a)$. Можно проверить, что D (10) при n = 4 есть строка D из примера 28, а $D(0)$ есть строка с лексикографическим порядком чисел.

Обратим внимание на то, что двоичный вектор из (17) есть g-вектор, соответствующий разбиению $\theta_k^v = \theta_{k_1}^1 + \ldots + \theta_{k_v}^1$. Вектор $j_{(2)}$ из (18) также можно интерпретировать как g -вектор, соответствующий разбиению, в один блок которого входят a и d_j, при этом $j_{(2)}$ есть g -вектор наименьшего разбиения среди всех разбиений, в которых a и d_j входят в один блок. Придавая j значения от 1 до 2^n — 1, мы как бы проходим по всем разбиениям структуры и получаем блок каждого разбиения, включающий a и d_j. Вообще говоря, этот блок содержит все d_k, такие, что

$$k_{(2)} \leqslant j_{(2)}. \quad (19)$$

Пример 29. По формуле (18) построим

$$\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{matrix}$$
$$D(2) = (2,\ 3,\ 0,\ 1,\ 6,\ 7,\ 4,\ 5,\ 10,\ 11,\ 8,\ 9,\ 14,\ 15,\ 12,\ 13).$$

Проиллюстрируем использование формулы (19). Пусть g =0101, т. е. j = 5, d_j = 7, найдем блок соответствующего разбиения, содержащий числа 2 и 7. В этот блок кроме d_0 и d_5 войдут d_1 и d_4, так как $0001 < 0101$ и $0100 < 0101$ И искомый блок $\overline{2,3,6,7}$.

Если в D-представлении структуры поменять местами любые два элемента, то, чтобы строка D осталась представлением структуры, все остальные элементы должны подвергнуться перестановке. Пусть поменяли местами элементы b и c: было $d_i = b$, $d_i = c$, стало $d_j = b$, $d_i = c$. Согласно формуле (17) в нетронутом еще D -представлении для $d_i = b$ и некоторого $d_k = e$ имеет место равенство

$$b_{(2)} \oplus e_{(2)} = i_{(2)} \oplus k_{(2)}. \quad \textbf{(20)}$$

После перестановки b и c для тех же элементов b и e и их новых мест d_j и d_x получим

$$b_{(2)} \oplus e_{(2)} = j_{(2)} \oplus x_{(2)}. \quad (21)$$

Из (20) и (21) находим

$$x_{(2)} = i_{(2)} \oplus j_{(2)} \bigoplus k_{(2)}. \quad (22)$$

Выражение (22) позволяет найти в D-представлении новое место d_x некоторому элементу, который первоначально занимал место d_k, если поменять местами два элемента, первоначально занимающие в D места i и j.

Синтез структуры на основе D -представления состоит в том, чтобы найти место каждому состоянию в строке D с учетом списка P_1 взвешенных желаемых соседств (пар) состояний или списка P_2 пар состязающихся подмножеств состояний. Например, для первого уровня это делается так. В строку D первоначально записывают состояния в таком порядке, чтобы начиная с d_0 рядом стоящие состояния образовывали пары разбиения θ_0^1. Построение каждого следующего разбиения θ_i^1 (i = 1, ..., n—2) состоит в том, чтобы переупорядочить элементы в D, сохранив ранее построенные разбиения. Такое переупорядочение производится с учетом выражения (22). Алгоритмы кодирования по существу и сводятся к построению структуры в виде строки D. Эти алгоритмы приведены дальше.

Компактное представление усеченной структуры. Условимся в строке D, представляющей усеченную структуру $\Lambda(a_h)$, вместо n-интервалов, не принадлежащих множеству $I^1(a_h)$, ставить знак $\varnothing$. Номера мест, которые должны занимать в $D(a_h)$ n-интервалы $a_k \in I^1(a_h)$, вычисляются по формуле $i_{(2)} = a_{k(2)} \bigoplus a_{h(2)}$, (23) следующей из (18).

В $D(a_h)$, представляющей $\Lambda(a_h)$, «пустых» мест может оказаться гораздо больше, чем «заполненных». В этом случае имеет смысл сжать строку $D(a_h)$, удалив «пустые» места. Строку, полученную из $D(a_h)$ в результате такого сжатия, будем обозначать через $\Delta(a_h) = \| \hat{a}_i \| \ (i = 1, \ldots, | I^1(a_h) |)$.

Пример 30. Пусть задана булева функция, имеющая следующие области нулевого и единичного значений: I^0={4,5,12,13,14,15}, I^1= {0,1,2,3,6,7,8,9,10,11}. Для усеченной структуры $\Lambda(2)$ $D(2)$ = = (2,3,0,1,6,7,∅,∅, 10,11,8,9, ∅,∅,∅,∅).. Можно сравнить полученную строку с $D\{2\}$ из примера 29. Сожмем D (2), удалив «пустые» места, получим:

$$\Delta(2) = (2,\ 3,\ 0,\ 1,\ 6,\ 7,\ 10,\ 11,\ 8,\ 9).$$

Векторы g_i, которые в D существуют как двоичные номера мест, в Δ утрачиваются, однако их можно легко восстановить по формуле (23). Более того, в $\Delta(a_h)$ вместо самих n-интервалов можно хранить утраченные g-векторы, а n-интервалы, если нужно, восстанавливать по g-векторам. Из такой строки легко выделить g-векторы, составляющие

множество $G'_{max}(a_h)$ (напомним, что каждый $g_i \in G'_{max}(a_h)$ представляет максимальный в $I^1(a_h)$ интервал, включающий a_h). Вектор g_i, записанный в $\Delta(a_h)$ или восстановленный из $\Delta(a_h)$ч по формуле (23), принадлежит $G'_{max}(a_h)$, если в $\Delta(a_h)$ не найдется такого g_j, что $g_j > g_i$. Заметим, что начать выделение максимальных (по аналогии с максимальным интервалом) *g*-векторов из $\Delta(a_h)$ лучше всего с правого конца строки (крайний правый в $\Delta(a_h)$ g-вектор есть первый максимальный *g*-вектор) и удалять из рассмотренных все g-векторы, которые оказались меньше очередного максимального. Такой прием позволяет уменьшить перебор.

Пример 31. Для строки Δ(2), полученной в примере 30, запишем Δ(2) для g-векторов:

$$\Delta(2) = (0000, 0001, 0010, 0011, 0100, 0101, 1000, 1001, 1010, 1011).$$

Первым максимальным g-вектором является 1011 (последний). Векторы 1010, 1001, 1000, 0011, 0010, 0001, 0000 удаляем из рассмотрения, так как каждый из них меньше, чем 1011. Из оставшихся *g*-векторов (0100, 0101) еще одним максимальным является 0101. Выделенным g-векторам соответствуют следующие максимальные интервалы:

$$1011 — \{0,1,2,3,8,9,10,11\}; \quad 0101 — \{2,3,6,7\}.$$

В заключение заметим, что компактное представление структуры важно для создания практических алгоритмов синтеза на основе теоретико-структурного подхода. Выгода, которую дает такое представление, очевидна сразу: для записи структуры в память ЭВМ необходимо 2^n ячеек, для хранения разбиений одного из самых простых нетривиальных уровней (первого или *(n*—1)-го) $n \cdot 2^n$ ячеек.

6.3. Алгоритмы приближенной минимизации систем булевых функций

6.3.1. Общие схемы алгоритмов

Предположим, что система булевых функций *F(X)* задана в виде множества Ψ_0 обобщенных интервалов. Это множество будем называть исходным. Задача приближенной минимизации системы *F(X)* состоит в построении из исходного Ψ_0 множества Ψ_R обобщенных интервалов, реализующего систему *F(X)* и по мощности близкого к кратчайшему. Слова «близкого к кратчайшему» нужно воспринимать как некоторую условность, поскольку определить точно, насколько полученное решение близко к кратчайшему, невозможно, хотя стремление подойти как можно ближе к кратчайшему множеству обобщенных интервалов

является основным при построении алгоритмов приближенной минимизации. (Понятия «обобщенный интервал», «множество обобщенных интервалов, реализующее систему *F(X)»,* «кратчайшее множество обобщенных интервалов» определены ранее).
Алгоритмы приближенной минимизации систем булевых функций чаще всего строятся по следующим двум схемам. Из исходного множества Ψ_0 выбирается некоторый обобщенный интервал, который будем называть *порождающим элемент решения* или просто *порождающим.* К порождающему интервалу шаг за шагом присоединяются интервалы, выбранные с помощью различных эвристических критериев так, чтобы полученный максимальный обобщенный интервал (см. определение) был бы «самым хорошим» в некотором смысле. «Самый хороший» интервал включается в решение Ψ_R, а все обобщенные интервалы, реализованные этим «самым хорошим», удаляются из Ψ_0. Такой процесс продолжается до тех пор, пока не окажутся реализованными все обобщенные интервалы из Ψ_0.
Другая схема состоит в построении для порождающего обобщенного интервала всех максимальных интервалов, содержащих порождающий, и в выборе из них «самого хорошего», который и включается в решение Ψ_R.
Качество решения (число обобщенных интервалов в решении Ψ_R), полученного с помощью алгоритмов, построенных по таким схемам, а также объем производимого при этом перебора зависят от того, какой обобщенный интервал из Ψ_0 назначается на каждом шаге алгоритма порождающим и насколько удачно выбирается направление поиска «самого хорошего» максимального обобщенного интервала.
Обозначим Ψ' множество всех максимальных обобщенных интервалов, включающих порождающий интервал. Для выбора из Ψ' «самого хорошего» интервала будем использовать следующий эвристический критерий: «самым хорошим» считается тот интервал, который реализует самое большое количество еще не реализованных единичных состояний функций системы. Будем говорить, что такой интервал обладает наибольшей ценой по сравнению со всеми другими из Ψ'. *Ценой* будем считать число единичных состояний функций системы, реализованных этим обобщенным интервалом и не реализованных ни одним ранее включенным в решение обобщенным интервалом.
Предлагаются **четыре эвристических алгоритма** построения близкого к кратчайшему множества обобщенных интервалов для системы *F(X).* Эти алгоритмы строятся по второй схеме, т. е. множество Ψ' для порождающего обобщенного интервала строится целиком или почти целиком. Следующие действия являются основными в одном шаге каждого из предлагаемых алгоритмов:

1) выбор из множества Ψ_0 порождающего обобщенного интервала;
2) построение множества Ψ' для порождающего интервала;
3) выбор из Ψ' интервала с наибольшей ценой, включение его в решение и удаление из множества Ψ_0 всех интервалов, реализованных им.

Все четыре алгоритма в случае использования одного и того же способа выбора из Ψ_0 порождающего интервала дают одинаковые решении, а различаются способами построения множества Ψ' и выбора из него интервала с наибольшей ценой. Каждый способ эффективен (в смысле экономии времени и (или) памяти ЭВМ) для своего класса систем булевых функций, характеризующегося определенным соотношением между числом переменных и мощностью множества Ψ_0.

Прежде чем изложить подробно три основные процедуры, составляющие алгоритмы минимизации систем булевых функций, уточним некоторые понятия.

Троичные векторы β_i, β_j—выходные части обобщенных интервалов — будем сравнивать поразрядно, чтобы установить между ними одно из отношений: „меньше по единицам" или „больше по единицам,, ($\overset{1}{<}$ или $\overset{1}{>}$), „равно по единицам" ($\overset{1}{=}$), „несравнимо" ($\overset{1}{\lessgtr}$). При сравнении разрядов, в которых стоят единицы и нули, считаем, что $\beta_i^k \overset{1}{=} \beta_j^k$, если $\beta_i^k = \beta_j^k$, и $\beta_i^k \overset{1}{<} \beta_j^k$, если $\beta_i^k < \beta_j^k$. Если в одном из двух сравниваемых разрядов или в обоих стоят звездочки $(*)$, то эти разряды считаем равными по единицам, т. е. Если $\beta_i^k = *$ или $\beta_j^k = *$, то $\beta_i^k \overset{1}{=} \beta_j^k$. Тогда $\beta_i \overset{1}{=} \beta_j$, если $\beta_i^k \overset{1}{=} \beta_j^k$ для всех k; $\beta_i \overset{1}{<} \beta_j$, если $\beta_i^k \overset{1}{\leqslant} \beta_j^k$ для всех k и хотя бы для одного k $\beta_i^k \overset{1}{<} \beta_j^k$; $\beta_i \overset{1}{\lessgtr} \beta_j$, если найдутся такие k и l, что $\beta_i^k < \beta_j^k$ и $\beta_i^l > \beta_j^l$. Например, сравнение векторов $\beta_1 = 00*11$, $\beta_2 = 011*0$, $\beta_3 = 0111*$ дает $\beta_1 \overset{1}{\lessgtr} \beta_2$, $\beta_1 \overset{1}{<} \beta_3$, $\beta_2 \overset{1}{=} \beta_3$. Заметим, что введенные отношения „$\overset{1}{=}$" и „$\overset{1}{<}$" не являются отношением эквивалентности и отношением порядка соответственно, поскольку не обладают свойством транзитивности. Интервал (α_i, β_i) реализуется интервалом (α_j, β_j), если $I_{\alpha_i} \subseteq I_{\alpha_j}$ и $\beta_i \overset{1}{=} \beta_j$. Интервал (α_i, β_i) частично реализуется интервалом (α_j, β_j), если $I_{\alpha_i} \subseteq I_{\alpha_j}$ и $\beta_i \overset{1}{>} \beta_j$.

В последнем случае будем считать, что реализован обобщенный интервал $(\alpha_i, \check{\beta}_i)$, вектор $\check{\beta}_i$ получен из β_i путем присвоения нулевых

значении всем тем его компонентам, которые соответствуют нулям в β_j, т. е. $\check{\beta}_i \overset{1}{=} \beta_j$. Например, интервалом $(\alpha_1, \beta_1) = (*10*, 11010)$ реализуется интервал $(\alpha_2, \beta_2) = (110*, 11010)$, так как $\beta_1 \overset{1}{=} \beta_2$ и $I_{\alpha_2} \subset I_{\alpha_1}$, $I_{\alpha_2} = \{1100, 1101\}$, $I_{\alpha_1} = \{0100, 1100, 0101, 1101\}$. Тот же интервал (α_1, β_1) частично реализует интервал $(\alpha_3, \beta_3) = (*100, 11011)$, так как $\beta_3 \overset{1}{>} \beta_1$ и $I_{\alpha_3} \subset I_{\alpha_1}$, $I_{\alpha_3} = \{0100, 1100\}$, при этом считается, что реализован обобщенный интервал $(\alpha_3, \check{\beta}_3) = (*100, 11010)$, т. е. осталось нереализованным единичное состояние функции f_0 на интервале $*100$. (Напомним, что нумерация функций в β идет справа налево.)

Уточним понятие цены обобщенного интервала. Пусть имеется множество обобщенных интервалов Ψ_0, реализующее систему булевых функций $F(X)$, и обобщенный для этой же системы интервал (α_i, β_i), который может и не принадлежать множеству Ψ_0. Пусть множество $\tilde{\Psi} \subset \Psi_0$, $\tilde{\Psi} = \{(\alpha_k, \beta_k)\}$ $(k = 1, \ldots, r)$ содержит интервалы, частично или полностью реализованные интервалом (α_i, β_i), т. е. $\beta_k \overset{1}{>} \beta_i$, $I_{\alpha_k} \subseteq I_{\alpha_i}$ для всех k. Цена интервала (α_i, β_i) равна суммарному числу единиц в выходных частях всех $(\alpha_k, \check{\beta}_k)$, $\check{\beta}_k \overset{1}{=} \beta_i$ $(k = 1, \ldots, r)$.

Пример 32. Пусть $(\alpha_i, \beta_i) = (*10*, 01110)$ и

$$\tilde{\Psi} = \begin{matrix} (\alpha_k, \beta_k) \\ \begin{cases} (1101, 11*10) \\ (0100, 011*0) \\ (0100, 01110) \end{cases} \end{matrix} \quad \begin{matrix} k \\ 1 \\ 2 \\ 3 \end{matrix}$$

Тогда $\check{\beta}_1 = 01*10$, $\check{\beta}_2 = 011*0$, $\check{\beta}_3 = 01110$. Суммируя единицы в этих векторах, получаем цену обобщенного интервала $(*10*, 01110)$, равною 7.

Будем считать, что удалить из Ψ_0 реализованный интервал $(\alpha_k, \check{\beta}_k)$ — это значит в выходной части интервала (α_k, β_k) заменить на звездочки $(*)$ все единицы, вошедшие в $\check{\beta}_k$. Так, множество $\tilde{\Psi}$ из примера 32, интервалы которого полностью или частично реализованы интервалом $(*10*, 01110)$, после удаления из него реализованных интервалов будет иметь вид

$$\tilde{\Psi} = \begin{cases} (1101, 1***0), \\ (0100, 0***0), \\ (0101, 0***0). \end{cases}$$

6.3.2. Алгоритмы определения максимальных интервалов для заданого порождающего

Пусть система *F (X)* булевых функций, в общем случае частичных, задана множеством Ψ_0 обобщенных интервалов. И пусть множество Ψ_0 таково, что входные части всех его элементов есть *n*-интервалы, которые мы условились обозначать через $a_i{:}\Psi_0 = \{(a_i,\ \beta_i)\}$ (i= 1,...,p_0).

Обозначим *I(F)* совокупность всех входных частей обобщенных интервалов, составляющих множество Ψ_0 : $I(F) = \{a_1,\ ...,\ a_{p_0}\}$.

Назначение одного из элементов множества Ψ_0 , пусть это $(a_h,\ \beta_h)$, на роль интервала, порождающего элемент решения Ψ_R, можно рассматривать как задание булевой функции $\hat{f}_h$ со следующими областями определенных значений:

$$\hat{I}_h^1 = \{a_i{:}a_i \in I(F) \text{ и } \beta_i \overset{1}{\geqslant} \beta_h\};$$

$$\hat{I}_h^0 = \{a_i{:}a_i \in I(F) \text{ и } \beta_i \overset{1}{<} \beta_h \text{ или } \beta_i \overset{*}{\geqslant} \beta_h\}, \qquad \textbf{(24)}$$

где $(a_i,\ \beta_i),\ (a_h,\ \beta_h) \in \Psi_0$. Заметим, что $\hat{f}_h \notin F$. Задание функций $\hat{f}_h$, равносильно выделению подсистемы $F_h \subset F$, в которую входят функции, равные единице на *n*-интервале a_h, т. е. функции, помеченные единицами в β_h . Все функции подсистемы F_h равны нулю на *n*-интервалах $a_i \in \hat{I}_h^0$ и единице на *n*-интервалах $a_i \in \hat{I}_h^1$.

(Вопрос о доопределении функций на *n*-интервале a_h, если в β_h имеются неопределенные компоненты $(*)$, как и вопрос о выборе самого порождающего интервала $(a_h,\ \beta_h)$ из исходного множества Ψ_0, отложим, а пока будем считать, что выходная часть порождающего интервала имеет только единицы и нули.)

Пример 33. Пусть система $F(X) = \{f_0,\ ...,\ f_4\}$, $X = \{x_0,\ ...,\ x_3\}$, задана множеством обобщенных интервалов Ψ_0

(α_i, β_i)	i
(0000, 11∗00)	1
(0001, 00111)	2
(0010, 11000)	3
(0011, ∗1001)	4
(0100, 1∗∗10)	5
(0110, 0∗011)	6
(1000, 11000)	7
(1001, 01011)	8
(1010, ∗∗110)	9
(1011, 1∗001)	10
(1100, 11100)	11
(1101, 110∗∗)	12
(1110, 00110)	13

Назначим интервал $(\alpha_3, \beta_3) = (0010, 11000)$ порождающим. Функция $\hat{f}_3$ имеет следующие область нулевых и единичных значений $\hat{T}_3^0 = \{0001, 0110, 1001, 1110\}$; $\hat{T}_3^1 = \{0000, 0010, 0011, 0100, 1000, 1010, 1011, 1100, 1101\}$. Подсистема F_3 содержит функции f_3, f_4.

Поиск максимального интервала. Задача поиска максимального интервала, удовлетворяющего следующим условиям: 1) максимального в области $I^{(n)} \setminus \hat{T}_h^0$, включающего в себя a_h и 2) имеющего наибольшую цену среди всех интервалов, удовлетворяющих условию 1, может быть решена как задача анализа структуры $\hat{\Theta}$ при заданной выражениями (24) булевой функции $\hat{f}_h$. Цены максимальных интервалов подсчитываются только с учетом функций подсистемы F_h. Напомним, что множество блоков $E^1_{\max}(a_h)$, максимальных в усеченной структуре $\Lambda(a_h)$ (множество интервалов, максимальных в области $I^{(n)} \setminus \hat{T}_h^0$ и включающих a_h), получается из разбиений структуры следующим образом. Из разбиений структуры $\hat{\Theta}$ удаляются все блоки, которые имеют непустое пересечение с $\hat{T}_h^0$, затем все блоки, не содержащие a_h, и из оставшихся блоков выделяют максимальные.

Таким образом, обобщенный интервал с наибольшей ценой выбирают из множества $\Psi'_h = \{(\alpha_{hj}, \beta_h)\}$, где α_{hj}—j-й максимальный интервал в $E^1_{\max}(a_h)$. Для того чтобы найти максимальные интервалы, составляющие множество $E^1_{\max}(a_h)$, будем строить строки $D(a_h)$ или $\Delta(a_h)$ (представления усеченной структуры $\Lambda(a_h)$) и из них

выделять эти максимальные интервалы. Строка D имеет 2^n клеток, которые нумеруются от 0 до 2^n—1. В строке $D(a_h)$ записаны n-интервалы, принадлежащие максимальным интервалам из $B^1_{\max}(a_h)$. Множество таких n-интервалов обозначено $I^1(a_h)$. Каждому n-интервалу $a_t \in I^1(a_h)$ отводится клетка строки $D(a_h)$ с двоичным номером

$$g_t = a_{h(2)} \oplus a_{t(2)} \quad (25)$$

(a_h стоит в клетке с номером нуль). Некоторые клетки строки $D(a_h)$ остаются пустыми (в них ставится 0), так как $|I^1(a_h)| < 2^n$. Строка $\Delta(a_h)$—это та же $D(a_h)$ после удаления из нее пустых клеток.

Для того чтобы при подсчете цен максимальных интервалов от строки $D(a_h)$ или $\Delta(a_h)$ не возвращаться к исходному множеству Ψ_0, будем записывать цену каждого обобщенного интервала $(a_t, \check{\beta}_t)$ прямо в строке $D(a_h)$ или $\Delta(a_h)$. В строке $D(a_h)$ это легко сделать, записав цену вместо самого a_i, а строку $\Delta(a_h)$ приходится дополнять второй строкой и в ней фиксировать цены интервалов $(a_t, \check{\beta}_t)$. В первой строке матрицы $\Delta(a_h)$ вместо $a_i \in I^1(a_h)$ записываем $g_t = a_{h(2)} \oplus a_{t(2)}$. Строку $D(a_h)$ с ценами и двустрочную матрицу $\Delta(a_h)$ условимся обозначать $D_{\text{ц}}(a_h)$ и $\Delta_{\text{ц}}(a_h)$. Клетку строки $D_{\text{ц}}(a_h)$ обозначим d_t $(i=0, \ldots, 2^n-1)$, клетку матрицы $\Delta_{\text{ц}}(a_h)$ — δ_{ij} $(i=1,2;\ j=1, \ldots, |I^1(a_h)|)$. Таким образом, $D_{\text{ц}}(a_h)$ и $\Delta_{\text{ц}}(a_h)$ несут одну и ту же информацию: цену интервалов $(a_t, \check{\beta}_t)$ и соответствующие им g-векторы, при этом g-векторы в строке $\Delta_{\text{ц}}(a_h)$ записаны в явном виде, а в строке $D_{\text{ц}}(a_h)$—в виде номера клетки. Если необходимо восстановить собственно n-интервал a_i (например, для того, чтобы удалить из Ψ_0 реализованный интервал), следует воспользоваться полученной из (25) формулой

$$a_{t(2)} = a_{h(2)} \oplus g_t \quad (26)$$

Описание алгоритмов построения $D_{\text{ц}}$ **и** $\Delta_{\text{ц}}$. Алгоритмы построения строки $D_{\text{ц}}(a_h)$ и матрицы $\Delta_{\text{ц}}(a_h)$ обозначим А1 и А2 соответственно. С помощью алгоритма А1 строится упорядоченное по g-векторам (см (25)) множество $I^1(a_h)$, для чего в соответствии с формулой (26) вектор $a_{h(2)}$ складывается по mod 2 поочередно со всеми g-вектрами длины от 0...0 до 1 ... 1. Полученным при каждом сложении «интервал a_i проверяется на принадлежность множеству $\hat{I}^0_h$. Если $a_t \in \hat{I}^0_h$, то булев интервал, включающий a_h, и a_i, не может принадлежать множеству $B^1_{\max}(a_h)$. Следовательно, g_i определяет запретную для a_h совокумность координат и принадлежит множеству $G''(a_h)$. Все $g_j > g_i$

также принадлежат множеству $G''(a_h)$. (Понятие запретной совокупности координат определено ранее, там же введены множества g-векторов $G'(a_h)$ и $G''(a_h)$.) Таким образом, если очередной вектор $g_i \in G''(a_h)$, то клетка строки $D_ц(a_h)$ с номером g_i объявляется «пустой», ей присваивается $\varnothing$, и такой же знак ставится во все клетки с номерами $g_j > g_i$. Если $a_i \notin \hat{I}_h^0$, то в клетку с номером g_i записывается цена обобщенного интервала $(a_i, \breve{\beta}_i)$.

Для того чтобы установить, принадлежит ли a_i множеству $\hat{I}_h^0$, в Ψ_0 отыскивается обобщенный интервал с входной частью, равной a_i. Если такой интервал найден, то его выходная часть β_i сравнивается с β_h. Если $\beta_i \overset{<}{\geq} \beta_h$ или $\beta_i \overset{!}{<} \beta_h$, то $a_i \in \hat{I}_h^0$, если $\beta_i \overset{!}{\geq} \beta_h$, то $a_i \notin \hat{I}_h^0$ (см. (24)). Если в Ψ_0 не оказалось интервала с входной частью a_i, то это значит, что на интервале a_i все функции системы не определены, и поэтому интервал a_i не вошел в множество $I(F)$. Цена такого интервала, естественно, равна нулю.

Чтобы отыскать n-интервал a_i в $I(F)$, можно применить метод дихотомии. Упорядочим обобщенные интервалы, образующие множество Ψ_0, по их входным частям, расположив n-интервалы в лексикографическом порядке. Интервал a_i отыскиваем методом деления пополам сначала всего списка, затем той его половины, в которой может оказаться a_i, той четверти и т. д. Чтобы узнать, в какой из двух частей может оказаться a_i, достаточно сравнить a_i с пограничными элементами частей. Таким способом a_i отыскивается в списке (или устанавливается, что $a_i \notin I(F)$) за r шагов ($r =]\log_2 p_0[$). Этот процесс в алгоритмах записывать не будем, чтобы не загромождать их второстепенными правилами.

Алгоритм А2 отличается от А1 тем, что если $a_i \in \hat{I}_h^0$, то соответствующий g-вектор фиксируется как принадлежащий множеству $G''(a_h)$, а для a_i в $\Delta_ц(a_h)$ естественно столбец не отводится. Столбцы в $\Delta_ц(a_h)$ не отводятся и тем a_i, которые соответствуют обобщенным интервалам с нулевой ценой. И вообще, на каждом новом витке алгоритма каждый g-вектор, соответствующий этому витку, сравнивается со всеми элементами множества $G''(a_h)$, и если этот g-вектор оказывается больше какого-либо из них, то g отбрасывается, и происходит переход к новому витку алгоритма. Таким образом, алгоритм А2 формирует $\Delta_ц(a_h)$ и $G''_{min}(a_h)$ — множество g-векторов, соответствующих минимальным запретным для a_h совокупностям координат.

В алгоритмах Al, A2 и последующих цена обобщенного интервала (a_i, β_i) обозначается $\rho(a_i, \beta_i)$, в зависимости от контекста *g* может быть и вектором, и двоичным числом. В алгоритмах используется знак «:=», $a := b$ означает, что переменной *a*, присваивается значение *b*.

Алгоритм А1

$g := 0,\ d_0 := \rho(a_h, \beta_h).$

1. $g := g + 1$. Если g=l ... 1, то выполняется п. 5. Если $d_g = \varnothing$, то повторяется п. 1.
2. $d_{g(2)} := a_{h(2)} \oplus g$. Если $d_g \notin I(F)$, то $d_g := 0$ и выполняется п. 1. Если $d_g \in I(F)$, пусть $d_g = a_q$, то проверяется условие $\beta_q \overset{1}{\geqslant} \beta_h$. Если условие выполнено, то совершается переход к п. 4.
3. Всем элементам d_j для которых верно, что $j_{(2)} \geqslant g$, присваивается «пустое» значение $d_j := \varnothing$. Выполняется п. 1.
4. $d_g := \rho(a_q, \breve{\beta}_q)$. Выполняется п. 1.
5. Конец.

Алгоритм А2

$g := 0,\ k := 0,\ l := 1$ (k — текущее значение мощности множества $\Delta_u(a_h)$), $\delta_{1l} := g$, $G''(a_h)$, l—текущий номер столбца матрицы $\delta_{2l} := \rho(a_h, \beta_h)$, $G''_{\min}(a_h) := \varnothing$.

1. $g := g + 1$. Если g=1...1, то выполняется п. 5. Если $k = 0$, то выполняется п. 2. Если $g > g_i$ $(i = 1, \ldots, k,\ g_i \in G''_{\min}(a_h))$, то g отбрасывается и выполняется п. 1.
2. $d_{(2)} := a_{h(2)} \oplus g$. Если $d \notin I(F)$, то выполняется п. 1. Если $d \in I(F)$, пусть $d = a_q$, то проверяется условие $\beta_q \overset{1}{\geqslant} \beta_h$. Если условие выполнено, то совершается переход к п. 4.
3. $k := k + 1$, $G''_{\min}(a_h) := G''_{\min}(a_h) \cup g$. Выполняется п. 1.
4. Если $\rho(a_q, \breve{\beta}_q) = 0$, то выполняется п. 1. В противном случае $l := l + 1,\ \delta_{1l} := g,\ \delta_{2l} := \rho(a_q, \breve{\beta}_q)$. Выполняется п. 1.
5. Конец.

Пример 34. Система булевых функций $F(X) = \{f_0, f_1, f_2, f_3, f_4\}$, $X = \{x_0, x_1, x_2, x_3\}$, задана множеством обобщенных интервалов Ψ_0

$(a_i,\ \beta_i)$	i
(0000, 11∗00)	1
(0001, 00111)	2
(0010, 11000)	3
(0011, ∗1001)	4
(0100, 1∗∗10)	5
(0110, 0∗011)	6
(1000, 11000)	7
(1001, 01011)	8
(1010, ∗∗110)	9
(1011, 1∗001)	10
(1100, 11100)	11
(1101, 110∗∗)	12
(1110, 00110)	13

Пусть интервал $(a_7,\ \beta_7) = (1000,\ 11000)$ -порождающий. Применим к $\{\Psi_0,\ (a_7, \bar{\beta}_7)\}$ алгоритм А1. $g := 0000,\ d_0 := \rho(a_7,\ \beta_7) = 2.$

1. $g := 0111,\ d_7 = \varnothing,\ g := 1000.$
2. $d_{8(2)} := 0000,\ a_1 = 0000,\ \beta_1 = \beta_7.$
4. $\rho(a_1,\ \tilde{\beta}_1) = 2,\ d_8 := 2.$
1. $g := 1001,\ d_9 = \varnothing,\ g := 1010.$
2. $d_{10(2)} := 1000 \oplus 1010,\ d_{10(2)} = 0010 = a_3,\ \beta_3 = \beta_7.$
4. $\rho(a_3,\ \tilde{\beta}_3) = 2,\ d_{10} := 2.$
1. $g := 1011,\ d_{11} = \varnothing,\ g := 1100.$
2. $d_{12(2)} := 1000 \oplus 1100,\ d_{12(2)} = 0100 = a_5,\ \beta_5 > \beta_7.$
4. $\rho(a_5,\ \tilde{\beta}_5) = 1,\ d_{12} := 1.$
1. $d_{13} = d_{14} = d_{15} = \varnothing.$
5. Конец.

Строка $D_{\text{ц}}(8)$ имеет вид

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$D_{\text{ц}}(8) = (2,\ \varnothing,\ 0,\ \varnothing,\ 2,\ \varnothing,\ \varnothing,\ \varnothing,\ 2,\ \varnothing,\ 2,\ \varnothing,\ 1,\ \varnothing,\ \varnothing,\ \varnothing).$

Пример 35. К $\{\Psi_0,\ (a_7,\ \beta_7)\}$ (Ψ_0 из примера 34) применим алгоритм А2. Не будем подробно описывать ход вычислений, так как он, за исключением нескольких моментов, совпадает с приведенным в примере 34. Результатом применения алгоритма А2 к $\{\Psi_0,\ (a_7,\ \beta_7)\}$ является матрица $\Delta_{\text{ц}}(8)$ и множество $G^{*}_{\min}(8)$:

$$\Delta_{\text{ц}}(8) = \begin{pmatrix} 0, & 4, & 8, & 10, & 12 \\ 2, & 2, & 2, & 2, & 1 \end{pmatrix};\quad G^{*}_{\min}(8) = \{0001,\ 0110\}.$$

Алгоритмы выделения максимального обобщенного интервала с наибольшей ценой. Выделение из $D_{\text{ц}}(a_h)$ или $\Delta_{\text{ц}}(a_h)$ всех

максимальных интервалов, содержащих a_h, сводится к отысканию наибольших g-векторов среди всех зафиксированных в $D_{\text{ц}}(a_h)$ или $\Delta_{\text{ц}}(a_h)$, т. е. к выделению множества $G'_{\max}(a_h) = \{g_1, \ldots, g_t\}, t = |G'_{\max}(a_h)|$. Каждый троичный вектор $\alpha_{hj} (j = 1, \ldots, t)$, соответствующий одному из максимальных интервалов, получается из двоичного вектора a_h, если в нем поставить звездочки в тех разрядах, которые равны единице в $g_j \in G'_{\max}(a_h)$.

Цена обобщенного максимального интервала (α_{hj}, β_n) $(j = 1, \ldots, t)$, равная сумме цен всех интервалов, реализованных этим максимальным, определяется путем суммирования содержимого «непустых» клеток строки $D_{\text{ц}}(a_h)$, таких, двоичные номера которых меньше g_j:

$$\rho(\alpha_{hj}, \beta_h) = \sum_{\substack{k=0, \\ k_{(2)} \leqslant g_j}}^{2^n-1} d_k, \quad (27)$$

или путем суммирования тех элементов второй строки матрицы $\Delta_{\text{ц}}(a_h)$, которым в первой строке соответствуют g-векторы $g_k \leqslant g_j$:

$$\rho(\alpha_{hj}, \beta_h) = \sum_{\substack{k=1 \\ \delta_{1k\,(2)} \leqslant g_j}}^{p_\Delta} \delta_{2k}, \quad (28)$$

где p_Δ,—число столбцов в $\Delta_{\text{ц}}(a_h)$.

Алгоритмы выдетеиия из $D_{\text{ц}}(a_h)$ и $\Delta_{\text{ц}}(a_h)$ максимальных обобщенных интервалов с наибольшей ценой назовем соответственно В1 и В2. Эти алгоритмы очень близки. Существенным моментом является порядок выделения максимальных g-векторов, а именно продвижение по $D_{\text{ц}}(a_h)$ или $\Delta_{\text{ц}}(a_h)$ справа палево. Такой порядок выборки обусловлен (как уже было замечено ранее) лексикографическим расположением g-векторов в матрицах D и Δ, при котором крайний справа g-вектор является первым максимальным, каждый следующий максимальный также оказывается крайним справа, если отбросить все столбцы матриц *D* и Δ, которые соответствуют g-векторам, меньшим предыдущего максимального. Результатом выполнения В1 и В2 является g-вектор, соответствующий максимальному обобщенному интервалу с наибольшей ценой.

Алгоритм В1

$q := 2^n$, $\rho_{\max} := 0$ (*q* - текущий номер элемента строки $D_{\text{ц}}(a_h)$, $\rho_{\max}$— максимальная цена).

1. *q*: = *q*—1. Если *q* = — 1, то выполняется п. 3. Если элемент d_q «пуст» или помечен, то повторяется п. 1.

2. Помечаются все «непустые» d_l, такие, что $l_{(2)} \leqslant q_{(2)}$.
В соответствии с (27) все вновь помеченные элементы суммируются, полученная цена присваивается переменной ρ. Если $\rho > \rho_{max}$, то $\rho_{max} := \rho$, $g := q_{(2)}$. Выполняется п. 1.

3. Конец.

Алгоритм В2

$q := p_{\Delta} + 1$, $\rho_{max} := 0$ (q — текущий номер столбца матрицы $\Delta_{ц}(a_h)$, ρ_{max}—максимальная цена).

1. $q := q - 1$. Если $q = 0$, то выполняется п. 3. Если элемент δ_{2q} помечен, то повторяется п. 1.

2. Помечаются все δ_{2l} такие, что $\delta_{1l(2)} \leqslant \delta_{1q(2)}$. В соответствии с (28) все вновь помеченные элементы второй строки суммируются, полученная цена присваивается переменной ρ. Если $\rho > \rho_{max}$, то $\rho_{max} := \rho$, $g := \delta_{1q(2)}$. Выполняется п. 1.

3. Конец.

Пример 36. Применим алгоритм В1 к строке

$$\begin{matrix} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{matrix}$$
$$D_{ц}(8) = (2, \varnothing, 0, \varnothing, 2, \varnothing, \varnothing, \varnothing, 2, \varnothing, 2, \varnothing, 1, \varnothing, \varnothing, \varnothing),$$

полученной в примере 34. Клетки с номерами 15,14,13 «пусты». Первая справа «непустая» клетка строки $D_{ц}(8)$ имеет номер 12, значит, первый максимальный g-вектор равен 1100. Помечаем клетки d_0, d_4, d_8, d_{12}, поскольку двоичные номера их меньше или равны двоичному числу 1100. Суммируя содержимое помеченных клеток, получаем: $\rho = 7$, $\rho_{max} = 7$, $g = 1100$. Следующая справа «непустая» и непомеченная клетка d_{10}, следовательно, второй максимальный g-вектор равен 1010. Помечаем клетки d_0, d_2, d_8, d_{10}, суммируем их содержимое, получаем ρ = 6. Максимальная цена остается прежней: $\rho_{max} = 7$ и $g = 1100$. Поскольку в строке $D_{ц}$ (8) не осталось ни одной «непустой» клетки, которая не участвовала бы в формировании цен, то алгоритм В1 выполнен, получен g-вектор 1100, который соответствует максимальному интервалу $**00$ $(a_h = 1000)$. В решение войдет максимальный обобщенный интервал $(**00, 11000)$.

Пример 37. Применим алгоритм В2 к матрице

$$\begin{matrix} & 1 & 2 & 3 & 4 & 5 & \\ \Delta_{ц}(8) := & \left[\begin{matrix} 0 & 4 & 8 & 10 & 12 \\ 2 & 2 & 2 & 2 & 1 \end{matrix}\right] & \begin{matrix} 1 \\ 2 \end{matrix} \end{matrix},$$

полученной в примере 35.

Первый максимальный g-вектор равен 1100 (12), цена максимального интервала $\rho = 7$ $(\rho = \delta_{25} + \delta_{23} + \delta_{22} + \delta_{21})$, второй максимальный g-вектор

1010 (10), $\rho=6$ ($\rho=\delta_{24}+\delta_{23}+\delta_{21}$).. Результат: максимальный g-вектор 1100. В Ψ_R войдет интервал $(**00,\ 11000)$.

Если множество Ψ_0 по мощности много меньше, чем множество всех *n*-интервалов *n*-мерного булева пространства, т. е. $p_0 \ll 2^n$, то построение $D_{\text{ц}}(a_k)$ и $\Delta_{\text{ц}}(a_h)$ по алгоритмам А1 и А2 связано с большим количеством ненужных вычислений. Приходится производить вычисления для *n*-интервалов, не принадлежащих множеству *I(F)* и поэтому являющихся входными частями обобщенных интервалов с нулевой ценой, а при $p_0 \ll 2^n$ мощность множества $I^{(n)} \setminus I(F)$ существенно больше, чем мощность множества *I(F).* Чтобы исключить такие вычисления, предложим другой алгоритм построения матрицы $\Delta_{\text{ц}}(a_h)$, в котором в отличие от алгоритма А2 в вычислениях участвуют только элементы множества *I(F).* Для построения матрицы $\Delta_{\text{ц}}(a_h)$ будем использовать формулу (25). Иными словами, для каждого $a_i \in I(F)$ будем вычислять g-вектор, затем по выходной части обобщенного интервала $(\alpha_i,\ \beta_i)$ определять, какой области ($\hat{I}_h^1$ или $\hat{I}_h^0$) принадлежит a_h, и в надлежащем случае записывать g-вектор и цену интервала $(\alpha_i,\ \hat{\beta}_i)$ в матрицу. Полученная в результате таких действий матрица $\Delta'_{\text{ц}}(a_h)$ отличается от $\Delta_{\text{ц}}(a_h)$ только порядком столбцов, так как информация в $\Delta'_{\text{ц}}(a_h)$ записывается не в порядке следования g-векторов от 0...0 до 1 ... 1, как это делается в алгоритмах А1 и А2, а в порядке следования интервалов в списке, представляющем множество Ψ_0. В новом алгоритме *n*-интервал a_i в множестве *I(F)* не отыскивается, зато каждый вновь найденный вектор $g_t \in G''(a_h)$ сравнивается со всеми g-векторами, зафиксированными в $\Delta'_{\text{ц}}(a_h)$,и с теми, что образуют отдельное множество, содержащее некоторые g-векторы из $G''(a_h)$. Все g-векторы, которые оказываются больше вновь найденного, отбрасываются. Таким образом, формируется множество $G^*_{\min}(a_h)$, а из $\Delta'_{\text{ц}}(a_h)$ удаляются столбцы, несущие информацию об *n*-интервалах $a_l \notin I^1(a_h)$. Алгоритм построения матрицы $\Delta'_{\text{ц}}(a_h)$ назовем А3.

Алгоритм А3

$j:=0,\ l:=0,\ k:=0$ (*j*, *I* — текущие номера элемента множества Ψ_0 и столбца матрицы $\Delta'_{\text{ц}}(a_h)$ соответственно; *k* — текущее значение мощности множества $G^*_{\min}(a_h)$, $G^*_{\min}(a_h):=\varnothing$).

1. $j:=j+1$. Если $j>p_0$, то выполняется п. 6.

2. $g := a_{h(2)} \oplus a_{l(2)}$ Если $k = 0$, то выполняется п. 3. Если $g > g_i$, где $g_i \in G''_{\min}(a_h)$ $(i = 1, \ldots, k)$, то g отбрасывается и выполняется п. 1.

3. Если $\beta_j \geq \beta_k$, то выполняется п. 5. Если $k = 0$, то выполняется п.

4. Все $g_i \in G''_{\min}(a_h)$ $(i = 1, \ldots, k)$, такие, что $g_i > g$, удаляются из $G''_{\min}(a_h)$ с соответствующим изменением k.

4. $k := k + 1$, $G''_{\min}(a_h) := G''_{\min}(a_h) \bigcup g$. Если $l = 0$, то выполняется п. 1. Все столбцы матрицы $\Delta'_{\text{ц}}(a_h)$, такие, что $\delta'_{1i(2)}$, удаляются с соответствующим изменением l, выполняется п. 1.

5. Если $\rho(a_j, \check{\beta}_j) = 0$, то выполняется п. 1. Иначе $l := l + 1$, $\delta'_{1l} := g$, $\delta'_{2l} := \rho(a_j, \check{\beta}_j)$. Выполняется п. 1.

6. Конец.

Пример 38. Применим алгоритм А3 к $\{\Psi_0, (a_7, \beta_7)\}$ из примера 34.

1. $j:=1$.
2. $g:=\underset{1}{1000} \oplus 0000$, $g=1000$.
3. $\beta_1=\beta_7$.
5. $\rho(a_1, \tilde{\beta}_1)=2$, $l:=1$, $\delta'_{11}:=8$, $\delta'_{21}:=2$.
1. $j:=2$.
2. $g:=1000 \oplus 0001$, $g=1001$.
3. $\beta_2 \overset{\succ}{\prec} \beta_7$.
4. $k:=1$, $G^*_{\min}(8):=\{1001\}$, $g_1=1001$.
1. $j:=3$.
2. $g:=\underset{1}{1000} \oplus 0010$, $g=1010$.
3. $\beta_3=\beta_7$.
5. $\rho(a_3, \tilde{\beta}_3)=2$, $l:=2$, $\delta'_{12}:=10$, $\delta'_{22}:=2$.
1. $j:=4$.
2. $g:=1011$, $g>g_1$, g отбрасывается.
1. $j:=5$.
2. $g:=\underset{1}{1000} \oplus 0100$, $g=1100$.
3. $\beta_5>\beta_7$.
5. $\rho(a_5, \tilde{\beta}_5)=1$, $l:=3$, $\delta'_{13}:=12$, $\delta'_{23}:=1$.
1. $j:=6$.
2. $g:=1000 \oplus 0110$, $g=1110$.
3. $\beta_6 \overset{\succ}{\prec} \beta_7$.
4. $k:=2$, $g_2:=1110$, $G^*_{\min}(8):=\{1001, 1110\}$.
1. $j:=7$.
2. $g:=\underset{1}{1000} \oplus 1000$, $g=0000$.
3. $\beta_7=\beta_7$.
5. $\rho(a_7, \beta_7)=2$, $l:=4$, $\delta'_{14}:=0$, $\delta'_{24}:=2$.
1. $j:=8$.
2. $g:=1000 \oplus 1001$, $g=0001$.
3. $\beta_8 \overset{\succ}{\prec} \beta_7$, $g<g_1$, g_1 удаляется из $G^*_{\min}(8)$, $k:=k-1$, $k=1$, $g_1:=1110$.
4. $k:=k+1$, $k=2$, $g_2=0001$, $G^*_{\min}(8):=\{1110, 0001\}$.
1. $j:=9$.
2. $g:=1000 \oplus 1010$, $g=0010$.
3. $\beta_9>\beta_7$.
5. $\rho(a_9, \tilde{\beta}_9)=0$.
1. $j:=10$.

2. $g:=1000 \oplus 1011$, $g=0011$, $g>g_2$, g отбрасывается.
1. $j:=11$.
2. $g:=1000 \oplus 1100$, $g=0100$.
3. $\beta_{11}>\beta_7$.
5. $\rho(a_{11}, \check{\beta}_{11})=2$, $l:=5$, $\delta'_{15}:=4$, $\delta'_{25}:=2$.
1. $j:=12$.
2. $g:=1000 \oplus 1101$, $g=0101$, $g>g_2$, g отбрасывается.
1. $j:=13$.
2. $g:=1000 \oplus 1110$, $g=0110$.
3. $\beta_{13} \not> \beta_7$, $g<g_1$, g_1 удаляется из $G''_{\min}(8)$, $k:=k-1$, $k=1$, $g_1:=0001$.
4. $k:=k+1$, $k=2$, $g_2:=0110$, $G'_{\min}(8):=\{0001, 0110\}$.
1. $j:=14$, $j>p_0$ $(p_0=13)$.

6. Конец.

В результате вычислений получили

$$\begin{bmatrix} 8 & 10 & 12 & 0 & 4 \\ 2 & 2 & 1 & 2 & 2 \end{bmatrix}; \quad G''_{\min}(8)=\{0001,\ 0110\}.$$

Для выделения из матрицы $\Delta'_{\text{ц}}(a_h)$ максимального обобщенного интервала с наибольшей ценой упорядочим столбцы ее по элементам первой строки от меньшего к большему. В результате получим матрицу $\Delta_{\text{ц}}(a_h)$, к которой можно применить алгоритм В2.

Опишем еще один алгоритм построения всех максимальных обобщенных интервалов для заданного порождающего. Множество входных частей всех $(\alpha_{hj}, \beta_h) \in \Psi'_h$ можно представить просто множеством $G'_{\max}(a_h)$, если отказаться от запоминания цен интервалов. В этом случае, чтобы определить цены максимальных обобщенных интервалов (α_{hj}, β_h), нужно обратиться к множеству Ψ_0 и узнать цену каждого $(a_k, \check{\beta}_k) \in \Psi_0$, реализованного интервалом (α_{hj}, β_h). Соответствующий алгоритм вычисления цен будет продемонстрирован в примере 39. Алгоритм А4 построения $G'_{\max}(a_h)$ и $G''_{\min}(a_h)$ состоит в том, чтобы решить, к какому из множеств: $G'(a_h)$ или $G''(a_h)$ принадлежит вектор, вычисленный по формуле (25), и, придавая этот вектор тому или иному множеству, удалять из $G'(a_h)$ немаксимальные, а из $G''(a_h)$ неминимальные векторы, формируя тем самым $G'_{\max}(a_h)$ и $G''_{\min}(a_h)$.

Алгоритм А4

$j:=0$, $k:=0$, $l:=0$ (j — текущий номер элемента множества Ψ_0, k, l — текущие значения мощностей множеств $G''_{\min}(a_h)$ и $G'_{\max}(a_h)$ соответственно).

1. $j:=j+1$. Если $j>p_0$, то выполняется п. 7.
2. $g:=a_{h(2)} \oplus a_{l(2)}$. Если $k=0$, то выполняется п. 3. Если $g>g_i$, где $g_i \in G''_{\min}(a_h)$ $(i=1, \ldots, k)$, то g отбрасывается и выполняется п. 1.
3. Если $\beta_j \overset{1}{\geqslant} \beta_h$, то выполняется п. 5. Если $k=0$, то выполняется п. 4. Все $g_i \in G''_{\min}(a_h)$ $(i=1, \ldots, k)$, такие, что $g>g_i$, удаляются с соответствующим изменением k.
4. $k:=k+1$. $G''_{\min}(a_h):=G''_{\min}(a_h) \cup g$. Если $l=0$, то выполняется п. 1. Все $g_i \in G'_{\max}(a_h)$ $(i=1, \ldots, l)$, такие, что $g_i>g$, удаляются с соответствующим изменением l. Выполняется п. 1.
5. Если $l=0$, то выполняется п. 6. Если $g<g_i$, где $g_i \in G'_{\max}(a_h)$ $(i=1, \ldots, l)$, то g удаляется и выполняется п. 1. Все $g_i \in G'_{\max}(a_h)$ $(i=1, \ldots, l)$, такие, что $g_i<g$, удаляются с соответствующим изменением l.
6. $l:=l+1$, $G'_{\max}(a_h):=G'_{\max}(a_h) \cup g$. Выполняется п. 1.
7. Конец.

Пример 39. Применим алгоритм А4 к $\{\Psi_0, (a_7, \beta_7)\}$ из примера34.

1. $j:=1$.
2. $g:=1000 \overset{1}{\oplus} 0000$, $g=1000$.
3. $\beta_1=\beta_7$.
6. $l:=l+1$, $l=1$, $g'_1:=1000$, $G'_{\max}(8):=\{1000\}$.
1. $j:=2$.
2. $g:=1000 \overset{1}{\oplus} 0001$, $g=1001$.
3. $\beta_6 \overset{<}{>} \beta_7$.
4. $k:=k+1$, $k=1$, $g''_1:=1001$, $G''_{\min}(8):=\{1001\}$.
1. $j:=3$.
2. $g:=1000 \overset{1}{\oplus} 0010$, $g=1010$.
3. $\beta_3=\beta_7$.
5. $g_1<g$, g'_1 удаляется из $G'_{\max}(8)$, $l:=l-1$, $l=0$, $G'_{\max}(8):=\varnothing$.
6. $l:=l+1$, $l=1$, $g'_1:=1010$, $G'_{\max}(8):=\{1010\}$.
1. $j:=4$.
2. $g:=1000 \oplus 0011$, $g=1011$, $g>g_1$. g удаляется.
1. $j:=5$.
2. $g:=1000 \overset{1}{\oplus} 0100$, $g=1100$.
3. $\beta_5>\beta_7$.

6. $l:=l+1$, $l=2$, $g_2':=1100$, $G'_{\max}(8):=\{1010,\ 1100\}$.
1. $j:=6$.
2. $g:=1000\oplus 0110$, $g=1110$.
3. $\beta_6 \overset{\succ}{\prec} \beta_7$.
4. $k:=k+1$, $k=2$, $g_2'':=1110$, $G''_{\min}(8):=\{1001,\ 1110\}$.
1. $j:=7$.
2. $g:=0000$.
3. $\beta_7=\beta_7$.
5. $g<g_1$ g отбрасывается.
1. $j:=8$.
2. $g:=1000\oplus 1001$, $g=0001$.
3. $\beta_8 \overset{\succ}{\prec} \beta_7$, $g_1''>g$, g_1'' удаляется из $G''_{\min}(8)$, $k:=k-1$, $k=1$, $G''_{\min}(8):=\{1110\}$, $g_1'':=1110$.
4. $k:=k+1$, $k=2$, $g_2'':=0001$, $G''_{\min}(8):=\{1110,\ 0001\}$.
1. $j:=9$.
2. $g:=1000\oplus 1010$, $g=0010$.
3. $\beta_9>\beta_7$.
5. $g<g_1''$, g удаляется.
1. $j:=10$.
2. $g:=1000\oplus 1011$, $g=0011$, $g>g_2''$, g удаляется.
1. $j:=11$.
2. $g:=1000\oplus 1100$, $g=0100$.
3. $\beta_{11}>\beta_7$.
5. $g<g_2''$, g удаляется.
1. $j:=12$.
2. $g:=1000\oplus 1101$, $g=0101$, $g>g_2''$, g удаляется.
1. $j:=13$.
2. $g:=1000\oplus 1110$, $g=0110$.
3. $\beta_{13} \overset{\succ}{\prec} \beta_7$, $g<g_1''$ из $G''_{\min}(8)$ g_1 удаляется, $k:=k-1$, $k=1$, $g_1''=0001$, $G''_{\min}(8):=\{0001\}$.
4. $k:=k+1$, $k=2$, $g_2'':=0110$, $G''_{\min}(8):=\{0001,\ 0110\}$.
1. $j:=14$, $j>p_0$.

7. Конец.

Таким образом, сформированы два множества g-векторов: $G'_{\max}(8)=\{1010,\ 1100\}$. $G''_{\min}(8)=\{0001,\ 0110\}$. Это значит, что найдены два максимальных обобщенных интервала, включающих интервал $(a_7,\ \beta_7)=(1000,\ 11000)$, а именно $(\tilde{\alpha}_1,\ \beta_7)=(*0*0,\ 11000)$ и $(\alpha_2,\ \beta_7)=(**00,\ 11000)$. Какой из них включить в решение Ψ_R, можно решить только после того, как будут вычислены цены этих интервалов. Для вычисления цены каждого максимального обобщенного интервала сравним его со всеми элементами множества Ψ_0 и найдем интервалы, реализованные максимальным. Сравнение про-

водим сначала по входным частям. Если входная часть максимального интервала (α_j, β_7) $(j=1,2)$ содержит входную часть некоторого $(a_k, \beta_k) \in \Psi_0$, т. е. $I_{a_k} \subset I_{\alpha_j}$, то сравниваются выходные части. Если $\beta_k \overset{1}{\leqslant} \beta_7$, то определяем вклад интервала (a_k, β_k) в цену $\rho(\alpha_j, \beta_7)$. Так, интервал $(*0*0, 11000)$ полностью или частично реализует следующие интервалы из множества Ψ_0: (a_1, β_1), (a_3, β_3), (a_7, β_7), (a_9, β_9), вклад в $\rho(*0*0, 11000)$ первых трех равен 2, последнего 0. Суммируем все вклады, получаем $\rho(\alpha_1, \beta_7)=6$. Проделаем то же самое для интервала (α_2, β_7): (a_1, β_1), 2; (a_5, β_5), 1; (a_7, β_7), 2; (a_{11}, β_{11}), 2; $\rho(\alpha_2, \beta_7)=7$.

В решение войдет интервал $(**00, 11000)$. Тоже самое получилось в примерах 36 и 37. 14.

6.3.3. Два замечания

Замечание 1. Алгоритмы А3 и А4 после небольшой модификации могут работать с исходным множеством обобщенных интервалов $\Psi_0 = \{(\alpha_l, \beta_l)\}$, входные части которых могут быть как двоичными, так и троичными векторами. В модифицированных алгоритмах вместо сложения по mod 2 выполняются следующие операции:

$$\alpha_h^k \overset{\max}{\bigoplus} \alpha_j^k = \begin{cases} 1, \text{ если } \alpha_h^k = * \wedge \alpha_j^k = * \vee \alpha_h^k \neq \alpha_j^k, \\ 0 \text{ во всех других случаях,} \end{cases}$$

$$\alpha_h^k \overset{\min}{\bigoplus} \alpha_j^k = \begin{cases} 1, \text{ если } \alpha_h^k = * \wedge \alpha_j^k = * \vee \alpha_h^k \neq \alpha_j^k \wedge \alpha_j^k \neq *, \\ 0 \text{ во всех других случаях,} \end{cases}$$

где $\alpha_h^k (\alpha_j^k)$ — k-й разряд вектора $\alpha_h (\alpha_j)$. Например,

$$**110 \overset{\max}{\bigoplus} *001* = 11101, \quad **110 \overset{\min}{\bigoplus} *001* = 11100.$$

Заметим, что операция $\bigoplus$ дает возможность найти минимальный объемлющий интервал для α_h и α_j. При $\beta_j \overset{1}{\geqslant} \beta_h$ вектор, получившийся в результате операции $\overset{\min}{\bigoplus}$, своими единицами определяет минимальную запретную для a_h совокупность координат. Операция $\overset{\min}{\bigoplus}$ неперестановочная.

В конце главы приведен модифицированный алгоритм А4, модификация алгоритма А3 может быть сделана аналогично. После модификации пп. 2—4 алгоритма А4 таковы:

2. Если $\beta_j \geq \beta_h$, то выполняется п. 4. $g := \alpha_h \overset{\min}{\bigoplus} \alpha_j$. Если $k = 0$, то выполняется п. 3. Если $g \geqslant g_i$, где $g_i \in G''_{\min}(\alpha_h)$ $(i = 1, 2, ..., k)$, g отбрасывается и выполняется п. 1. Все $g_i \in G''_{\min}(\alpha_h)$ $(i = 1, 2, ..., k)$, такие, что $g_i > g$, удаляются из $G''_{\min}(\alpha_h)$ с соответствующим изменением *k*.

3. $k := k + 1$, $G''_{\min}(\alpha_h) := G''_{\min}(\alpha_h) \bigcup g$. Если $l = 0$, то выполняется в п. 1. Все $g_i \in G'_{\max}(\alpha_h)$ $(i = 1, 2, ..., l)$, такие, что $g_i \geqslant g$, удаляются из $G'_{\max}(\alpha_h)$ с соответствующим изменением *l*. Выполняется в п. 1.

4. $g := \alpha_h \overset{\max}{\bigoplus} \alpha_j$. Если $k = 0$, то выполняется п. 5. Если $g \geqslant g_i$, где $g_i \in G''_{\min}(\alpha_h)$ то *g* отбрасывается и выполняется п. 1.

Замечание 2. Множество $G'_{\max}(\alpha_h)$, которое строится неявно с помощью алгоритмов А2, А3 и явно с помощью алгоритма А4, может содержать g-векторы, не максимальные в $G'(a_h)$. Причиной этого является то, что алгоритм А2 работает только с *n*-интервалами, имеющими ненулевую цену, а А3 и А4 — только с *n*-интервалами, принадлежащими области *I(F)*. И может оказаться, что среди g-векторов, соответствующих *n*-иптервалам с нулевой ценой или принадлежащих области $I^n/I(F)$, останется максимальный (или несколько максимальных). Используя множество $G''_{\min}(a_h)$, нетрудно проверить, действительно ли каждый вектор $g_i \in G'_{\max}(a_h)$ максимальный в $G'(a_h)$, и если нет, то найти максимальный. Проверка основана на том, что булев вектор $\overline{g_i}$, полученный из $g_i \in G'_{\max}(a_h)$ инвертированием всех его компонентов, соответствует одному из безызбыточных столбцовых покрытий булевой матрицы *Q*, образованной g-векторами $g_j \in G''_{\min}(a_h)$, каждый из которых записан в одной строке (о взаимосвязи g-векторов множеств $G'_{\max}(a_h)$ и $G''_{\min}(a_h)$). Подмножество столбцов матрицы *Q*, каждый из которых соответствует единице в $\overline{g_i}$, является безызбыточным покрытием матрицы *Q*.

Очевидное достаточное условие, с помощью которого можно проверить, соответствует ли $\overline{g_i}$ безызбыточному покрытию (или является ли g_i максимальным в $G'(a_h)$), состоит в следующем: для каждого единичного компонента g_i^t вектора g_i в $G''_{\min}(a_h)$ отыщется вектор g_j (строка в матрице Q), такой, что

$$\overline{g_i} \wedge g_j = 0 \ldots 0 \underset{t}{1} 0 \ldots 0. \quad (29)$$

Это означает, что для каждого столбца, принадлежащего безызбыточному покрытию матрицы *Q*, в *Q* найдется строка, которая покрыта одним этим столбцом. Если условие (29) не выполнено, будем решать задачу поиска столбцового безызбыточного покрытия (пользуясь, например, методами для матрицы Q', которая получается из матрицы Q следующим образом. Из матрицы вычеркиваются две группы столбцов. Столбцам первой группы соответствуют нули в $\overline{g}_i$, второй группы — те единицы вектора $\overline{g}_i$, для которых выполнялось условие (29). Из матрицы Q вычеркиваются все строки, на пересечении которых со столбцами второй группы стоят единицы. Столбцы второй группы составляют часть безызбыточного покрытия, и строки, покрытые этой группой столбцов, вычеркнуты.

Обозначим B0 алгоритм, который проверяет соответствие каждого $g_i \in G'_{\max}(a_h)$ безызбыточному столбцовому покрытию матрицы *Q* и строит такое покрытие, если проверка дает отрицательный ответ. Вообще говоря, множество всех g-векторов, максимальных в $G'(a_h)$, могло бы быть найдено путем построения всех безызбыточных столбцовых покрытий матрицы Q . Мы эту задачу упростили, сведя ее к решению задачи покрытия для нескольких матриц существенно меньших размеров. Однако заметим, что если не стремиться к тому, чтобы входные части всех $(\gamma_{hi}, \beta_h) \in \Psi'_h$ были максимальными интервалами в множестве $\Upsilon'_h \cup \Upsilon''_h$ (такое стремление часто не приводит к существенному улучшению качества решения задачи минимизации), то алгоритм B0 можно опустить. Если разработчик все-таки сочтет необходимым выполнить алгоритм B0, то его следует включить как самостоятельный п. 0 в алгоритм B2. Алгоритм B0 выполняется для каждого $g_i \in G'_{\max}(a_h)$.

Алгоритм B0

j: = 0 (*j* — текущий номер элемента множества $G''_{\min}(a_h)$).

1. *j*: = *j*1. Если $j > |G''_{\min}(a_h)|$, то выполняется п. 3.

2. $z := \overline{g}_i \wedge g_j$. Если вес вектора *z* больше единицы, то выполняется п. 1. В противном случае столбец, соответствующий единственной единице в *z* (пусть это *k-й* столбец), помечается. Из матрицы Q вычеркивается *j-я* строка и вместе с ней все строки, на пересечении которых с *k-м* столбцом стоит единица. Выполняется п. 1.

3. Из матрицы Q вычеркиваются все столбцы, соответствующие нулям в $\overline{g}_i$, и все столбцы, оказавшиеся помеченными. Если все

столбцы из Q удалены (а это значит, что $\overline{g}_t$ соответствует безызбыточному покрытию Q), то выполняется п. 6.

4. Определяются все безызбыточные столбцовые покрытия матрицы Q', полученной из Q после выполнения пп. 1—3.

5. Каждое покрытие матрицы Q' объединяется с подмножеством помеченных ранее в Q столбцов. Результатом объединения является безызбыточное покрытие матрицы Q, а соответствующий покрытию булев вектор является инверсией g-вектора, максимального в $G'(a_h)$.

6. Конец.

Пример 40. Применим алгоритм В0 к $\{Q, \overline{g}\}$:

$$Q = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 1 & 1 & 0 \\ 5 & 1 & 0 & 0 & 0 & 1 & 0 \\ 6 & 0 & 0 & 1 & 1 & 0 & 0 \\ 7 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}, \quad \overline{g} = 110111.$$

1, 2. $j:=1$; $z:=110100 \wedge 110111$, $z=110100$;

$j:=2$, $z:=101001 \wedge 110111$, $z=100001$;

$j:=3$, $z:=010010 \wedge 110111$, $z=010010$;

$j:=4$, $z:=000110 \wedge 110111$, $z=000110$;

$j:=5$, $z:=100010 \wedge 110111$, $z=100010$;

$j:=6$, $z:=000100$

Помечается четвертый столбец. Вычеркивается шестая строка и вместе с ней первая.

$j:=7$, $z:=010000$.

Помечается второй столбец. Вычеркивается седьмая строка и вместе с ней третья.

3. Из матрицы Q вычеркиваем третий столбец (нуль в векторе $\overline{g}$), второй и четвертый столбцы (помеченные).

4. После вычеркиваний получаем матрицу

$$Q' = \begin{array}{c|ccc} & 1 & 5 & 6 \\ \hline 1 & 1 & 0 & 1 \\ 2 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 \end{array},$$

которая имеет два безызбыточных столбцовых покрытия: {1,5} и {5,6}.
5. Объединив каждое из найденных покрытий с подмножеством помеченных столбцов — {2,4}, получим два безызбыточных покрытия матрицы Q: {1,2,4,5} и {2,4,5,6}. Соответствующие этим покрытиям g-векторы равны 001001 и 101000.

6.3.4. Выбор порождающего интервала

Обсудим вопрос выбора из множества Ψ_0 порождающего интервала и формирования на его основе выходной части вносимого в решение максимального обобщенного интервала. Естественно строить решение Ψ_R начиная с порождающего интервала (a_h, β_h), который имеет выходную часть β_h наименьшего веса, т. е. *n*-интервал a_h обобщен на наименьшее число функций (весом троичного вектора, как и двоичного будем считать число единиц в нем). Если в множестве Ψ_0 несколько обобщенных интервалов, выходные части которых имеют одинаковые и притом наименьшие веса, то в качестве порождающего выгоднее всего с точки зрения уменьшения объема вычислений брать интервал, имеющий наименьшее число соседей в Ψ_0. (Соседом интервала (a_h, β_h) будем считать (a_i, β_i), такой, что a_h и a_i— соседние *n*-интервалы и $a_i \notin \hat{T}_h^0$, где $\hat{T}_h^0$— область нулевых значений функции $\hat{f}_h$, определенной для (a_h, β_h) по формуле (24).)

Основанием для такого выбора являются следующие соображения. Во-первых, в строке $D_{\text{ц}}(a_h)$ будет тем больше «пустых» элементов (а в $\Delta_{\text{ц}}(a_h)$ тем меньше столбцов), чем меньше соседей имеет *n*-интервал a_h в области $\hat{T}_h^1 \bigcup \hat{T}_h^*$. Действительно, если a_h не имеет соседа по *j*-й координате, а это значит, что вектор 0... 010... 0 принадлежит множеству $G_{\min}^*(a_h)$, все элементы d_k из $D_{\text{ц}}(a_h)$, для которых $k_{(2)} \geqslant 0 \ldots \underset{j}{1} \ldots 0$, «пусты», а это половина $D_{\text{ц}}(a_h)$. Если a_h не имеет соседа еще и по *l*-й координате, то значит «пусто» три четверти строки $D_{\text{ц}}(a_h)$ и т. д. Отсюда следует, что чем меньше соседей имеет порождающий интервал (a_h, β_h), тем меньше усилий затрачивается на построение *D* или Δ-представления усеченной структуры $\Lambda(a_h)$. Во-вторых, чем меньше соседей имеет *n*-интервал в области $\hat{T}_h^1 \bigcup \hat{T}_h^*$, тем меньшим числом максимальных в ней интервалов он представлен. Отсюда следует, что, выбирая порождающие с учетом числа соседей,

легче сделать весь процесс более целенаправленным. Заметим, что выбор порождающего интервала с учетом числа соседей более существен при $p_0 \approx 2^n$, чем при $p_0 \ll 2^n$ (т. е. для системы слабоопределениых функций), когда почти все соседи нескольких интервалов, претендующих на роль порождающего, могут оказаться в области $\hat{I}_n^*$.

Если порождающий интервал назначен и его выходная часть $\beta_{h\,t}$ имеет неопределенные компоненты $(*)$, возникает вопрос об их доопределении, поскольку интервал, вносимый в решение, должен иметь в выходной части только единицы и нули. Понятно, что выгодно так доопределить вектор β_h, чтобы повысить цену интервала, вносимого в решение. Приведем несколько вариантов доопределения β_h. Первый вариант самый простой, но не гарантирует максимально возможной цены интервала, вносимого в решение. Третий вариант дает эти гарантии, но зато требует существенного дополнительного перебора.

Вариант 1. Выходной частью порождающего интервала является вектор γ_0, полученный из β_h присвоением нулевого значения всем неопределенным компонентам. Нужно найти максимальный интервал с наибольшей ценой, пусть это $(\alpha_{hj},\ \gamma_0)$, затем попытаться расширить выходную часть этого интервала за счет неопределенных значений в β_h. Под расширением выходной части будем понимать увеличение в ней числа единиц, т. е. увеличение числа функций системы, единичные состояния которых реализуются максимальным обобщенным интервалом. Расширенную выходную часть будем обозначать через γ_j. Если цена расширенного интервала $(\alpha_{hj},\ \gamma_j)$ увеличилась, то он включается в решение Ψ_R. Если же расширение невозможно или возможно только без увеличения цены соответствующего интервала, то в решение включается интервал $(\alpha_{hj},\ \gamma_0)$. Для расширения выходной части интервала нужно, $(\alpha_{hj},\ \beta_h)$ сравнить с интервалами из Ψ_0.

Вариант 2. Проверить возможность расширения выходных частей для всех максимальных интервалов $(\alpha_{hj},\ \gamma_0)$ независимо от их цен. В решение вносится максимальный обобщенный интервал, имеющий наибольшую цену с учетом расширения.

Вариант 3. Решить задачу нахождения максимального обобщенного интервала с наибольшей ценой для каждого порождающего из множества $\{(a_h,\ \gamma_0),$ где $\gamma_0, \ldots, \gamma_{2^r-1}$ —все возможные $(a_h,\ \gamma_1), \ldots, (a_h,\ \gamma_{2^r-1})\}$. варианты доопределения выходной части β_h, содержащей r неопределенных компонентов, и в решение внести

интервал (α_{hj}, γ_j), имеющий абсолютно максимальную цену среди всех найденных интервалов. В таком случае алгоритмы А1 (А2, А3, А4) и Bl (В2) придется выполнить 2^r раз.
Заметим, что использование варианта 2 в большинстве случаев должно давать очень хорошее приближение к точному решению задачи нахождения максимального обобщенного интервала с наибольшей ценой для заданного порождающего. Такая уверенность основана на том, что в испытании участвует достаточно большое число обобщенных интервалов.

6.3.5. Схемы алгоритмов приближенной минимизации

Теперь, когда подробно описаны основные части алгоритмов минимизации системы булевых функций *F(X),* приведем эти алгоритмы целиком. Предлагаются две пары алгоритмов: $\Sigma 1$, $\Sigma 2$ и $\Sigma 3$, $\Sigma 4$. Второй алгоритм каждой пары является модификацией первого и отличается от него тем, что имеет меньшую емкостную (по памяти ЭВМ) и несколько большую временную сложность. Алгоритмы Σ1 и Σ3 различаются между собой способами построения компактного представления усеченной структуры $\Lambda(a_h)$, т. е. строки $D_{\text{ц}}(a_h)$ и матрицы $\Delta_{\text{ц}}(a_h)$. В алгоритмах первой пары строка $D_{\text{ц}}(a_h)$ или матрица $\Delta_{\text{ц}}(a_h)$ строятся путем сравнения порождающего a_h с *n*-интервалами, выстроенными в лексикографическом порядке. Это значит, что с a_h сравниваются *n*-интервалы, как принадлежащие, так и не принадлежащие области *I(F).* В алгоритмах второй пары матрица $\Delta_{\text{ц}}(a_h)$ или множество $G'_{\max}(a_h)$ строятся путем сравнения a_h только с *n*-интервалами из множества *I(F).* Отсюда следует, что при $|I(F)| \gg 2^n$ имеет смысл использовать алгоритмы $\Sigma 3$, $\Sigma 4$, а $\Sigma 1$, $\Sigma 2$ для случая, когда $|I(F)| \approx 2^n$, и для систем функций от небольшого (n<12) числа переменных.
Шагом любого алгоритма будем считать все действия, которые необходимо выполнить, чтобы получить один обобщенный интервал, вносимый в решение. Исходной формой задания системы *F(X)* для алгоритмов $\Sigma 1$, $\Sigma 2$ является множество обобщенных интервалов, входными частями которых являются *n*-интервалы, для алгоритмов $\Sigma 3$, $\Sigma 4$— множество Ψ_0, элементы которого в качестве входных частей могут иметь как двоичные, так и троичные векторы.

Множество обобщенных интервалов, получаемое в результате работы одного из алгоритмов $\Sigma 1 - \Sigma 4$, обозначим Ψ_R. Множество, полученное из Ψ_0 после удаления из него реализованных за t шагов обобщенных интервалов, обозначим Ψ_{0t}. Будем считать, что, выполнив шаг t любого алгоритма, получим множество обобщенных интервалов Ψ_t мощности p_t, являющееся подмножеством решения Ψ_R, и множество Ψ_{0t}, в котором может находиться p_{0t} нереализованных обобщенных интервалов (p_{0t} равно числу интервалов в Ψ_{0t}, в выходных частях которых есть хотя бы одна единица). Эта значит, что если остановить работу алгоритма на t-м шаге, то система $F(X)$ может быть представлена матрицей **M**, имеющей $p_t + p_{0t}$ строк.

Будем считать, что решение Ψ_R получено, если выполнено одно из следующих двух условий: 1) в Ψ_{0t} нет ни одного нереализованного интервала, 2) $p_t + p_{0t} \leqslant \tau$, где τ—некоторое наперед заданное число, определяемое, например, размером стандартной ПЛМ или имеющее другой смысл. Если выполнено условие 1, то

$$\Psi_R = \Psi_t, \quad p = |\Psi_R| = p_t.$$

Если выполнено условие 2, но не выполнено условие 1, то

$$\Psi_R = \Psi_t \cup \Psi'_{0t}, \quad p = p_t + p_{0t},$$

где Ψ'_{0t} получено из множества Ψ_{0t} удалением обобщенных интервалов, в выходных частях которых нет единиц, и присвоением нулевого значения всем неопределенным компонентам выходных частей.

Запишем действия, выполняемые на любом t-м шаге алгоритма $\Sigma 1$ ($\Sigma 2$, $\Sigma 3$, $\Sigma 4$).

1. В $\Psi_{0(t-1)}$ выбирается порождающий интервал. Им будет интервал (α_h, β_h), выходная часть которого имеет наименьший вес. Если в $\Psi_{0(t-1)}$ несколько интервалов с наименьшим весом выходной части, то в качестве порождающего берется любой из них. Если входными частями интервалов в $\Psi_{0(t-1)}$ являются n-интервалы, то для выбора порождающего можно использовать еще один критерий — минимум соседей у порождающего по сравнению со всеми интервалами с тем же весом выходной части.

2. Выполняются алгоритмы A1 и B1 в $\Sigma 1$ (A2 и B1 в $\Sigma 2$, A3 и B1 в $\Sigma 3$, A4 и B2 в $\Sigma 4$) для порождающего (α_h, γ_0) (γ_0 получен из вектора β_h путем присвоения нулевых значений всем его неопределенным компонентам). Полученный интервал (α_{hi}, γ_0) вносится в решение или при необходимости (или желании) расширяется любым из перечисленных ранее способов, и расширенный интервал включается в

решение Ψ_R. Из $\Psi_{0(t-1)}$ удаляются все интервалы, реализованные полученным в этом пункте интервалом.

3. Проверяются условия окончания работы алгоритма. Если хотя бы одно из них верно, то выполняется п. 4, в противном случае — *(t+1)*-й шаг алгоритма $\Sigma 1\,(\Sigma 2, \Sigma 3, \Sigma 4)$.

4. Конец.

Нетрудно видеть, что множество Ψ_R, полученное из Ψ_0 с помощью любого алгоритма $\Sigma 1 - \Sigma 4$, реализует систему *F(X)*. Действительно, алгоритмы $\Sigma 1 - \Sigma 4$ построены так, что Ψ_R реализует все единичные состояния функций системы *F(X)* и не интерпретирует как единичное ни одно нулевое состояние функций системы. Множество Ψ_R может оказаться избыточным в том смысле, что некоторый интервал в Ψ_R может оказаться реализованным группой других интервалов из Ψ_R. Как следует ожидать, предлагаемые алгоритмы дают лучшее решение (множество Ψ_R меньшей мощности) в случае, когда система *F (X)* задана множеством Ψ_0, обобщенных интервалов, входные части которых являются *n*-интервалами, чем в случае, когда входные части могут быть интервалами любого ранга. Заметим, что теоретико-структурный подход дает способ построения элемента решения для заданного порождающего и может использоваться в сочетании с любыми эвристическими приемами выбора порождающего интервала и доопределения неопределенных состояний функций системы.

В заключение проиллюстрируем работу алгоритмов Σ1 и Σ3 на примере одной системы булевых функций.

Пример 41. С помощью алгоритма Σ1 будем минимизировать систему, заданную следующим множеством обобщенных интервалов Ψ_0:

(α_i, β_i)	i
(0000, 0000)	1
(0001, 0101)	2
(0010, 0111)	3
(0011, 1000)	4
(0100, 1001)	5
(0101, 1101)	6
(0110, 1111)	7
(0111, 1001)	8
(1000, 011*)	9
(1001, *111)	10
(1010, 0110)	11
(1011, 1110)	12

1 Начнем с интервал имеющего номер 4, выходная часть которого имеет наименьший вес. Построим $D_{\text{ц}}(3)$. $d_0:=1$, $d_1:=$ $=0011 \oplus 0001$, $d_1=0010$, d_1 является входной частью обобщенного интервала 3, выходная часть которого $\beta_3 \lessgtr \beta_4$, поэтому элементу d_1 присваиваем значение «пусто» $(\varnothing)$, такое же значение присваиваем всем d_i с нечетными номерами, поскольку такие номера $i_{(2)} > 0001$.

Вычисляем элемент d_2. Получаем $d_2:=0011 \oplus 0010$, $d_2=0001$, $d_2=$ $=a_2$, $\beta_2 \gtrless \beta_4$, поэтому все d_i с номерами $i_{(2)} \geqslant 0010$ присваиваем значение $\varnothing$.

В этот момент стрка $D_{\text{ц}}(3)$ имеет вид

$$D_{\text{ц}}(3) = (1,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing)$$
$$0\quad 1\quad 2\quad 3\quad 4\quad 5\quad 6\quad 7\quad 8\quad 9\quad 10\quad 11\quad 12\quad 13\quad 14\quad 15$$

Заполним оставшиеся незаполненными места, соответствующие трем n-интервалам:

$d_4:=0011 \oplus 0100$, $d_4=0111$, $d_4:=\rho(a_8, \tilde{\beta}_8)$, $d_4=1$;

$d_8:=0011 \oplus 1000$, $d_8=1011$, $d_8:=\rho(a_{12}, \tilde{\beta}_{12})$, $d_8=1$;

$d_{12}:=0011 \oplus 1100$, $d_{12}=1111$, $d_{12}:=0$.

Получим

$$D_{\text{ц}}(3) = (1,\ \varnothing,\ \varnothing,\ \varnothing,\ 1,\ \varnothing,\ \varnothing,\ \varnothing,\ 1,\ \varnothing,\ \varnothing,\ \varnothing,\ 0,\ \varnothing,\ \varnothing,\ \varnothing).$$
$$0\quad 1\quad 2\quad 3\quad 4\quad 5\quad 6\quad 7\quad 8\quad 9\quad 10\quad 11\quad 12\quad 13\quad 14\quad 15$$

Из строки $D_{\text{ц}}(3)$ можно выделить единственный максимальный g-вектор и, следовательно, единственный максимальный интервал $**11$, цена которого равна трем. Полученный максимальный интервал вносим в решение: $\Psi_1=\{(**11, 1000)\}$.

Реализованные интервалы (0011, 1000), (0111, 1000), (1011, 1000) удаляем из Ψ_0, изменив элементы множества Ψ_0, стоящие под номерами 4, 8, 12, следующим образом:

(0011, $*000$) 4

(0111, $*001$) 8

(1011, $*110$) 12

2. Этот и последующие шаги запишем не так подробно, как п. 1. Возьмем интервал (0111, $*001$) и для порождающего (0111, 0001) построим строку

$$D_{\text{ц}}(7) = (1,\ 1,\ 1,\ 1,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ 0,\ 0,\ 0,\ 0,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing)$$
$$0\quad 1\quad 2\quad 3\quad 4\quad 5\quad 6\quad 7\quad 8\quad 9\quad 10\quad 11\quad 12\quad 13\quad 14\quad 15$$

Выделим максимальный интервал $*1**$, он единственный. За счет неопределенного компонента выходной части порождающего интервала (0111, $*001$) расширим выходную часть интервала, вносимого в решение, получим $(*1**, 1001)$. Таким образом,

$\Psi_2 = \{(**11,\ 1000),\ (*1**,\ 1001)\};$

$$\Psi_{02} = \begin{cases} (0000,\ 0000) & 1 \\ (0001,\ 0101) & 2 \\ (0010,\ 0110) & 3 \\ (0011,\ *000) & 4 \\ (0100,\ *00*) & 5 \\ (0101,\ *10*) & 6 \\ (0110,\ *11*) & 7 \\ (0111,\ *00*) & 8 \\ (1000,\ 011*) & 9 \\ (1001,\ *111) & 10 \\ (1010,\ 0110) & 11 \\ (1011,\ *110) & 12 \end{cases}$$

3. Берем $(0101,\ *10*)$. Для порождающего интервала (0101, 0100) построим строку

$$0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15$$

$$f_{\text{н}}(5) = (1,\ \varnothing,\ \varnothing,\ \varnothing,\ 1,\ \varnothing,\ \varnothing,\ \varnothing,\ 0,\ \varnothing,\ \varnothing,\ \varnothing,\ 1,\ \varnothing,\ \varnothing,\ \varnothing).$$

Выделяем интервал $(** 01,\ 0100)$, расширяем его выходную часть и вносим в решение

$$\Psi_3 = \{(**11,\ 1000),\ (*1**,\ 1001),\ (**01,\ 0101)\}.$$

4. Берем интервал $(1001,\ **1*)$, для порождающего (1001, (1010) построим строку

$$0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15$$

$$f_{\text{н}}(9) = (1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing).$$

В решение вносим обобщенный интервал $(1***,\ 0110)$. Получаем

$$\Psi_{04} = \begin{cases} (0000,\ 0000) & 1 \\ (0001,\ 0*0*) & 2 \\ (0010,\ 0110) & 3 \\ (0011,\ *000) & 4 \\ (0100,\ *00*) & 5 \\ (0101,\ **0*) & 6 \\ (0110,\ *11*) & 7 \\ (0111,\ *00*) & 8 \\ (1000,\ 0***) & 9 \\ (1001,\ ****) & 10 \\ (1010,\ 0**0) & 11 \\ (1011,\ ***0) & 12 \end{cases}$$

5. Остались нереализованными обобщенные интервалы 3 и 7. Для интервала (0010, ОНО) построим строку

$$\begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array}$$

$$D_{\text{ц}}(2) = (2,\ \varnothing,\ \varnothing,\ \varnothing,\ 2,\ \varnothing,\ \varnothing,\ \varnothing,\ 0,\ \varnothing,\ \varnothing,\ \varnothing,\ 0,\ \varnothing,\ \varnothing,\ \varnothing).$$

Интервал $(**10,\ 0110)$ вносим в решение.

За пять шагов получено множество максимальных обобщенных интервалов

$$\Psi_R = \begin{cases} (**11,\ 1000) & 1 \\ (*1**,\ 1001) & 2 \\ (**01,\ 0101) & 3 \\ (1***,\ 0110) & 4 \\ (**10,\ 0110) & 5 \end{cases}$$

реализующее исходную систему функций.

Пример 42. Решим ту же задачу, что в примере 41, с помощью алгоритма Σ3

1. Для порождающего (0011, 1000) построим матрицу $\Delta_{\text{ц}}$ и множество $G''_{\min}$ (в первой строке $\Delta_{\text{ц}}$ записываются g-векторы, во второй—цены соответствующих *n*-интервалов):

$g_1 = 0000 \oplus 0011 = 0011$, $\beta_1 < \beta_3$, g_1 вносим в $G^*_{\min}(3)$;

$g_2 = 0001 \oplus 0011 = 0010$, $\beta_3 \overset{<}{>} \beta_3$, $g_2 < g_1$, поэтому g_1 удаляем, g_2 вносим в $G''_{\min}(3)$;

$g_3 = 0010 \oplus 0011 = 0001$, $\beta_2 \overset{<}{>} \beta_1$, g_3 вносим в $G^*_{\min}(3)$;

$g_4 = 0000$, $\rho(a_4, \beta_4) = 1$, эти числа вносим в $\Delta_{\text{ц}}(3)$;

$g_5 = 0111$, g_5 удаляем, так как $g_5 > g_3$, а $g_3 \in G^*_{\min}(3)$;

$g_6 = 0110$, g_6 удаляем, так как $g_6 > g_2$, а $g_2 \in G^*_{\min}(3)$;

$g_7 = 0111$, g_7 удаляем, так как $g_7 > g_3$;

$g_8 = 0100$, $\beta_8 > \beta_4$, $\rho(a_8, \tilde{\beta}_8) = 1$, вносим g_8 и $\rho(a_8, \tilde{\beta}_8)$ в $\Delta_{\text{ц}}(3)$;

$g_9 = 1011$, $g_9 > g_3$, g_9 удаляем;

$g_{10} = 1010$, $g_{10} > g_2$, g_{10} удаляем;

$g_{11} = 1001$, g_{11} удаляем, так как $g_{11} > g_3$;

$g_{12} = 1000$, g_{12} и $\rho(a_{12}, \tilde{\beta}_{12}) = 1$ вносим в $\Delta_{\text{ц}}(3)$.

В результате получим

$$\Delta_{\text{ц}}(3) = \begin{bmatrix} 0000 & 0100 & 1000 \\ 1 & 1 & 1 \end{bmatrix};\ G''_{\min}(3) = \{0010,\ 0001\}.$$

Из $\Delta_{\text{ц}}$ (3) можно выделить два интервала, имеющие одинаковые цены. Проверим, являются ли векторы 0100, 1000 максимальными в G'(3). Матрица

$$Q = \begin{bmatrix} 0010 \\ 0001 \end{bmatrix}.$$

Данный случай тривиален. Единственному покрытию матрицы Q соответствует вектор 0011, отсюда следует, что максимальным является g=1100. Следовательно, в решение заносим интервал $(**11,\ 1000)$.

2. Запишем для порождающего интервала (0111, 0001)

$$\Delta_{\text{ц}}(7) = \begin{bmatrix} 0000 & 0001 & 0010 & 0011 \\ 1 & 1 & 1 & 1 \end{bmatrix};\ G'_{\min}(7) = \{0100\}.$$

из $\Delta_{\text{ц}}$ (7) выделим единственный интервал, соответствующий максимальному в $\Delta_{\text{ц}}$ (7) g-вектору 0011. Однако этот g-вектор не является максимальным в G'(7), единственный вектор в $G'_{\min}(7)$ свидетельствует о том, что максимальным является g=1011 Расширим выходную часть порождающего интервала за счет неопределенного компонента в выходной части интервала $(0111,\ *001)$, в решение вносим интервал $(*1*,\ 1001)$.

3 Для порождающего интервала (0101, 0100)

$$\Delta_{\text{ц}}(5) = \begin{bmatrix} 0000 & 0100 & 1100 \\ 1 & 1 & 1 \end{bmatrix};\ G'_{\min} = \{0001,\ 0010\}.$$

В решение вносим интервал $(**01,\ 0101)$.

4. Для порождающего интервала (1001, 0010)

$$\Delta_{\text{ц}}(9) = \begin{bmatrix} 0000 & 0001 & 0010 & 0011 \\ 1 & 1 & 1 & 1 \end{bmatrix};\ G'_{\min}(9) = \{1000\}.$$

В решение вносим интервал $(1***,\ 0110)$.

5. Для порождающего интервала (0010,0110)

$$\Delta_{\text{ц}}(2) = \begin{bmatrix} 0000 & 0100 \\ 2 & 2 \end{bmatrix};\ G'_{\min}(2) = \{0010,\ 0001\}.$$

В решение включаем интервал $(**10,\ 0110)$.

Получили множество обобщенных интервалов, реализующее исходную систему:

$$\Psi'_R = \begin{cases} (**11,\ 1000), \\ (*1**,\ 1001), \\ (1***,\ 0110), \\ (**10,\ 0110), \\ (**01,\ 0101). \end{cases}$$

В заключение приведем модифицированный алгоритм А4, а также дадим пример работы алгоритма Σ4, в котором выполняется модифицированный А4.

Модифицированный алгоритм А4

Переменные j, k, l и массивы $G'_{\max}(\alpha_h)$, $G''_{\min}(\alpha_h)$ имеют тот же смысл и те же начальные значения, что и в алгоритме А4.

1. $j:=j+1$. Если $j>p_0$, то выполняется п. 7.

2. Если $\beta_j \geqslant \beta_h$, то выполняется п. 4. $g:=\alpha_h \overset{\min}{\bigoplus} \alpha_j$. Если k=0, то выполняется п. 3. Если $g \geqslant g_i$, где $g_i \in \in G''_{\min}(\alpha_h)$, $i=1, \ldots, k$, то g отбрасывается и выполняется п. 1. Все $g_i \in G''_{\min}(\alpha_h)$ $(i=1, \ldots, k)$, такие, что $g_i > g$, удаляются из $G''_{\min}(\alpha_h)$ с соответствующим изменением k.

3. $k:=k+1$, $G''_{\min}(\alpha_h):=G''_{\min}(\alpha_h) \bigcup g$. Если $l=0$, то выполняется п. 1. Все $g_i \in G'_{\max}(\alpha_h)$ $(i=1, \ldots, l)$, такие, что $g_i \geqslant g$, удаляются из $G'_{\max}(\alpha_h)$ с соответствующим изменением l. Выполняется п. 1.

4. $g:=\alpha_h \overset{\max}{\bigoplus} \alpha_j$. Если $k=0$, то выполняется п. 5. Если $g \geqslant g_i$, где $g_i \in G''_{\min}(\alpha_h)$ $(i=1, \ldots, k)$, то g отбрасывается и выполняется п. 1.

5. Если $l=0$, то выполняется п. 6. Если где $g \leqslant g_i$, $g_i \in G'_{\max}(\alpha_h)$ $(i=1, \ldots, l)$, то g отбрасывается и выполняется п. 1. Все $g_i \in G'_{\max}(\alpha_h)$ $(i=1, \ldots, l)$, такие, что $g_i < g$, удаляются из $G'_{\max}(\alpha_h)$ с соответствующим изменением l.

6. $l:=l+1$, $G'_{\max}(\alpha_h):=G'_{\max}(\alpha_h) \bigcup g$. Выполняется п. 1.

7. Конец.

Пример 43. Применим алгоритм Σ4 (с модифицированным А4) к исходному множеству обобщенных интервалов

$$\Psi_0 = \begin{array}{|ll|l|} \hline \alpha_i & \beta_i & i \\ \hline (0000, & 10*) & 1 \\ (0010, & 110) & 2 \\ (0001, & 101) & 3 \\ (010*, & 001) & 4 \\ (0110, & 001) & 5 \\ (01*1, & 001) & 6 \\ (1010, & 1*0) & 7 \\ (1001, & 110) & 8 \\ (1*00, & 110) & 9 \\ (1101, & 1*1) & 10 \\ (11*0, & 1*0) & 11 \\ \hline \end{array}$$

1. В качестве порождающего интервала перем первый по порядку в Ψ_0, т. е. $(\alpha_h, \beta_h) = (0000, 100)$. Начинаем выполнять модифицированный алгоритм А4.

1. $j=1$.
1
2. $\beta_1 = \beta_h$.
4. $g = 0000$.
6. $l:=1$, $G'_{max}(\alpha_h) = \{0000\}$.
1. $j=2$.
1
2. $\beta_1 = \beta_h$.
4. $g = 0010$.
5, 6. Из $G'_{max}(\alpha_h)$ удаляется вектор 0000, $G'_{max}(\alpha_h) = \{0010\}$.
1. $j:=3$.
1
2. $\beta_3 > \beta_h$.
4 $g:=0001$.
6. $l:=2$, $G'_{max}(\alpha_h) = \{0010, 0001\}$.
1. $j:=4$.
1
2. $\beta_4 \not> \beta_h$, $g:=0000 \overset{min}{\oplus} 010* = 0100$.
3. $k:=1$. $G''_{min}(\alpha_h) = \{0100\}$.
1. $j:=5$.
1

2. $\beta_5 \not> \beta_h$, $g:=0110$, g отбрасывается, так как в множество $G''_{min}(\alpha_h)$ входит вектор 0100.

6. j:=6.
1

2. $\beta_6 \not> \beta_h$, $g:=0101$. g отбрасывается по той же причине, что и привыполнении п. 2 при j=5.

1. $j:=7$.
1
2. $\beta_7 = \beta_h$.
4. $g:=1010$.

5. Из $G'_{max}(\alpha_h)$ удаляется вектор 0010, так как 0010<1010, l:=1.

6. $l:=2$, $G'_{max}(\alpha_h) = \{0001, 1010\}$.
1. $j:=8$.
1
2. $\beta_8 \geqslant \beta_h$.
4. $g:=1001$.
5. Из $G'_{max}(\alpha_h)$ удаляется вектор 0001, $l=1$.
6. $l:=2$, $G'_{max}(\alpha_h) = \{1010, 1001\}$.
1. $j:=9$.
1
2. $\beta_9 \geqslant \beta_h$.

4. g: = **1100,** g отбрасывается, так как в множество $G''_{min}(\alpha_h)$ входит вектор 0100.

Выполнение модифицированного алгоритма А4 для j=10,11 не вносит изменения в сформированный ранее массив $G'_{max}(\alpha_k)$. Поэтому

7. Конец.

В решение нужно включить один из максимальных обобщенных интервалов (∗ 0∗ 0, 100) или (∗00∗, 100). Первый реализует обобщенный интервал $(\alpha_7, \beta_7) \in \Psi_0$ и частично реализует $(\alpha_2, \beta_2) \in \Psi_0$, второй частично реализует два интервала (α_3, β_3), $(\alpha_8, \beta_8) \in \Psi_0$. Поскольку цены этих максимальных интервалов одинаковы, то в решение включаем любой из них, например первый, т. е. $\Psi_R = \{(*0*0, 100)\}$. Удаляем из Ψ_0 реализованные интервалы (0000, 100), (0010,100), (1010,100). Изменяем элементы множества Ψ_0 имеющие номера 1,2,7, следующим образом:

(0000, ∗0∗) 1
(0001, ∗10) 2
(1010, ∗00) 7

II. Этот и последующие шаги преобразования исходного множества Ψ_0 запишем, не фиксируя выполнение каждого пункта алгоритма А4. В качестве порождающего интервала берем (0010, 010), полученный из $(\alpha_2, \beta_2) \in \Psi_0$. Выполнив для него модифицированный алгоритм А4, получим $G'_{max}(\alpha_k) = \{1000\}$. В решение вносим интервал (∗ 010, 010), в Ψ_0 изменяется выходная часть обобщенного интервала под номером два: $\beta_2 = **0$.

III. Порождающим становится интервал (010∗, 001), имеющий в Ψ_0 номер 4. Получаем $G'_{max}(\alpha_k) = \{0101, 0011\}$. Из максимальных обобщенных интервалов (0 ∗ 0 ∗, 001) и (01∗∗, 001) выбираем второй, так как он имеет большую цену, чем первый, и, включив его в решение, получим:

$$\Psi_R = \{(*0*0, 100), (0010, 010), (01**, 001\}.$$

На этом шаге

$$\Psi_{03} = \begin{cases} (0000, *0*) & 1 \\ (0010, **0) & 2 \\ (0001, 101) & 3 \\ (010*, 00*) & 4 \\ (0110, 00*) & 5 \\ (01*1, 00*) & 6 \\ (1010, *00) & 7 \\ (1001, 110) & 8 \\ (1*00, 110) & 9 \\ (1101, 1*1) & 10 \\ (11*0, 1*0) & 11 \end{cases}.$$

IV. Берем интервал $(11 * 0,1 * 0)$, имеющий номер 11 в Ψ_{03}. Для порождающего $(11*0,100)$ строим $G'_{max}(\alpha_h)=\{0111\}$. Единственный максимальный g-вектор соответствует максимальному обобщенному интервалу $(1 ** *, 100)$. Выходную часть этого интервала можно расширить (см. в Ψ_{03} строки 8... 11), и в решение внести интервал $(1 ***, 110)$.... Таким образом, получили

$$\Psi_{04}=\begin{cases}(0000, *0*) & 1\\ (0010, **0) & 2\\ (0001, 101) & 3\\ (010*, 00*) & 4\\ (0110, 00*) & 5\\ (01*1, 00*) & 6\\ (1010, *00) & 7\\ (1001, **0) & 8\\ (1*00, **0) & 9\\ (1101, **1) & 10\\ (11*0, **0) & 11\end{cases}$$

V. Единицы остались в выходных частях обобщенных интервалов, имеющих в Ψ_{04} номера 3 и 10. Из обобщенного интервала, имеющего номер 10 в Ψ_{04}, получаем порождающий (1101,001). Выполнив для него модифицированный алгоритм А4, получим $G'_{max}(\alpha_h)=\{1010\}$. В решение вносим интервал $(*1*1,001)$.

VI. Получаем последний элемент решения — $(000*,101)$. Таким образом решение задачи

$$\Psi_R=\begin{cases}(*0*0, 100) & 1\\ (*010, 010) & 2\\ (01**, 001) & 3\\ (1***, 110) & 4\\ (*1*1, 001) & 5\\ (000*, 101) & 6\end{cases}$$

6.4. Алгоритмы кодирования

6.4.1. Экономичное кодирование

Исходной информацией в задаче экономичного кодирования множества состояний S является совокупность К-множеств $\tilde{K}=\{K_0, \ldots, K_{m-1}\}$, $K_i \subset S$ Постановка этой задачи и определение К-множеств состояний даны ранее. Здесь напомним, что K_i — такие подмножества, каждому из которых мы

стремимся поставить в соответствие булев интервал (или всей совокупности $\tilde{K}$ минимальное число булевых интервалов).

Решая задачу экономичного кодирования, будем рассматривать три случая, различающиеся между собой характером исходной информации:

1) $K_i \cap K_j = \varnothing \quad (i, j = 0, \ldots, m-1, i \neq j)$, т. е. К-множества попарно не пересекаются;

2) совокупность $\tilde{K} = \{K_0, \ldots, K_{m-1}\}$ такова, что граф отношения пересечения К-множеств является связным;

3) граф отношения пересечения К -множеств состоит из нескольких (больше одного) компонентов связности.

(Под графом отношения пересечения К -множеств понимается неориентированный граф, каждой вершине которого поставлено в соответствие одно из К -множеств, принадлежащих совокупности $\tilde{K}$, две вершины графа связаны ребром, если соответствующие К -множества пересекаются.) Алгоритмы кодирования будем строить отдельно для первого и второго случаев, а алгоритм для третьего случая, очевидно, объединяет алгоритмы для первых двух.

Случай 1 (К -множества попарно не пересекаются). Если выполнено условие

$$\sum_{i=1}^{m} 2^{\lceil \log_2 | K_i | \rceil} \leqslant 2^n, \qquad (1)$$

где $n = \lceil \log_2 |S| \rceil$, то каждому К -множеству можно поставить в соответствие интервал *n*-мерного булева пространства и такие интервалы, как и сами К -множества, не будут иметь общих элементов. Если условие (1) не выполняется, то можно разбить некоторые множества K_i на попарно не пересекающиеся подмножества $K_{i1}, \ldots, K_{ik}$ и получить новое множество $\tilde{K}' = \{K_0, \ldots, K_{i1}, \ldots, K_{ik}, \ldots, K_{m-1}\}$; для которого (1) справедливо. Очевидно, это всегда можно сделать, так как $|S| \leqslant 2^n$.

Ранее задача кодирования была сведена к задаче построения булевой структуры $\tilde{\Theta}$, образованной разбиениями множества *S*, причем такими разбиениями, из блоков которых можно было бы составить минимальное покрытие совокупности К-множеств. В рассматриваемом случае задача построения (*n*+1)-уровневой структуры $\tilde{\Theta}$ распадается на *m* задач (по числу К-множеств). Решение каждой из них состоит в том, что свое множество K_i просто объявляется единицей (тривиальным одноблочным разбиением) булевой структуры $\tilde{\Theta}_i$ с числом уровней $n_i = \lceil \log_2 | K_i | \rceil + 1$.

Структуру $\hat{\Theta}$ будем строить в виде D-представления, т. е. строки $D = (d_0, d_1, \ldots, d_{2^n-1})$. Построение строки *D* сводится к записи состояний S в ее клетки (клеткой будем считать один элемент строки). Кодирующее слово каждого состояния будет равно двоичному номеру той клетки, в которую попало состояние.

Вспомним, как устроена строка *D*. Целиком строка является тривиальным одноблочным разбиением, т. е. *n*-м уровнем структуры $\hat{\Theta}$. Если строку *D* поделить пополам, то получится одно разбиение *(n*—1)-го уровня, если каждую половину снова поделить попалам, то получится одно разбиение *(n*—2)-го уровня структуры, и т. д. Другие разбиения уровней *(n*—1), *(n*—2), ... извлекаются из *D* не таким тривиальным способом, но тоже достаточно легко. Из сказанного вытекает простой (хотя не единственный) способ кодирования состояний в том случае, когда K-множества не пересекаются. В строку *D* записываются множества K_i в порядке убывания n_i , каждому K_i отводится 2^{n_i} клеток. Если $|K_i| < 2^{n_i}$, то $2^{n_i} - |K_i|$ клеток в части строки, отведенной для K_i, останутся пустыми. При такой записи *K*-множеств в *D* каждое из них сразу становится блоком разбиения n_i -уровня. Заметим, что если не соблюдать порядок записи K_i в *D* по убыванию n_i, то *K*-множество, записанное первым, станет блоком разбиения, а записанное вторым уже нет, если его мощность больше мощности первого. Действительно, пусть первым записано в *D* множество $K_1 = \{a, b\}$, $d_0 = a, d_1 = b$, а вторым $K_2 = \{c, d, e, f\}, d_2 = c, d_3 = d, d_4 = e, d_5 = f$.

Очевидно, что множество {010, 011, 100, 101} не является булевым интервалом и, следовательно, (d_2, d_3, d_4, d_5) не может оказаться блоком какого-либо разбиения . Чтобы и множество K_2 стало блоком разбиения, его нужно записать в клетках (d_2, d_3, d_6, d_7). Но такой способ кодирования был бы более сложным, чем запись в строке *D* подряд K-множеств.

Пусть условие (1) выполнено для совокупности K-множеств. Тогда алгоритм кодирования состояний S состоит из следующих простых процедур.

Алгоритм $\Xi 1$

1. K-множества располагаются в порядке убывания n_i .
2. Упорядоченные K-множества записываются в *D* друг за другом, при этом каждому K_i отводится 2^{n_i} клеток строки *D*.

Если $|K_i| < 2^{n_i}$, то последние $2^{n_i} - |K_i|$ клеток в подстроке, отведенной для K_i. остаются пустыми.

3. Каждому $s_k \in S$ присваивается код $j_{(2)}$, если $d_j = s_k$.
4. Конец.

Пример 44. Пусть для множества $S=\{a, b, c, d, e, f, g, h, i, j, k, l, m\}$ задана совокупность попарно не пересекающихся К-множеств: $K_0=\{a, b, c, d, e, f\}$, $K_1=\{g, h, i\}$, $K_2=\{j, k, l\}$, $K_3=\{m\}$.
Проверим условие (1). Поскольку $|S| = 13$, постольку $n = 4$.
Вычислим для каждого K_i величину n_i, получим $n_0=3$, $n_1=2$, $n_2=2$, $n_3=1$. Тогда $2^3+2^2+2^2+2^0>2^4$. Условие (1) не выполнено.
Разобьем K_0 на два подмножества: $K_{01}=\{a, b, c, d\}$ и $K_{02} = \{e, f\}$.
Тогда $n_{01}=2$, $n_{02}=1$ и $2^2+2^1+2^2+2^2+2^0<2^4$.
Расположим К-множества в порядке убывания n_i, получим K_{01}, K_1, K_2, K_{02}, K_3. Запишем К-множества в строку D:

$$D=(\underbrace{a, b, c, d}_{K_{01}}, \underbrace{g, h, i, \varnothing}_{K_1}, \underbrace{j, k, l, \varnothing}_{K_2}, \underbrace{e, f}_{K_{02}}, \underbrace{m}_{K_3}, \varnothing).$$

Получим следующую кодирующую таблицу.

0 0 0 0	a
0 0 0 1	b
0 0 1 0	c
0 0 1 1	d
1 1 0 0	e
1 1 0 1	f
0 1 0 0	g
0 1 0 1	h
0 1 1 0	i
1 0 0 0	j
1 0 0 1	k
1 0 1 0	l
1 1 1 0	m

К-множества реализуются следующими булевыми интервалами: K_0 — 00**, И110*, K_1 — 01**, K_2 — 10**, K_3 — 1110 ИЛИ 111*.
Алгоритм кодирования в случае попарно не пересекающихся К-множеств тривиален. Второй случай существенно более сложен.
Случай 2 (К-множества пересекаются, граф отношения пересечения является связным). Проведем синтез $\hat{\Theta}$ как построение разбиений первого уровня на основе списка P_1 взвешенных пар состояний. (Напомним, пара состояний (s_i, s_j) входит в список P_1 если эта пара включена хотя бы в одно К-множество. Вес пары равен числу К-множеств, включающих ее).
Правила построения разбиений первого уровня сформулированы ранее. Поскольку структуру $\hat{\Theta}$ строим в виде D-представления, перенесем эти правила на построение строки D.

Для разбиения θ^1_i $(i=0, ..., n-1)$ строка D представляется состоящей из 2^{n-i-1} подстрок D^i_j по 2^{i+1} элемента в каждой. Так, разбиение θ^1_0 обусловливает расчленение строки D на 2^{n-1} подстроки, каждая подстрока D^0_j $(j=0, ..., 2^{n-1}-1)$ содержит два элемента—пару состояний, которая является блоком разбиения θ^1_0. Для разбиения θ^1_1 строка D рассматривается как состоящая из 2^{n-2} подстрок D^1_j $(j=0, ..., 2^{n-2}-1)$ по четыре элемента в каждой. Элементы 1 и 3, 2 и 4 каждой подстроки составляют пары разбиения θ^1_1 и т. д. Имея в виду такое устройство D, формировать D будем подстроками. Разбиения первого уровня будем строить по порядку возрастания их номеров. Записав в D разбиение θ^1_0, т. е. 2^{n-1} пар из списка P_1, мы уже заполнили всю строку D. Построение разбиения θ^1_1 сводится к переупорядочению состояний в строке D. Чтобы не разрушить в строке D разбиение θ^0_1, нужно ограничить переупорядочение следующим образом: разрешается менять местами подстроки D^0_j $(j=0, ..., 2^{n-1}-1)$ (т. е. их номера) и переставлять состояния внутри любой подстроки D^0_j. Когда сформированы все D^1_j $(j=0, ..., 2^{n-2}-1)$ (т. е. построено разбиение θ^1_1, то при формировании подстрок D^2_j $(j=0, ..., 2^{n-3}-1)$ (т. е. при построении разбиения θ^1_2) можно менять местами подстроки D^1_j, внутри них — D^0_j и, наконец, внутри D^n_j —состояния. И вообще, построить θ^1_i— это значит сформировать множество подстрок $\{D^i_0, ..., D^i_{2^{n-i-1}-1}\}$, не нарушив при этом ранее построенных более мелких подстрок

$$\{D^0_0, ..., D^0_{2^{n-1}-1}, D^1_0, ..., D^1_{2^{n-2}-1}, ..., D^{i-1}_0, ..., D^{i-1}_{2^{n-i}-1}\}$$

в том смысле, что элементы этих подстрок можно переставлять лишь в той мере, в какой эти подстроки не перестают соответствовать своим разбиениям. Такие перестановки равносильны тому, что в разбиении θ^1_k $(k = 0, ..., n-1)$ переставили блоки или в одном блоке поменяли местами состояния. Правила перестановки элементов строки D определены формулами (20) — (22).

Будем считать, что алгоритмы построения D-представления структуры $\hat{\Theta}$ в с учетом списка взвешенных пар P_1 состоит из n тагов $\{0, 1, ..., n-1\}$ (по числу разбиений первого уровня структуры $\hat{\Theta}$). Результатом выполнения $(i-1)$-го шага является строка $D^{i-1} = \{D^{i-1}_0, ..., D^{i-1}_{2^{n-i}-1}\}$. Строка D^{n-1}, полученная на последнем шаге, является D-представлением структуры $\hat{\Theta}$.

Построение строки D^i производится путем переупорядочения состояний в D^{n-1}. Для этого необходимо 2^{n-i-1} итераций. За одну

итерацию формируется подстрока D^i_j. Алгоритм, выполняющий формирование подстроки, назовем Е1. Особенностью его является то, что он применяется к уже «заполненной» строке, т. е. по меньшей мере к D^0. Точнее говоря, Е1 есть алгоритм переупорядочения двух подстрок из D^{i-1} с целью формирования из них одной подстроки в D^i. Подстрока D^i_j должна быть сформирована на месте подстрок D^{i-1}_{2j} и D^{i-1}_{2j+1}. Формирование D^i_j начинается с того, что из списка P_1 берется пара состояний с наибольшим весом, пусть это *(a, b)*. Ранее было установлено, что подстрока D^i_j соответствует коллекции C^i_j, порождаемой в θ^1_i разбиениями $\{\theta^1_0, \ldots, \theta^1_{i-1}\}$, ранее описано построение коллекции C^i_j, которое начинается с некоторой начальной пары состояний, взятой из списка P_1. Состояния *a* и *b* могут оказаться в любых двух подстроках D^{i-1}_k и D^{i-1}_l: одно состояние в D^{i-1}_k, другое — в D^{i-1}_l. (Согласно следствию теоремы 1 в качестве начальной пары для формируемой строки может быть взята только такая пара, состояния которой принадлежат разным подстрокам в D^{i-1}.) Номера клеток строки *D* (вернее, подстрок D^{i-1}_{2j} и D^{i-1}_{2j+1}), в которые должны быть помещены состояния *a* и *b*, определяются по формуле (16). Состояния *a* и *b* ставятся в эти клетки, а в подстроках D^{i-1}_k, D^{i-1}_l и D^{i-1}_{2j}, D^{i-1}_{2j+1} производятся перестановки состояний по формуле (22). При этом, если состояние *a* взято из строки D^{i-1}_k и поставлено в D^{i-1}_{2j}, а состояние *b* из строки D^{i-1}_l переставлено в D^{i-1}_{2j+1} (по формуле (16) состояние *a* должно занять первую клетку в D^{i-1}_{2j}, а *b* — первую клетку в D^{i-1}_{2j+1}), то это влечет за собой перемену местами подстрок D^{i-1}_k и D^{i-1}_{2j}, D^{i-1}_l и D^{i-1}_{2j+1}. Кроме того, состояния в этих подстроках тоже переупорядочиваются в соответствии с формулой (22). Таким образом, при формировании D^i_j могут быть затронуты четыре подстроки из D^{i-1}. Если же состояние *a* окажется в строке D^{i-1}_{2j}, а состояние *b* — в D^{i-1}_{2j+1}, то для формирования D^i_j нужно произвести перестановки только в двух подстроках: D^{i-1}_{2j} и $D^{i-1}_{2,+1}$.

Алгоритм Е1

1. Если список P_1 пуст, то подстрока D^i_j считается построенной и выполняется п. 5. В противном случае из списка P_1 выбирается пара с наибольшим весом и такая, что состояния, ее составляющие, входят в разные подстроки строки D^{i-1}, пусть это пара *(a, b)*. Если в списке P_1 подходящей пары не оказалось, то подстрока D^i_j считается построенной и совершается переход к п. 5.

2. Просмотром строки D^{i-1} устанавливаются номера тех клеток, в которых стоят состояния a и b, пусть $d_t = a$, $d_u = b$. При этом пусть d_t находится в подстроке D_k^{i-1}, а d_u — в D_l^{i-1}. Номера v и w клеток, которые надлежит занять соответственно состояниям a и b согласно (16), определяются по следующим формулам (в (16) положили $k = 0$):

$$v = j \cdot 2^{i-l}, \quad w = j \cdot 2^{i+1} + 2^i. \quad (2)$$

Пусть $d_v = q$, $d_w = r$ (d_v — это первая клетка в подстроке D_{2j}^{i-1}; d_w - первая клетка в D_{2j+1}^{i-1}).

3. Состояния a и q меняются местами — $d_v := a$, $d_t := q$. Элементы d_{v+m} $(m = 1, \ldots, 2^i - 1)$ и $d_{t'}$ меняются местами, где согласно (22)

$$t'_{(2)} = t_{(2)} \oplus v_{(2)} \oplus (v + m)_{(2)}. \quad (3)$$

4. Состояния b и r меняются местами — $d_w := b$, $d_u := r$. Элементы d_{w+m} $(m = 1, \ldots, 2^i - 1)$ и $d_{u'}$ меняются местами, где согласно (22)

$$u'_{(2)} = u_{(2)} \oplus w_{(2)} \oplus (w + m)_{(2)}. \quad (4)$$

5. Конец.

Пример 1. Пусть имеется строка

$$\begin{array}{cccccccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ D^1 = (& \underbrace{b,} & c, & a, & e, & \underbrace{d,} & h, & i, & k, & \underbrace{l,} & f, & p, & o, & \underbrace{g,} & j, & m, & n). \\ & D_0^1 & & & & D_1^1 & & & & D_2^1 & & & & D_3^1 & & & \end{array}$$

Применим к D^1 алгоритм E1 и сформируем подстроки D_0^2 и D_1^2, т. е. строку D^2.

Пусть начальной для $D^2{}_0$ будет пара (h, p). Подстрока $D^2{}_0$ должна быть сформирована на месте подстрок D_0^1 и D_1^1. Согласно (2) состояния начальной пары (h, p) должны занять в строке D^2 клетки 0 и 4. h принадлежит D_1^1, $p \leftarrow D_2^1$. Поставим h в клетку 4 и p в клетку 0. Это повлечет за собой перестановку элементов в D_1^1, D_0^1 и D_2^1. При этом состояния, принадлежащие подстроке $D^1{}_1$, будут переставлены в ее пределах, а состояния из D_1^0 и D_2^1 поменяются местами

Опишем перестановки в $D^1{}_1$. Поскольку h ставится в клетку 4, постольку d переходит из клетки 4 в клетку 5. Новые номера клеток для i и k соответственно:

$$0101 \oplus 0100 \oplus 0110 = 0111,$$
$$0101 \oplus 0100 \oplus 0111 = 0110.$$

Это значит, что i и k просто меняются местами.

Опишем перестановки в D_0^1 и D_2^1. Поскольку p встало в клету 0, постольку b заняло клетку 10. Дальнейшие перестановка таковы:

c — клетка 11 $(0000 \oplus 1010 \oplus 0001 = 1011)$, o — клетка 1;
a — клетка 8 $(0000 \oplus 1010 \oplus 0010 = 1000)$, l — клетка 2;
e — клетка 9 $(0000 \oplus 1010 \oplus 0011 = 1001)$, f — клетка 3.

В результате всех перестановок получим

$$\underbrace{(p,\ o,\ l,\ f\ h,\ d,\ k,\ i,}_{D_0^2}\ a,\ e,\ b,\ c,\ g,\ j,\ m,\ n).$$

Пусть для D_1^2 начальной будет пара (c, j). Произведя необходимые перестановки, получим:

$$D^2 = \underbrace{(p,\ o,\ l,\ f,\ h,\ d,\ k,\ i,}_{D_0^2}\ \underbrace{c,\ b,\ e,\ a,\ j,\ g,\ n,\ m)}_{D_1^2}.$$

Перед тем как представить целиком алгоритм кодирования множества состояний S для случая, когда K-множества пересекаются по связному графу, сделаем следующее замечание. Если мощность множества S меньше чем 2^n $(n = \lceil \log_2 |S| \rceil)$, то $2^n - |S|$ n-разрядных кодирующих слов останутся незанятыми. Наличие свободных кодов можно использовать для того, чтобы успешнее решить задачу экономичного кодирования. С этой целью введем $2^n - |S|$ фиктивных состояний и те $K_i \in \tilde{K}$, мощность которых не равна 2^{n_i} $(n_i = \lceil \log_2 |K_i| \rceil)$ и которые по этой причине не могут быть реализованы одним булевым интервалом каждое, дополним фиктивными состояниями до $|K_i| = 2^{n_i}$ так, чтобы K-множества не пересекались по подмножествам фиктивных элементов, вошедших в них. Очевидно, не всегда фиктивных состояний будет достаточно, чтобы дополнить все требующие того K-множества. Чтобы дополнить большее число K-множеств, начнем с тех K_i, мощность которых меньше всего отличается от 2^{n_i}.

Запишем алгоритм экономичного кодирования множества S для случая, когда граф отношения пересечения K-множеств является связным и $n = \lceil \log_2 |S| \rceil$. Назовем этот алгоритм $\Xi 2$.

Алгоритм $\Xi 2$

1. Если $|S| < 2^n$, то в множество S вводится $2^n - |S|$ фиктивных элементов, K-множества, начиная с тех K_i, мощность которых меньше всего отличается от 2^{n_i}, дополняется до $|K_i| = 2^{n_i}$ фиктивными элементами таким образом, чтобы K-множества не пересекались по подмножествам фиктивных элементов, входящих в них.

2. На основе К-множеств составляется список взвешенных пар состояний P_1.
3. Из списка P_1 выбирается 2^{n-1} пар с наибольшим весом, попарно не пересекающихся. Выбранные пары заносятся в D^0. Если в списке P_1 не оказалось 2^{n-1} подходящих пар, то оставшиеся места в D^0 заполняются влементами $s_k \in S$, не вошедшими в D^0.
4. Каждая строка D^i $(i=1, ..., n-1)$ строится путем применения к D^{i-1} алгоритма Е1 2^{n-i-1} раз.
5. Каждому $s_k \in S$ присваивается кодирующее слово $j_{(2)}$, если $d_j = s_k$.
6. Конец.

Пример 2. Требуется найти экономичное кодирование элементов множества $S=\{a, b, c, d, e, f, g, h\}$ при n=3, если матрица $\mathbf{M}_2$ имеет вид

$$\mathbf{M}_2 = \begin{array}{c} \begin{matrix} f_4 & f_3 & f_2 & f_1 & f_0 \end{matrix} \\ \left[\begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}\right] \end{array} \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{matrix}$$

1. В соответствии с расстановкой единиц в $\mathbf{M}_2$ определяются К-множества и список взвешенных пар:

$$K_0 = \{c, d, g, h\},\ K_1 = \{b, e\},\ K_2 = \{c, d\},$$
$$K_3 = \{b, d, e, g\},\ K_4 = \{a, b, e, f\};$$
$$P_1 = \{(a, b),\ (e, f),\ (a, e),\ (a, f),\ (b, e)\,(3),\ (b, f),$$
$$(b, d),\ (e, g),\ (b, g),\ (d, e),\ (d, g),\,(2),\ (c, d)\,(2),\ (g, h),$$
$$(c, g),\ (c, h),\ (d, h)\}.$$

В скобках после трех пар в списке P_1 указан их вес, остальные пары имеют вес, равный единице.
2 Строится строка $D^0 = (D_0^0, D_1^0, D_2^0, D_3^0)$. Каждая ее подстрока D_i^0 $(i=0,1,2,3)$ является одной из пар списка P_1:

$$\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix}$$
$$D^0 = \underbrace{(b,\ e,}_{D_0^0}\ \underbrace{d,\ g,}_{D_1^0}\ \underbrace{a,\ f,}_{D_2^0}\ \underbrace{c,\ h).}_{D_3^0}$$

3. Строится строка $D^1 = (D_0^1, D_1^1)$.

а) Для построения подстроки $D^1{}_0$ из списка P_1 выбирается пара (c, d). В подстроке D_0^1 должно быть $d_0^1 = c$, $d_2^1 = d$. В строке D^0 есть $d_6^0 = c$, $d_2^0 = d$ (Верхние индексы в обозначении элементов строки D говорят о принадлежности этих элементов строкам D^0, D^1.) Следовательно, переупорядочению подвергаются подстроки D_0^0, D_3^0: $d_0^1 := c$, $d_6^1 := b$, затем по (3)

$$110 \oplus 000 \oplus 001 = 111;$$

$$d_7^1 := e, \quad d_1^1 := h.$$

После этих перестановок получаем

$$\underbrace{(c,\ h,\ d,\ g}_{D_0^1},\ a,\ f,\ b,\ e).$$

0 1 2 3 4 5 6 7

б) Для подстроки D_1^1 из списка P_1 берется пара (a, b), и оказывается, что ее элементы стоят в подобающих им клетках, поэтому переупорядочение не требуется, и

$$D^1 = (\underbrace{c,\ h,\ d,\ g}_{D_0^1},\ \underbrace{a,\ f,\ b,\ e}_{D_1^1}).$$

4. Строится $D^2 = (D_0^2) = D$. Из списка P_1 выбирается пара (b, d). Согласно (2) в строке D^2 должно быть $d_0^2 = b$, $d_4^2 = d$ в D^1 есть $d_6^1 = b$, $d_2^1 = d$. Переставляем $d_1^2 := b$, $d_6^2 := c$, далее по (3)

$$110 \oplus 000 \oplus 001 = 111, \quad d_7^2 := h, \quad d_1^2 := e;$$

$$110 \oplus 000 \oplus 010 = 100, \quad d_4^2 := d, \quad d_2^2 := a;$$

$$110 \oplus 000 \oplus 011 = 101, \quad d_5^2 := g, \quad d_3^2 := f.$$

Дальнейшие перестановки не требуются, так как d оказалось в своей клетке — $d^2{}_4$. Таким образом, получили

0 1 2 3 4 5 6 7

$$D = (b,\ e,\ a,\ f,\ d,\ g,\ c,\ h).$$

5. Кодирующая таблица имеет вид

$$C_S = \begin{bmatrix} 010 \\ 000 \\ 110 \\ 100 \\ 001 \\ 011 \\ 101 \\ 111 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{matrix}$$

K-множества реализуются следующими булевыми интервалами:

$K_0 - 1{*}{*}$, $K_1 - 00{*}$, $K_2 - 1{*}0$, $K_3 - {*}0{*}$, $K_4 - 0{*}{*}$.

Соответствующее матричное представление системы функций имеет вид

x_2	$\bar{x}_2$	x_1	$\bar{x}_1$	x_0	$\bar{x}_0$	f_4	f_3	f_2	f_1	f_0
0	1	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0	1	0	0
0	1	0	1	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	1

Заметим, что в данном примере случайно каждое К-множество оказалось покрытым одним интервалом. В общем случае это получается не всегда.

Случай 3 (К-множества пересекаются, граф отношения имеет несколько компонентов связности). Алгоритмы Э1 и Э2 могут быть использоваться совместно, когда пересечение К-множеств имеет место внутри их подмножеств. Это значит, что на множестве $\tilde{K} = \{K_0, \ldots, K_{m-1}\}$ существует разбиение на подмножества $U = \{\tilde{K}_1, \ldots, \tilde{K}_q\}$, каждое $\tilde{K}_i$ содержит несколько К-множеств, которые пересекаются по связному графу, а пересечение К-множеств из разных $\tilde{K}_i$ пусто. В соответствии с разбиением U на множестве состояний S также существует разбиение на подмножества $R = \{S_1, \ldots, S_q\}$. Каждое подмножество состояний S_i равно объединению К-множеств, вешедших в $\tilde{K}_i$: $S_i = \bigcup_{K_j \in \tilde{K}_i} K_j$.

Таким образом, задачу кодирования множества состояний S можно свести к q независимым задачам кодирования подмножеств $S_i \subset S$ $(i = 1, \ldots, q)$. Как в случае кодирования состояний для непересекающихся К-множеств проверялось выполнение условия (1), так и в данном случае проверяется, выполнено ли аналогичное условие

$$\sum_{i=1}^{q} 2^{\lceil \log_2 |S_i| \rceil} \leqslant 2^n, \quad n = \lceil \log_2 |S| \rceil. \quad \textbf{(5)}$$

Если это условие не выполняется, то некоторые S_t нужно объединить и получить новое разбиение множества состояний $R' = \{S'_1, \ldots, S'_r\}$ $(S'_i \subset S,\ i = 1, \ldots, r,\ r < q)$, для которого бы выполнилось условие (5).

Теперь кодирование множества состояний S проводится в два этапа. На первом выполняется алгоритм Э1, на втором Э2. Это значит, что в

строку D с числом элементов 2^n записываются подмножества S_i в порядке убывания $n_i =]\log | S_i | [$. Затем отдельно для каждой подстроки D_i (отведенной для S_i) с числом элементов 2^{ni} выполняется алгоритм Ξ2. Покажем такое кодирование на примере.

Пример 3. Пусть имеется следующая совокупность K-множеств:

$K_0=\{a, b\}$; $K_1=\{b, e\}$; $K_2=\{c, f, e, g\}$; $K_3=\{d, f, h\}$;

$K_4=\{i, j\}$; $K_5=\{i, k\}$; $K_6=\{j, m\}$; $K_7=\{l, n, o, p\}$.

Совокупность $\tilde{K}$ разбивается на три подмножества, внутри которых K-множества пересекаются. Разбиение имеет вид

$U=\{\tilde{K}_1, \tilde{K}_2, \tilde{K}_3\}$, $\tilde{K}_1=\{K_0, K_1, K_2, K_3\}$;

$\tilde{K}_2=\{K_4, K_5, K_6\}$; $\tilde{K}_3=\{K_7\}$.

Соответствующее разбиение множества $S=\{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p\}$.

$R=\{S_1, S_2, S_3\}$, $S_1=\{a, b, c, d, e, f, g, h\}$;

$S_2=\{i, j, k, m\}$; $S_3=\{l, n, o, p\}$.

Для множеств S_1, S_2, S_3 условие (5) выполнено. Действительно, $n=4$, $n_1=3$, $n_2=2$, $n_3=2$ и $2^3+2^2+2^2=2^4$. Записав подмножества S_1, S_2, S_3 в строку D, получим:

$$D=(\underbrace{a, b, c, d, e, f, g, h}_{D_1}, \underbrace{i, j, k, m}_{D_2}, \underbrace{l, n, o, p}_{D_3}).$$

Применяя отдельно к подстрокам D_1, D_2, D_3 алгоритм Ξ2, получаем строку

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$D=$	a,	b,	d,	h,	c,	e,	f,	g,	i,	j,	k,	m,	l,	n,	o,	p)

или кодирующую таблицу

0	0	0	0	a
0	0	0	1	b
0	1	0	0	c
0	0	1	0	d
0	1	0	1	e
0	1	1	0	f
0	1	1	1	g
0	0	1	1	h
1	0	0	0	i
1	0	0	1	j
1	0	1	0	k
1	1	0	0	l
1	0	1	1	m
1	1	0	1	n
1	1	1	0	o
1	1	1	1	p

К-множества реализуются следующими булевыми интервалами:
$K_0 - 000*,\ K_1 - 0*01,\ K_2 - 01**,\ K_3 - 0*10,\ 0011,$
$K_4 - 100*,\ K_5 - 10*0,\ K_6 - 10*1,\ K_7 - 11**.$

6.4.2. Противогоночное кодирование

Два условия отсутствия опасных состязаний в асинхронном автомате, выраженные в терминах теоретико-структурного подхода, приведены ранее . Напомним их. Первое является необходимым и достаточным, оно формулируется для пар состязающихся переходов и состоит в следующем. В автомате не будет опасных состязаний между элементами памяти, если, и только если для любой пары состязающихся переходов ($s_l \to s_k,\ s_a \to s_r$) среди двублочных разбиений $\{\theta_0^{n-1}, \ldots, \theta_{n-1}^{n-1}\}$, соответствующих *n*-разрядному двоичному коду, найдется хотя бы одно такое, в котором состязающиеся переходы находятся в разных блоках:

$$\{\overline{s_l,\ s_k},\ \overline{s_a,\ s_r}\} \ll \theta_j^{n-1}. \quad (6)$$

Второе условие является достаточным, оно накладывается на пары состязающихся *k*-множеств. В автомате не будет опасных состязаний, если для любой пары состязающихся *k*-множеств (k_l, k_r) среди двублочных разбиений $\{\theta_0^{n-1}, \ldots, \theta_{n-1}^{n-1}\}$ найдется хотя бы одно такое, в котором состязающиеся *k*-множества находятся в разных блоках:

$$\{\overline{k_l},\ \overline{k_r}\} \ll \theta_j^{n-1}. \quad (7)$$

Противогоночное кодирование, как это следует из (6) и (7), сводится к построению *(n*—1)-го уровня булевой структуры $\hat{\Theta}$, т. е. множества двублочных разбиений совокупности состояний S. Такая постановка задачи противогоночного кодирования близка к известным. Заметим, что в отличие от известных нам не приходится заботиться о том, чтобы все элементы множества S получили различные коды. Наличие для каждого $s_k \in S$ своего кодирующего слова обеспечивается условием существования нулевого разбиения в структуре $\hat{\Theta}$.

Правила построения двублочных разбиений $\{\theta_0^{n-1}, \ldots, \theta_{n-1}^{n-1}\}$, сформулированные ранее, одни и те же и для списка пар состязающихся переходов, и для списка пар состязающихся *k*-множеств. Поэтому не будем различать эти списки, а будем считать, что есть один список $P_2 = \{\pi_i\}$ пар подмножеств множества S ($\pi_i = (S_{i1},\ S_{i2})$), который в зависимости от того, какое из условий (6) или (7) используется, будет списком пар либо состязающихся переходов, либо состязающихся *k*-множеств.

Алгоритм противогоночного кода C_s с минимальной или близкой к минимальной длине *n* кодирующего слова состоит в *n*-кратном выполнении двух основных процедур.

1. Для построения каждого разбиения θ_i^{n-1} из списка P_2 выбирается множество согласованных пар. Выбор производится с учетом требования минимальной (или близкой к минимальной) длины кодирующего слова.

2.Построение при θ_i^{n-1} условии, что разбиения $\theta_0^{n-1}, \ldots, \theta_{i-1}^{n-1}$ уже построены. Разбиения θ_i^{n-1} будем формировать путем переупорядочения состояний S в строке *D.*

Задача построения минимального по мощности множества двублочных разбиений множества S , развязывающих все пары списка P_2, формулируется как задача поиска кратчайшего покрытия списка P_2 двублочными разбиениями. Мы не будем ставить целью найти кратчайшее покрытие. Воспользуемся естественными для таких задач эвристическими приемами, позволяющими построить покрытие, близкое к кратчайшему, и состоящими в следующем.

Для построения каждого θ_i^{n-1} $(i=0, \ldots, n-1)$ будем использовать наиболее мощное подмножество пар $\{\pi_i\} \subset P_2$, согласованных с некоторой парой $\pi_j \in P_2$ и не развязанных в уже построенных разбиениях $\theta_0^{n-1}, \ldots, \theta_{i-1}^{n-1}$. Пару π_j будем считать базовой для разбиения θ_i^{n-1} и обозначать $\tilde{\pi}$. В качестве базовой $\tilde{\pi}$ из неразвязанных пар списка P_2 будем выбирать пару, для которой число ω подмножеств согласованных с нею пар наименьшее. Совокупность подмножеств пар, согласованных с базовой $\tilde{\pi}$, будем обозначать через $\{\tilde{P}_i\}$, а наиболее мощное подмножество — через $\tilde{P}_{max}$. Первый эвристический прием — использование наиболее мощного подмножества согласованных пар — позволяет за меньшее число шагов (и, значит, с меньшим *n)* исчерпать список P_2. Второй прием — использование в качестве базовой пары с наименьшим ω— позволяет делать выбор среди наименьшего количества претендентов и тем самым ближе подходить к точному решению задачи.

Пример 4. Пусть имеется список P_2:

$$\pi_1=(a, b; c, d);\ \pi_2=(a, e; c, f);\ \pi_3=(e, f; c, d);$$
$$\pi_4=(a, f; c, g);\ \pi_5=(b, g; c, e).$$

Совокупность $\{\tilde{P}_i\}$ для базовой $\tilde{\pi}=\pi_1$ содержит три множества:

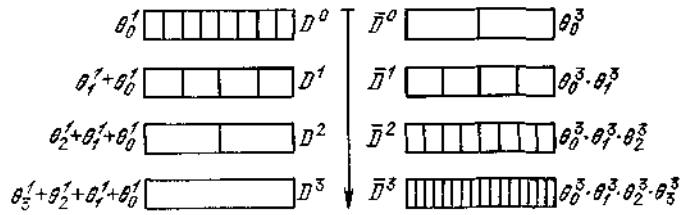
$$\tilde{P}_1=\{\pi_1, \pi_2\};\ \tilde{P}_2=\{\pi_1, \pi_3, \pi_4\};\ P_3=\{\pi_1, \pi_5\},$$

При этом $\tilde{P}_{max}=\tilde{P}_2$. Вычислим для всех пар списка P_2 числа

$$\omega(\pi_1)=3,\ \omega(\pi_2)=\omega(\pi_3)=\omega(\pi_4)=\omega(\pi_5)=1$$

Вторая основная процедура алгоритма противогоночного кодирования требует перенесения правил построения разбиений *(n*—1)-го уровня структуры $\hat{\Theta}$ на построения D-представления структуры. Подобно тому как мы считали, что множество разбиений $\{\theta_0^{n-1}, \ldots, \theta_{n-1}^{n-1}\}$ порождает в разбиении θ_i^{n-1} 2^{i+1} секцию $E_0^i, \ldots, E_{2^{i+1}-1}^i$, будем считать, что строка $\bar{D}^i$ состоит из 2^{i+1} подстроки $\bar{D}_0^i, \ldots, \bar{D}_{2^{i+1}-1}^i$ (в отличие от строки $D^i=(D_0^i, \ldots, D_{2^{n-i-1}-1}^i)$, которая является результатом построения *i*-го разбиения первого уровня, строку, являющуюся результатом формирования *i*-го разбиения (*n*—1)-го уровня структуры, обозначим $\bar{D}^i=(\bar{D}_0^i, \ldots, \bar{D}_{2^{i+1}-1}^i)$.)

Заметим, что если для формирования разбиения первого уровня мы строили строку *D,* начиная с мелких подстрок, укрупняя их с каждым θ_j^1, при этом фиксировали относительное расположение состояний S в подстроках D_j^i, не закрепляя в *D* порядок самих этих подстрок, то для формирования разбиений *(n*—1)-го уровня мы должны строить *D,* начиная с крупных подстрок, размельчая их с каждым θ_i^{n-1} при этом фиксировать порядок расположения подстрок $\bar{D}_l^i$ в *D,* оставляя свободу в расположении состояний S внутри подстрок. Двойственность процедур построения строк $D^0, \ldots, D^{n-1}$ и $\bar{D}^0, \ldots, \bar{D}^{n-1}$ изобразим следующей диаграммой для $n = 4$:



Как видно из диаграммы, кроме двойственного характера самих действий, которые выполняются для синтеза структуры $\hat{\Theta}$ при экономичном и противогоночном кодировании: в первом случае это объединение состояний во все более крупные блоки, во втором — расчленение множества состояний на все более мелкие блоки, двойственны и уровни в $\hat{\Theta}$, разбиения которых оказываются соответствующими строкам D^i и $\bar{D}^i$. Действительно, все начинается в первом случае с формирования разбиения первого уровня θ_0^1 в виде

D^0, а во втором — с построения разбиения третьего уровня θ_0^3 в виде $\overline{D}^0$. Уровни первый и третий двойственны в $\hat{\Theta}$ при $n = 4$. Когда построены строки D^1 и $\overline{D}^1$,т. е. к θ_0^1 присоединено разбиение θ_1^1, а к θ_0^3—разбиение θ_1^3, то это значит, что сформированы разбиения второго уровня $\theta_0^1 + \theta_1^1$ и $\theta_0^3 \cdot \theta_1^3$. В структуре в при $n = 4$ второй уровень является двойственным самому себе . Строки D^2 и $\overline{D}^2$ дают разбиения соответственно третьего $\theta_0^1 + \theta_1^1 + \theta_2^1$ и первого $\theta_0^3 \cdot \theta_1^3 \cdot \theta_2^3$ уровней. И наконец, строки $\overline{D}^3$ и D^3 содержат двойственные друг другу тривиальные разбиения: на 16 одноэлементных блоков $\theta_0^3 \cdot \theta_1^3 \cdot \theta_2^3 \cdot \theta_3^3$ и одноэлементное $\theta_0^1 + \theta_1^1 + \theta_2^1 + \theta_3^1$.

Построение разбиения θ_0^{n-1} при условии, что для него уже выбраны из списка P_2 базовая пара $\tilde{\pi}$ и подмножество $\tilde{P}_{\max}$, состоит в следующем. Объединяются в одно множество S_1 все $S_{k1} \in \pi_k$, $\pi_k \in \tilde{P}_{\max}$ $(k = 1, \ldots, |\tilde{P}_{\max}|)$ и в другое множество S_2 все $S_{k2} \in \pi_k$ (ранее нами было принято считать, что для любых двух пар π_i и π_j, согласованных между собой, выполняются $S_{i1} \cap S_{j2} = \varnothing$ и $S_{i2} \cap S_{j1} = \varnothing$). Если $|S_1| > 2^{n-1}$ или $|S_2| > 2^{n-1}$, то удаляем из $\tilde{P}_{\max}$ некоторые пары, чтобы мощности множеств S_1 и S_2 не превышали 2^{n-1}. Затем в одну половину пустой строки D записываем множество S_1, в другую половину S_2. Таким образом оказалось построенным разбиение θ_0^{n-1} в виде $\overline{D}^0 = (\overline{D}_0^0, \overline{D}_1^0)$. Некоторые клетки в $\overline{D}_0$ могут остаться пустыми. Строка $\overline{D}_0$ развязала множества S_1 и S_2, так как они оказались в разных подстроках, т. е. в разных блоках разбиения θ_0^{n-1}. Таким образом, развязаны подмножества S_{k1} и S_{k2} пар π_k, из которых были образованы S_1 и S_2. Состояния внутри подстрок $\overline{D}_0^0$ и $\overline{D}_1^0$ могут как угодно переставляться. И эта возможность будет использоваться при формировании строки $\overline{D}_1 = (\overline{D}_0^1, \overline{D}_1^1, \overline{D}_2^1, \overline{D}_3^1)$, представляющей кроме θ_0^{n-1} еще и разбиение θ_1^{n-1}. Подстроки $\overline{D}_0^0$ и $\overline{D}_1^0$ будут расчленяться пополам, и какие состояния попадут в их половины, зависит от того, какие подмножества состояний надо будет развязать в разбиении θ_1^{n-1}. Один блок этого разбиения составят подстроки $(\overline{D}_0^1, \overline{D}_2^1$, другой— $(\overline{D}_1^1, \overline{D}_3^1)$. Формировать строку $\overline{D}^1$ можно, развязывая поочередно пары из подмножества $\tilde{P}_{\max}$, выделенного из списка P_2 для построения разбиения θ_1^{n-1}. Пусть развязывается пара

$\pi_k = (S_{k1}, S_{k2})$, $\pi_k \in \tilde{P}_{max}$. Все $s_l \in S_{k1}$, которые оказались в правых половинах подстрок $\bar{D}_0^0$ и $\bar{D}_1^0$, перемещаем в левые половины, все $s_l \in S_{k2}$ должны оказаться в правых половинах. Таким образом, все состояния S_{k1} окажутся в одном блоке разбиения θ_1^{n-1}, т. е. в $\bar{D}_0^1$ и $\bar{D}_2^1$, а все состояния S_{k2} — в другом блоке разбиения θ^{n-1} 1, т. е. в $\bar{D}_1^1$ и в $\bar{D}_3^1$. Затем развязывается другая пара из $\tilde{P}_{max}$ и т. д.

Алгоритм построения θ_i^{n-1} $(i = 0, \ldots, n-1)$, т. е. строки $\bar{D}^i$ в виде набора подстрок $(\bar{D}_0^i, \ldots, \bar{D}_{2^i-1}^i)$ на основе $\bar{D}^{i-1} = (\bar{D}_0^{i-1}, \ldots, \bar{D}_{2^{i-1}-1}^{i-1})$, назовем Е2. Если i=0, то D считается пустой.

Алгоритм Е2

1. В списке P_2 отыскивается пара с наименьшим ω, которая назначается базовой — $\tilde{\pi}$.
2. Определяется $\tilde{P}_{max}$ для $\tilde{\pi}$.
3. Берется пара $\pi_k \in \tilde{P}_{max}$ $(k = 1, \ldots, |\tilde{P}_{max}|)$. Для каждого состояния s_l из S_{k1} или S_{k2} выполняется следующее. Если состояние s_l входит в $\bar{D}_l^{i-1}$, то оно помещается при $s_l \in S_{k1}$ в левую половину подстроки и при $s_l \in S_{k2}$ в правую. Если состояние s_l не вошло в $\bar{D}^{i-1}$, то s_l помещается в любую $\bar{D}_j^{i-1}$, имеющую пустую клетку, причем в левую половину при $s_l \in S_{k1}$ и в правую — при $s_l \in S_{k2}$. Состояние s_l помечается.

Если s_l оказывается $(2^{n-i-1}+1)$-м помеченным элементом в соответствующей половине $\bar{D}_l^{i-1}$, то пара π_k не может быть развязана в $\bar{D}^i$ (т. е. в θ_l^{n-i}). Все метки, связанные с π_k, удаляются.

4. Строка, полученная в п. 3, считается строкой $\bar{D}^i$, все метки удаляются; пары, развязанные в θ_l^{n-1}, выбрасываются из P_2.
5. Конец.

Пример 5. Пусть имеется строка (n=3)

$$\bar{D}^0 = (\underbrace{a,\ c,\ b,\ g,}_{\bar{D}_0^0}\ \underbrace{e,\ f,\ d,\ \varnothing}_{\bar{D}_1^0})$$

и список P_2 (из примера 4)

$$\pi_1 = (a,\ b;\ c,\ d);\ \pi_2 = (a,\ e;\ c,\ f);\ \pi_3 = (e,\ f;\ c,\ d);$$
$$\pi_4 = (a,\ f;\ c,\ g);\ \pi_5 = (b,\ g;\ c,\ e).$$

Применим к $\overline{D}^0$ и P_2 алгоритм Е2, чтобы сформировать $D^1 = \{\overline{D}_0^1, \overline{D}_1^1, \overline{D}_2^1, \overline{D}_3^1\}$. В соответствии с п. 1 алгоритма в списке P_2 отыскиваем пару с наименьшим ω. Как видно из примера 4, в нашем списке все пары, кроме π_1, имеют $\omega = 1$ В качестве базовой берем из этих пар первую по порядку, это π_2. Единственное подмножество согласованных с π_2 пар равно $\{\pi_1, \pi_2\}$.

Начнем выполнять п. 3 с пары $\pi_1 = \{a, b, c, d\}$. С подмножеством *(a, b)* пересекается подстрока $\overline{D}_0^0$ с *(c, d)*— подстроки $\overline{D}_0^0$ и $\overline{D}_1^0$. Произведем в подстроках $\overline{D}_0^0$ и $\overline{D}_1^0$ перестановки: *b* и *c* поменяем местами, так как *b* должно быть помещено в левую половину подстроки $\overline{D}_0^0$, а *c* — в правую Пометим a, b, c, d *(a, d* без перестановок оказались на своих местах): $(a, \dot{b}, \dot{c}, g, e, f, \dot{d}, \varnothing)$. Проделав то же самое с парой $\pi_2 = (a, e; c, f)$, получим: $(a, \dot{b}, \dot{c}, g, \dot{e}, \varnothing, \dot{d}, f)$.

Таким образом, развязав пары π_1 и π_2, получили

$$\overline{D}^1 = \underbrace{(a, b}_{\overline{D}_0^1}, \underbrace{c, g}_{\overline{D}_1^1}, \underbrace{e, \varnothing}_{\overline{D}_2^1}, \underbrace{d, f)}_{\overline{D}_3^1}$$

ИЛИ

$$\theta_1^2 = \{\overline{a, b, e}; \overline{c, g, d, f}\}.$$

Начиная выполнять кодирование состояний асинхронного автомата, будем считать, что $n =]\log_2 |S|[$ и D имеет 2^n элементов. В процессе кодирования может оказаться, что достаточно $n_0 < n$ двублочных разбиений для развязывания всех пар из P_2, или возникает другая ситуация: n разбиений не хватает, чтобы развязать все пары из P_2. В этом случае приходится увеличивать n и вместе с тем D до тех пор, пока не окажутся выполненными все условия противогоночного кодирования. Алгоритм противогоночного кодирования состояний асинхронного автомата назовем Э3.

Алгоритм Э3

1. К списку P_2 и пустой строке D с числом элементов 2^n $(n =]\log_2 |S|[)$ применяется алгоритм Е2 n или $n_0 < n$ раз. Применение его прекращается, если все пары из P_2 оказались развязанными. В первом случае выполняется п. 2, во втором — п. 4.
2. Если все пары из P_2 развязаны, то выполняется п. 4.
3. $N := n$, $n := n + 1$, каждая подстрока $\overline{D}_i^{N-1}$ $(i = 0, \ldots, 2^{N-1})$ увеличивается вдвое (клетки, добавленные при этом в $\overline{D}^{N-1}$ пусты).

К строке $D^{\overline{N-1}}$ применяется алгоритм Е2. Если список P_2 оказался исчерпанным, то выполняется п. 4, в противном случае — снова п. 3.

4. Каждому элементу $s_k \in S$ присваивается код $j_{(2)}$, если $d_j = s_k$

5. Конец.

Замечание 3. В результате выполнения п. 3 алгоритма $\Xi 3$ получится, что В $|S| < 2^{n-1}$. этом случае, чтобы улучшить емкостную сложность алгоритма, можно вместо *D* работать с матрицей $\Delta = \|\delta_{ij}\|$ размера $2 \times |S|$, в первой строке которой записаны элементы $s_k \in S$, во второй — номера мест, которые эти элементы занимают в *D*.

Замечание 4. Замечено, что в ряде случаев противогоночное кодирование оказывается неплохим с точки зрения его экономичности. Это можно объяснить тем, что при противогоночном кодировании *k*-множества так или иначе оказываются представленными блоками разбиений булевой структуры (К-множества являются подмножествами соответствующих *k*-множеств или совпадают с ними). Проиллюстрируем работу алгоритма $\Xi 3$ на примере.

Пример 6 . Таблица перехода автомата имеет вид

	σ_1	σ_2	σ_3	σ_4
a	a	c	d	c
b	a	f	c	b
c	f	c	c	c
d	—	d	d	b
e	a	d	c	e
f	f	f	—	e

Список P_2 (состязающихся переходов):

$$\pi_1 = (a,\ b;\ c,\ f);\ \pi_2 = (a,\ e;\ c,\ f);\ \pi_3 = (a,\ c;\ d,\ e);$$
$$\pi_4 = (a,\ c;\ b,\ f);\ \pi_5 = (b.\ f;\ d,\ e);\ \pi_6 = (a,\ d;\ b,\ c);$$
$$\pi_7 = (a,\ d;\ c,\ e);\ \pi_8 = (a,\ c;\ b,\ d);\ \pi_9 = (a,\ c;\ e,\ f);$$
$$\pi_{10} = (b,\ d;\ e,\ f).$$

Задаем $n = 3$. Выполняем Е2 для θ_0^2. Ни одна пара из P_2 не имеет $\omega = 1$, $\omega = 2$ для π_1, π_2, π_8 и т. д Поэтому в качестве базовой для θ_0^2 берем π_1. Подмножества согласованных с нею пар:

$$\tilde{P}_1 = \{\pi_1,\ \pi_2\};\ \tilde{P}_2 = \{\pi_1,\ \pi_7,\ \pi_{10}\};\ \tilde{P}_{\max} = \tilde{P}_2.$$

Объединение пар π_1, π_7, π_{10} дает

$$\overline{D}^0 = (\underbrace{a,\ b,\ d,\ \varnothing}_{\overline{D}_0^0},\ \underbrace{c,\ e,\ f,\ \varnothing}_{\overline{D}_1^0});$$

$$\theta_0^2 = \{\overline{a,\ b,\ d},\ \overline{c,\ e,\ f}\}.$$

Выполняем Е2 для θ_1^2. После удаления из P_2 пар $\pi_1,\ \pi_7,\ \pi_{10}$ $P_2 = \{\pi_2,\ \pi_3,\ \pi_4,\ \pi_5,\ \pi_6,\ \pi_8,\ \pi_9\}$. Для пар $\pi_2,\ \pi_8,\ \pi_9$ $\omega = 1$. Берем в качестве базовой для θ_1^2 пару π_2. Единственное подмножество согласованных с нею пар $P = \{\pi_2,\ \pi_5,\ \pi_6\}$. Его пары будем развязывать в D^1. Начнем с $\pi_2 = (a, e; c, f)$, получаем

$$(\dot{a},\ b,\ d,\ \varnothing,\ \underbrace{\dot{e},\ \varnothing}_{\overline{D}_2^1},\ \underbrace{\dot{c},\ \dot{f}}_{\overline{D}_3^1}).$$

Развязав пару $\pi_5 = (d, e; b, f)$, получим:

$$\overline{D}_1 = (\underbrace{\dot{a},\ d}_{\overline{D}_0^1},\ \underbrace{\dot{b},\ \varnothing}_{\overline{D}_1^1},\ \underbrace{\dot{e},\ \varnothing}_{\overline{D}_2^1},\ \underbrace{\dot{c},\ \dot{f}}_{D_3^1}).$$

Пара π_6 оказывается развязанной в $\overline{D^1}$. Таким образом, разбиение

$$\theta_1^2 = \{\overline{a,\ d,\ e};\ \overline{b,\ c,\ f}\}.$$

Выполняем Е2 для θ_2^2. Все оставшиеся в P_2 пары составляют одно множество $\overline{P} = \{\pi_3,\ \pi_4,\ \pi_8,\ \pi_9\}$. Развязав пару π_3, получим:

$$(\underbrace{\dot{a}}_{\overline{D}_0^2},\ \underbrace{\dot{d}}_{\overline{D}_2^1},\ b,\ \varnothing,\ \underbrace{\varnothing}_{\overline{D}_4^2},\ \underbrace{\dot{e}}_{\overline{D}_5^2},\ \underbrace{\dot{c}}_{\overline{D}_6^2},\ \underbrace{\dot{f}}_{\overline{D}_7^2}).$$

Развязав пару π_4, получим:

$$\overline{D}^2 = (\dot{a},\ \dot{d},\ \varnothing,\ \dot{b},\ \varnothing,\ \dot{e},\ \dot{c},\ \dot{f}).$$

Пары $\pi_8,\ \pi_9$ оказываются развязанными в $\overline{D^2}$. Таким образом, разбиение

$$\theta_2^2 = \{\overline{a,\ c};\ \overline{d,\ b,\ e,\ f}\}.$$

Итак, построенытри разбиения, которые развязывают все пары списка P_2 Таким образом, алгоритм Б3 выполнен, получен код: $a = 000$, $b = 011$, $c = 110$, $d = 001$, $e = 101$, $f = 111$.

6.4.3. Противогоночное кодирование в сочетании с экономичным

После того как развязаны все пары состязающихся переходов или *k*-множеств, возникает одна из двух ситуаций.

1. Оказывается, что для развязывания всех пар списка P_2 достаточно $n_0<n$ $(n=]\log_2|S|[)$ разбиений, т. е. сформирована строка $\overline{D}^{n_0-1}$, состоящая из 2^{n_0} подстрок, в каждой из которых по 2^{n-n_0} элементов. Эти элементы могут подвергнуться переупорядочению внутри каждой подстроки, поскольку им еще не присвоены кодирующие слова длины *n,* хотя вся группа элементов, составляющая подстроку $\overline{D}^{n_0-1}$, уже получила код, равный $i_{(2)}$. Так, если *a, b, c, d* составляют подстроку $\overline{D}_3^{n_0-1}$ $(n=5)$, то каждый из них уже имеет часть кодирующего слова 011.., кодирующие слова для состояний *a, b, c, d* будут различаться в двух младших разрядах.

2. Для развязывания всех пар списка P_2 потребовалось $n_0 \geqslant n$ разбиений, т. е построена строка $\overline{D}^{n_0-1}$, каждая подстрока которой содержит один элемент, и это либо состояние из множества S, либо знак $\varnothing$ («пусто»).

В первой ситуации условия экономичного кодирования реализуются путем упорядочения элементов подстрок $\overline{D}_i^{n_0-1}$. Для этого может быть использован алгоритм $\Xi 2$ в модифицированном виде. Модификация связана с тем, что подстроки $\overline{D}_i^{n_0-1}$ нельзя менять местами; произвольная перестановка их разрушит построенные двублочные разбиения (этот запрет не относится к циклическим перестановкам, но такие перестановки не дают нового качества). Из запрета на перестановки подстрок в строке $\overline{D}^{n_0-1}$, например, следует, что никогда не получит соседних кодов пара состояний *(g, h),* если *g* оказалось в $\overline{D}_1^{n_0-1}$, а *h* —в $\overline{D}_2^{n_0-1}$. Никакими перестановками внутри этих подстрок нельзя добиться того, чтобы пара *(g, h)* оказалась блоком в каком-либо разбиении θ_j^1, так как части кодирующих слов: 01... для g и 10... для *h* — уже различаются в двух разрядах. Точнее, модификация касается в основном алгоритма E1, который в этом случае не может быть применен к произвольным подстрокам $\overline{D}_i^{n_0-1}$, а либо к каждой подстроке отдельно, либо к парам подстрок $\overline{D}_i^{n_0-1}$, $\overline{D}_{i+1}^{n_0-1}$ $(i=0, 2, 4, ...)$, либо к четверкам $\overline{D}_i^{n_0-1}$, $\overline{D}_{i+1}^{n_0-1}$, $\overline{D}_{i+2}^{n_0-1}$, $\overline{D}_{i+3}^{n_0-1}$ $(i=0, 4, ...)$ и т. д.

Список P_1 в этой ситуации формируется на основе тех К-множеств, которые не могут быть реализованы одним интервалом в рамках той

булевой структуры, D-представлением которой является строка $\overline{D}^{n_0-1}$. Совокупность таких (нереализованных) K-множеств обозначим через $\hat{K}$.

Не будем записывать подробно модифицированный алгоритм, а продемонстрируем его на примере.

Пример 7. Пусть имеется строка

$$\overline{D}_1 = (\underbrace{a,\ b,\ c,\ d}_{\overline{D}_0^1},\ \underbrace{e,\ f,\ \varnothing,\ h}_{\overline{D}_1^1},\ \underbrace{i,\ j,\ k,\ l}_{\overline{D}_2^1},\ \underbrace{m,\ n,\ \varnothing,\ p}_{\overline{D}_3^1})$$

и совокупность $\hat{K}$

$$K_1 = \{a,\ f\};\ K_2 = \{a,\ i,\ j,\ m\};\ K_3 = \{i,\ k\};\ K_4 = \{a,\ i,\ j,\ k\};$$
$$K_5 = \{l,\ n\};\ K_6 = \{b,\ l\};\ K_7 = \{c,\ d\};\ K_8 = \{e,\ h\}.$$

Составим список взвешенных пар P_1:

$$(a, f)\,(2),\ (a,\ j)\,(2),\ (a,\ m),\ (f,\ j),\ (f,\ m),\ (j,\ m),\ (i,\ k)\,(2),$$
$$(a,\ i),\ (a,\ k),\ (i,\ j),\ (j,\ k),\ (l,\ n),\ (b,\ l),\ (c,\ d),\ (e,\ h).$$

Строим θ_0^1. Для каждой подстроки $\overline{D}_i^1$ выбираем из P_1 по две непересекающиеся пары, вложенные в множество элементов подстроки и имеющие по сравнению с другими вложенными наибольший вес. Если для какой-либо подстроки в списке P_1 не найдется подходящих пар, то ее элементы делятся на пары произвольно. Для $\overline{D}_0^1$ нашли пару *(c, d)*, для $\overline{D}_1^2 = (e,\ h)$, для $D_2^2 = (i,\ k)$, элементы подстроки $\overline{D}_3^2$ расчленяем на пары произвольно, получаем

$$D^0 = (\underbrace{c,\ d}_{D_0^0},\ \underbrace{a,\ b}_{D_1^0},\ \underbrace{e,\ h}_{\cdot},\ \underbrace{f,\ \varnothing}_{\cdot},\ \underbrace{i,\ k}_{\cdot},\ \underbrace{j,\ l}_{\cdot},\ \underbrace{m,\ n}_{\cdot},\ \underbrace{\varnothing,\ p}_{D_7^0}).$$

Сделаем небольшое отступление, которое касается обозначений. Может возникнуть недоумение, отчего появилась строка D^0, когда в начале примера уже была $\overline{D^1}$. Мы ввели обозначение строки $\overline{D^i}$ $(i<n-1)$, соответствующей двублочным разбиениям, чтобы отличить от строки $D^i\,(i<n-1)$, представляющей разбиения первого уровня. Строки же $\overline{D}^{n-1}$, D^{n-1} являются D-представлением структуры, и по каждой из них можно восстановить и двублочные разбиения, и разбиения на блоки мощности 2 каждый. Очевидно, что строка $\overline{D^1}$ не дает нам ни одного разбиения первого уровня, поэтому построение таких разбиений мы начинаем с θ_0^1, и, значит, формируем строку D^0, не забывая, что мы ограничены закрепленным расположением подстрок $\overline{D}_i^1$.

Из списка P_1 удаляем все пары, вошедшие в θ_0^1. Строим θ_1^1.

Снова работаем отдельно с каждой подстрокой D_i^1: к сформирован-

ным в ней D^0_{2i}, D^0_{2i+1} применяем алгоритм Е1, чтобы получить D^1_i. Для этого отыскиваем в P_1 пару, вложенную в множество элементов подстроки $\bar{D}^1_i$ и имеющую наибольший вес по сравнению со всеми вложенными. Для $\bar{D}^1_0$ такой пары в списке P_1 не нашлось, для $\bar{D}^1_1$ и $\bar{D}^1_3$ также нет подходящих пар в списке P_1. Для $\bar{D}^1_2$ подходящей является пара $(j, k) \in P_1$. Чтобы реализовать ее, необходимо в $\bar{D}^1_2 = (D^0_4, D^0_5)$ поменять местами i и k. Никаких других перестановок не требуется,

$$D^1 = (\underbrace{c,\ d,\ a,\ b}_{D^1_0},\ \underbrace{e,\ h,\ f,\ \varnothing}_{D^1_1},\ \underbrace{k,\ i,\ j,\ l}_{D^1_2},\ \underbrace{m,\ n,\ \varnothing,\ p}_{D^1_3}).$$

Строим θ^1_2. Для формирования D^2_0 отыскиваем в списке P_1 пару, один элемент которой входит в D_0, а другой — в D^1_1. Это пара *(a, f)*. Для формирования D^2_1 по аналогичным признакам подходит пара *(j, m)*. Применяя алгоритм Е1 сначала к строке (D^1_0, D^1_1) затем к (D^1_2, D^1_3), получаем:

$$D^2 = (\underbrace{a,\ b,\ c,\ d,\ f,\ \varnothing,\ e,\ h}_{D^2_0},\ \underbrace{j,\ l,\ k,\ i,\ m,\ n,\ \varnothing,\ p}_{D^2_1}).$$

И наконец, для формирования θ^1_3, т. е. строки D^3, следует взять в качестве начальной пары (a, j). Элементы этой пары в D^2 оказались на подобающих им в строке D^3 местах. Таким образом, $D^3 = D^2$.

Кодирование завершается присвоением кодирующих слов состояниям в соответствии с их местами в D^3. Заметим, что парам *(a, m)* и *(f, j)* из-за запрета на перестановки подстрок в $\bar{D}^1$ в принципе не могут быть присвоены соседние коды.

Обратимся ко второй ситуации. Построение структуры $\hat{\Theta}$ завершено, сформировано ее D-представление, и каждому состоянию может быть присвоено n_0-разрядное кодирующее слово. В этом случае либо приходится удовлетвориться степенью экономичности полученного кода, либо увеличить разрядность кода на единицу и попытаться учесть условия экономичности. С увеличением разрядности кода на единицу строка $\bar{D}^{n_0-1}$ вдвое увеличивается по длине, так как в каждую ее подстроку включается один «пустой» элемент. Строка становится сильно «разреженной» в том смысле, что в ней «пустых» элементов столько же или больше, чем состояний в S. В этом случае целесообразно работать не с парами из списка P_1 (сформированного на основе $\hat{K}$), а прямо с К-множествами из совокупности $\hat{K}$ и так размещать состояния в своих подстроках $\bar{D}^{n_0-1}_i$, чтобы К-множества

оказывались реализованными одним интервалом каждое.
Продемонстрируем такое размещение на примере.

Пример 8. Возьмем строку $\overline{D^2}$ из примера 6:

$$\overline{D}^2 = (\overset{000}{a},\ \overset{001}{d},\ \overset{011}{\varnothing},\ \overset{101}{b},\ \overset{110}{\varnothing},\ \overset{111}{e},\ c,\ f).$$

Введя в каждую ее подстроку $\overline{D}_i^2$ «пустой» элемент (n=4), получим:

$$\overline{D}^2 = (\underbrace{a, \varnothing}_{\overline{D}_0^2}, \underbrace{d, \varnothing}_{\overline{D}_1^2}, \varnothing,\ \varnothing,\ \underbrace{b, \varnothing}_{\overline{D}_3^2},\ \underbrace{\varnothing, \varnothing},\ \underbrace{e, \varnothing}_{\overline{D}_5^2}, \underbrace{c, \varnothing}_{\overline{D}_6^2}, \underbrace{f, \varnothing}_{\overline{D}_7^2}). \qquad \textbf{(8)}$$

Из таблицы перехода автомата, для которого выполнялось противогоночное кодирование состояний в примере 6, выделим K- множества:

$$K_1 = \{a,\ b,\ e\};\ K_2 = \{c,\ f\};\ K_3 = \{a,\ c\};\ K_4 = \{b,\ f\};$$
$$K_5 = \{d,\ e\};\ K_6 = \{a,\ d\};\ K_7 = \{b,\ c,\ e\};\ K_8 = \{a,\ c\};$$
$$K_9 = \{b,\ d\};\ K_{10} = \{e,\ f\}.$$

Из (8) видно, что одним интервалом реализуются множества K_2, K_3, K_4, K_5, K_6, K_9, K_{10}. Из девяти К-множеств ($K_3 = K_8$) остаются нереализованными два: K_1, K_7. Естественно начинать с тех К-множеств, для реализации которых потребуется наименьшее число перестановок, т. е. будет затронуто наименьшее число подстрок. Число затрагиваемых подстрок определяется тем, в каком количестве разрядов кода длины n_0 (здесь $n_0 = 3$) различаются состояния, входящие в К-множество. Код длины n_0 для каждого состояния определяется местом этого состояния в строке $\overline{D^2}$, т. е. равен двоичному номеру соответствующей подстроки $\overline{D}_i^2$. Так, n_0-разрядные коды состояний, составляющих K_1, разнятся во всех разрядах: a = 000, b= 011, e=101, значит, все восемь подстрок $\overline{D}_i^2$ должны быть затронуты. То же самое имеет место и для K_7. Поскольку оба эти К-множества с рассматриваемой точки зрения одинаковы, то берем первое по порядку, это K_1. Для каждой подстроки D_i^2 делаем следующее. Если «непустой» элемент подстроки $\overline{D}_i^2$ принадлежит K_1, то помещаем его в левую клетку подстроки, если не принадлежит — в правую. Получаем

$$\overline{D}_3 = (\underbrace{a}_{\overline{D}_0^3},\ \underbrace{\varnothing}_{\cdot},\ \underbrace{\varnothing}_{\cdot},\ \underbrace{d}_{\overline{D}_3^3},\ \varnothing,\ \varnothing,\ \underbrace{b}_{\overline{D}_6^3},\ \varnothing,\ \varnothing,\ \varnothing,\ \underbrace{e}_{\overline{D}_{10}^3},\ \varnothing,\ \varnothing,\ \underbrace{c}_{\overline{D}_{13}^3},\ \varnothing,\ \underbrace{f}_{\overline{D}_{15}^3}).$$

Поскольку всех подстрок $\overline{D}_i^2$ коснулись перестановки, то реализовать и K_7 одним интервалом не удастся.

В заключение сделаем два замечания.

Замечание 5. Основным содержанием описанных алгоритмов кодирования является построение D-представления булевой структуры

Θ. В сочетании с тем или иным способом построения строки могут быть использованы какие угодно эвристические приемы.

Замечание 6. Две задачи: кодирование состояний логического преобразователя и построение компактного матричного представления этого преобразователя — могут быть решены как одна задача сквозного синтеза логического преобразователя. Наметим путь решения такой задачи. Первым этапом решения является кодирование состояний, т. е. построение D-представления булевой структуры $\hat{\Theta}$. Второй этап — это решение задачи кратчайшего покрытия совокупности К-множеств блоками разбиений построенной структуры. При решении задачи покрытия используется сформированная на первом этапе строка *D*.

Ищется хорошее приближение к кратчайшему покрытию. Для этого могут быть использованы любые эвристические приемы, характерные для задач покрытия и достаточно полно освещенные в литературе. Регулярную часть алгоритма покрытия (т. е. ту его часть, которая выполняется одинаково при любых эвристиках) можно построить следующим образом. На каждом шаге алгоритма блоками разбиений структуры покрывается одно из К-множеств (это может быть, например, К-множество, не имеющее пересечений с другими) или некоторая часть одного из К-множеств (например, та, которая является пересечением нескольких К-множеств). Пусть на *i*-м шаге покрывается множество K_i. Из строки *D*, полученной на первом этапе, извлекаются блоки разбиений, содержащиеся в K_i (естественно, что берутся блоки, максимальные для K_i), при этом используются формулы (15) — (19). Каждый блок B_j, вошедший в покрытие (он же булев интервал), представляется троичным вектором α_j, и этот вектор становится входной частью обобщенного интервала. Выходная часть β_j этого обобщенного интервала определяется К-множествами, содержащими блок B_j. Обобщенный интервал (α_j, β_j) реализуется строкой матрицы **M**. Процесс продолжается до тех пор, пока вся совокупность К-множеств не окажется покрытой блоками разбиений. Матрица **M**, реализующая логический преобразователь, будет иметь столько строк, сколько блоков содержит покрытие совокупности К-множеств.

7. Методы проектирования автоматов

7.1. Основные этапы проектирования автоматов

Если рассматривать автомат не как устройство, а изучать его строение (структуру), то следует представлять этот автомат не в виде машины Тьюринга, а в виде блок-схемы, изображенной на рис. 1.

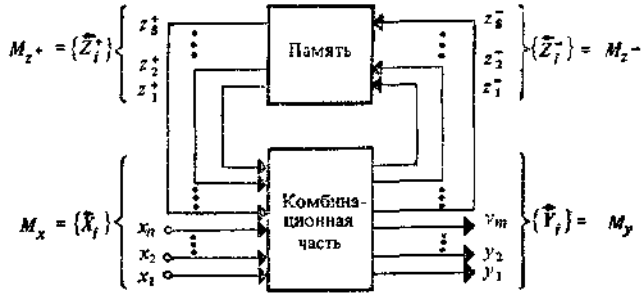


Рис. 1

Рассмотрим проблему проектирования автомата. На рис. 1

M_x *-множество входных векторов*

$$X = x_1^{\sigma_1} x_2^{\sigma_2} \dots x_n^{\sigma_n};$$

M_y *— множество выходных векторов*

$$Y = y_1^{\sigma_1} y_2^{\sigma_2} \dots y_m^{\sigma_m},$$

M_{z-} —множество векторов, характеризующих *входные каналы обратной связи* (памяти),

$$Z^- = (z_1^-)^{\sigma_1} (z_2^-)^{\sigma_2} \dots (z_s^-)^{\sigma_s},$$

M_{z+} — множество векторов, характеризующих *выходные каналы обратной связи,*

$$Z^+ = (z_1^+)^{\sigma_1} (z_2^+)^{\sigma_2} \dots (z_s^+)^{\sigma_s}.$$

Каждый из каналов в случае *k*-значной логики может находиться в одном из *k* значений $\sigma \in (0, 1, \dots, k-1)$.

Условимся обозначать переменную α, равную $\sigma \in (0, 1, 2, \dots, k-1)$, как α^σ. Тогда правила вывода в случае автоматной грамматики удобней определить как подстановку $XZ^+ \to Z^- Y$, в которой начальное значение вектора Z^+ и последующие его значения получаются приравниванием значению вектора Z^-, вычисленному на предыдущем шаге, т. е.

$Z^+(t=0) = Z_0^+,\ Z^+(t+\tau) = Z^-(t),$

где τ — некоторая постоянная времени.

Векторы XZ^+ и Z^-Y получаются приписыванием справа вектора Z^+ к *X* и Y к Z^- соответственно:

Состояния каналов обратной связи в дальнейшем будем называть *внутренними состояниями автомата,* а постоянную времени τ — временем перехода из одного внутреннего состояния в другое, причем в зависимости от назначения автомата и его реализации τ может быть постоянной для данного автомата или же зависеть от изменения вектора *X.* В первом случае автомат называют *синхронным,* во втором — *асинхронным.*

При заданном $(Z^+)_0$ последовательность входных векторов *X (входная последовательность)* однозначно определяет последовательность выходных векторов *Y(выходную последовательность).* Проиллюстрируем это на следующем примере.

Пусть имеется устройство с входным каналом *x,* каналом обратной связи *z* и выходным каналом *y*, реализующее отображение $XZ^+ \to Z^-Y$, заданное в виде таблицы (табл. 1),

Таблица 1

x	z^+	z^-	y
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	0

Определим выходную последовательность, если входная последовательность имеет вид 101001 и начальное значение вектора Z^+ равно 0. В начальный момент времени вектор XZ^+ равным 10, определяет Z^-Y = 01 (третья строка таблицы). Через промежуток времзни, равный τ , XZ^+ = 00 определит Z^-Y = 11 (первая строка) и т. д. Запишем определение выходной последовательности в виде табл.2.

Таблица 2

Время	x	z^+	z^-	y
0	1	0	0	1
τ	0	0	1	1
2τ	1	1	0	0
3τ	0	0	1	1
4τ	0	1	1	0
5τ	1	1	0	0

Следовательно, $101001 \xrightarrow{Z_0^+ = 0} 110100$.

При $Z_0^+ = 1$, согласно заданному автоматному отображению входная последовательность 101001 преобразуется в выходную последовательность 010100.

Таким образом, автоматное отображение $XZ^+ \to Z^- Y$ однозначно определяет выходную последовательность (Y_i) по заданной входной последовательности (X_i) и начальному внутреннему состоянию Z^+_0: $(X_i) \xrightarrow{Z_0^+} (Y_i)$.

Одной из основных характеристик автомата является объем его памяти. **Число внутренних состояний автомата называется *объемом памяти автомата.***

Как известно, **преобразование информации является результатом выполнения некоторого алгоритма, при этом *операционный автомат* реализует шаги алгоритма, а *управляющий автомат* — порядок выполнения шагов.** Операционный и управляющий автоматы различаются не только своим назначением, но и объемом памяти. В операционном автомате происходит преобразование информации, которая задана в виде некоторого множества чисел, записанных в регистрах. Практически объем памяти операционного автомата бесконечен. Так, например, блок регистров ЭВМ, входящей в операционный автомат и состоящий из 22 16-разрядных двоичных регистров, имеет объем памяти $2^{352} > 10^{100}$ бит. Объем памяти управляющих автоматов обычно составляет от нескольких десятков до нескольких десятков тысяч бит, т. е. является небольшим по сравнению с объемом памяти операционных автоматов.

Рассмотрим взаимодействие операционного и управляющего автомата (рис. 5).

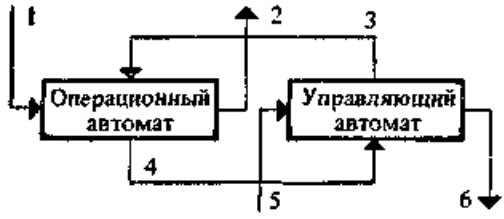


Рис. 5

На вход операционного автомата (канал *1)* поступает преобразуемая информация; с выхода операционного автомата (канал 2) снимаются результаты преобразований; по каналу *3* поступают управляющие воздействия, соответствующие реализуемому алгоритму; по каналу *4* на вход управляющего автомата поступают признаки, харак-

теризующие преобразуемую информацию; по каналу *5* — сигнал, определяющий выполняемое преобразование и его начало; по каналу 6 — сигнал окончания операции. Каналы *1* и 2 называются *информационными,* каналы *3 — 6 — управляющими.* Рассмотрим, например, выполнение операции сложения в арифметическом устройстве (АУ) ЭВМ. В данном случае операционным автоматом является арифметическое устройство, т. е. регистры, сумматор и связи между ними, а управляющим автоматом — устройство управления АУ. При выполнении алгоритма сложения по каналу *1* из запоминающего устройства (ЗУ) машины в соответствующие регистры АУ записываются первое и второе слагаемые; по каналу 5 из центрального устройства управления (ЦУУ) машины поступает код операции («сложение»), по каналу 3 из управляющего устройства АУ, согласно алгоритму сложения, посылаются управляющие сигналы: сдвиг регистров, возбуждение соответствующих шин в сумматоре, запись суммы в регистр результата и другие сигналы. По каналу *4* поступают признаки (например, содержание знаковых разрядов в сумматоре, используемое для обнаружения нарушения нормализации), определяющие ход дальнейшего управления. По каналу *6* в ЦУУ передается сигнал окончания выполнения операции. По каналу 2 в ЗУ посылается результат операции.

ЭВМ - сложный преобразователь информации, и ее целесообразно рассматривать, следуя В. М. Глушкову, как *композицию пар автоматов* каждая из которых состоит из операционного и управляющего автоматов. При этом каждое устройство ЭВМ (ввода, вывода, ЗУ, ЦУУ) представляется, аналогично арифметическому устройству, в виде пары или нескольких пар таких автоматов. Такое представление ЭВМ диктуется не только удобством анализа и синтеза ЭВМ, но и является естественным развитием структуры ЭВМ. Как один из методов увеличения производительности ЭВМ используется мультипрограммный режим, что, в частности, уменьшает задержки, вызываемые низкой скоростью внешних устройств и несоответствием скоростей работы внешних и арифметического устройств. Для реализации мультипрограммного режима ЭВМ необходима некоторая «самостоятельность» отдельных устройств, т. е. возможность хранения, преобразования информации и управления этим преобразованием внутри устройства. Другими словами, необходимо, чтобы каждое устройство представляло собой композицию операционного и управляющего автоматов. Как иллюстрацию такого положения приведем блок-схему вычислительной машины (рис. 6).

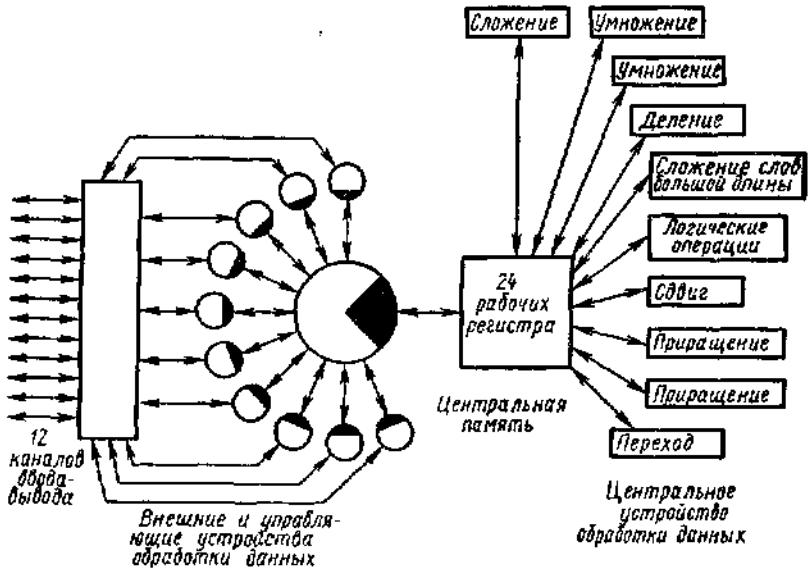


Рис. 6

В ЭВМ сходная информация, поступающая по 12 каналам ввода — вывода, сначала подается на периферийные устройства обработки информации. Каждое периферийное устройство представляет собой композицию операционного и управляющего автомата. Эти небольшие по объему устройства могут либо выдать преобразованные результаты в устройство вывода, либо направить их в центральную память, представляющую собой целую иерархию запоминающих устройств. Последняя операция выполняется с помощью централизованной системы управления, которая распределяет «работу» между специализированными устройствами обработки информации, имеющими высокое быстродействие. Каждое из этих устройств представляет собой композицию операционного и управляющего автоматов. Каждая пара таких автоматов выполняет одну или несколько операций, например операцию умножения.

При мультипрограммной работе ЭВМ осуществляется принцип децентрализованной работы отдельных устройств. В пределе принцип децентрализации переходит в принцип абсолютной децентрализации или в принцип Холланда, предложившего полностью распределенную сеть устройств обработки информации, каждое из которых представляет собой композицию операционного и управляющего автоматов. Все эти устройства работают независимо. При такой

структуре ЭВМ реализуется концепция «рабочих программ, плавающих в море оборудования».

Одним из этапов развития этой тенденции является концепция *распределенной обработки информации,* реализуемая в виде сетей ЭВМ, вычислительных центров коллективного пользования и т. д.

При проектировании ЭВМ можно выделить три основных этапа: *системный, логический* и *технический.*

На этапе системного проектирования строится композиция пар автоматов — операционных и управляющих (общая блок-схема ЭВМ), определяются необходимый объем памяти автоматов, их взаимодействие на основе выбора системы команд, внешние и внутренние языки машины и т. д. **Исходной информацией на системном этапе является совокупность классов задач, для решения которых предназначена проектируемая машина, и ее параметров (быстродействия, стоимости, габаритных размеров и др.).**

Рассматриваемые на этапе системного проектирования автоматы имеют большой объем памяти, превышающий, как уже отмечалось, 10^{100} бит. Поэтому задачи системного этапа решаются с помощью **программного моделирования**.

Процесс решения задачи системного этапа методом программного моделирования обычно состоит из следующих шагов:

а) составление математической модели, отражающей важные свойства моделируемого устройства или машины в целом;

б) построение моделирующего алгоритма, запись его на некотором языке, предназначенном для описания моделей ЭВМ;

в) реализация моделирующего алгоритма на ЭВМ;

г) анализ результатов и корректировка модели устройства или машины.

Программное моделирование позволяет определить основные характеристики проектируемой машины, «узкие» места ее структуры.

На этапе логического проектирования ЭВМ синтезируются непосредственно логические (функциональные) схемы всех блоков машины. Исходной информацией для этого этапа являются алгоритмы функционирования блоков.

На этапе *технического проектирования* на основе логических схем строятся принципиальные монтажные схемы и готовится техническая документация для производства ЭВМ.

***Логическое проектирование* заключается в синтезе как операционных, так и управляющих автоматов.** В силу практически бесконечного объема памяти операционных, автоматов последние в синтезируются с помощью программного моделирования. И если для формализованного синтеза операционных автоматов необходимо

развитие теории бесконечных автоматов, то для формализации синтеза управляющих автоматов возможно применение теории конечных автоматов.

Согласно этой теории, при проектировании управляющих автоматов будем различать два основных этапа: *построения автоматного оператора* и *структурного синтеза автомата.*

I. Этап построения автоматного оператора. На этом этапе производится построение *системы выходных функций* $Y = f(X, Z^{\tau})$ и *системы функций возбуждения* $Z^{-} = \varphi(X, Z^{+})$. *Автоматное отображение,* записанное в виде системы выходных функций и функций возбуждения, будем называть *автоматным оператором.*

Этап построения автоматного оператора, в свою очередь, состоит из трех подэтапов: *алгоритмического, абстрактного* и этапа *кодирования (размещения) внутренних состояний автомата.*

На этапе *алгоритмического* проектирования заданный оператор *А* оформляемся к виде алгоритма исходя из поставленных требований (простоты выполнения операций, быстродействия, максимального уменьшении аппаратурных затрат и др.). Поиск оптимального алгоритма по заданному оператору *А* можно представить в виде дерева, каждая висячая вершила которого соответствует определенному алгоритму, т. е. определенной композиции операционного и управляющего автоматов. Для поиска оптимального алгоритма необходим перебор всех висячих вершин, число которых неизвестно и определяется степенью развития рассматриваемых преобразований. Пусть, например, необходимо синтезировать арифметическое устройство, реализующее четырехместную операцию суммирования. В зависимости от выбора алгоритма, реализующего это задание, получаем определенную блок-схему синтезируемого устройства, характеризующуюся аппаратурными затратами и быстродействием. В данном случае можно предложить, например, три варианта алгоритма.

1-й вариант. Суммируем два первых числа, к полученной сумме прибавляем третье число и к вновь полученной сумме прибавляем последнее число. Этот вариант алгоритма характеризуется временем выполнения операции τ_1 и аппаратурными затратами в виде одного сумматора, четырех регистров, устройства управления и необходимых каналов связи. Этому алгоритму соответствует блок-схема, изображенная на рис. 7, *а.*

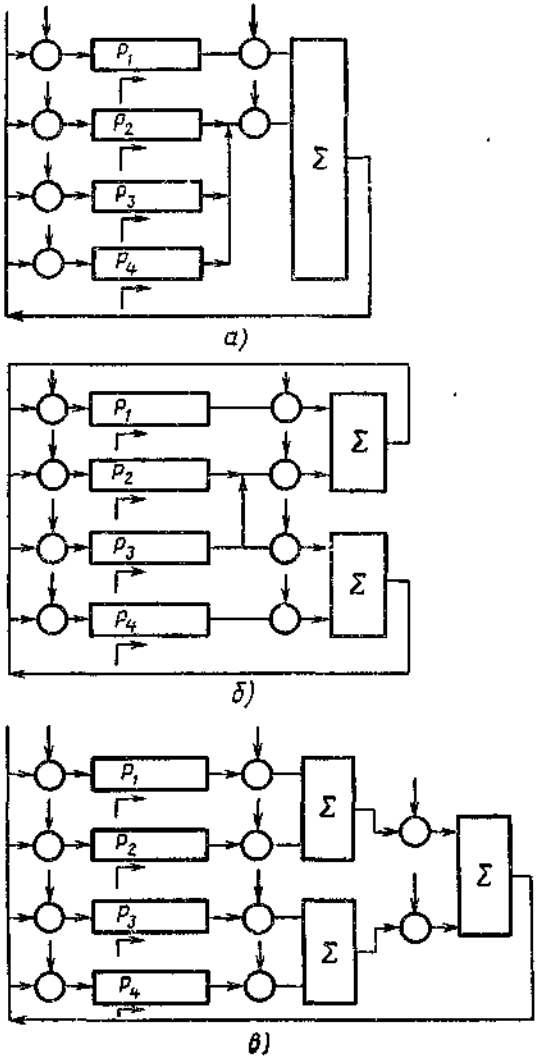


Рис.7.

2-й вариант. Суммируем одновременно первое число со вторым, третье — с четвертым, результаты записываем соответственно в первый и третий регистры, затем суммируем их на первом сумматоре. Окончательный результат записываем в первом регистре. Этот вариант выполнения задания A характеризуется временем выполнения τ_2 и

аппаратурными затратами, представленными на блок-схеме этого варианта (рис. 7, б).

3-й вариант. Суммируем первые два числа на одном сумматоре, вторые два — на втором сумматоре, полученные суммы — на третьем сумматоре, результат записываем в первый регистр. Этот вариант характеризуется временем выполнения операции τ_3 и аппаратурными затратами, приведенными на рис. 7, *в*.

Тот или иной вариант алгоритма, реализующего задание *A,* выбирается исходя из конкретных ограничений на аппаратурные затраты и на время выполнения заданных операций.

Каждая висячая вершина дерева поиска оценивается временем выполнения данного преобразования и сложностью аппаратуры (сложностью операционного и управляющего автоматов). Сложность операционного автомата оценивается непосредственным подсчетом аппаратурных затрат. Сложность управляющего автомата можно оценить с помощью понятия производной от модели.

На абстрактном этапе решается задача минимизации объема памяти автомата. На этапе кодирования (размещения) внутренних состояний автомата каждому внутреннему состоянию автомата сопоставляется код, т. е. определенная совокупность состояний (значений) элементов памяти.

В результате выполнения двух рассмотренных этапов составляют системы выходных функций и функций возбуждения автомата — строят **автоматный оператор**. После построения автоматного оператора переходят к структурному синтезу.

II. Этап структурного синтеза автомата. Этап структурного синтеза автомата заключается в построении из заданных элементов логической (функциональной) схемы автомата, реализующей полученный автоматный оператор. Более подробно структурный синтез рассмотрим ниже.

7.2. Арифметические основы операционных автоматов

Вычислительный процесс в ЭВМ, реализующий программу, представляет собой параллельно-последовательные преобразования чисел. Каждое число представляет собой последовательность цифр. Слово «цифра» — перевод арабского слова «сыфр» на латинский язык, которое, в свою очередь, является переводом древнеиндийского слона (алфавит «деванагари») «сунья», что означает «пустое место»

(«разряд»), в которое помещается числовой знак при задании количественных соотношений.

Системой счисления или *нумерацией* называется совокупность приемов и правил для обозначения и наименования чисел. Система счисления задает правила кодированной записи количественных эквивалентов, позволяющие для каждого количества однозначно получать его кодовую запись и по каждой кодовой записи — соответствующий ей количественный эквивалент. **Множество элементарных знаков, используемых для кодирования, называют *цифрами* системы счисления.**

Системы счисления бывают *непозиционные* и *позиционные.* В непозиционных системах каждой цифре однозначно сопоставлен некоторый стандартный количественный эквивалент, а количественный эквивалент кода числа вычисляется как некоторая функция от количественных эквивалентов цифр, входящих в запись этого кода. Примером такой системы является система счисления, в которой используется только одна цифра, например 1. Этой цифре сопоставлен количественный эквивалент, равный единице. Тогда код 111 111 означает количественный эквивалент шесть, функцией при вычислении количественного эквивалента кода здесь является **функция сложения**. В позиционных системах каждой цифре сопоставляется некоторый количественный эквивалент не однозначно, а в зависимости от ее положения в коде числа. Будем рассматривать только линейно упорядоченные записи. Выберем в записи начало отсчета (нулевой разряд). Разряды влево от него будем нумеровать 1,2,..., а вправо от него — 1, —2, … Каждой цифре a_i, стоящей в разряде с номером j, сопоставляется количественный эквивалент $\varphi(a_i, j)$. Функция φ либо одинакова для всех разрядов, либо ее вид изменяется от разряда к разряду. В дальнейшем будут рассматриваться только позиционные системы с одинаковой функцией φ для всех разрядов. Всякая позиционная система задается тремя компонентами: <*A*, φ, F>. Здесь *A — множество цифр системы;* φ — функция, определяющая для цифр в каждом разряде их *количественный эквивалент; F* — функция, определяющая по количественным эквивалентам в записи числа *количественный эквивалент самого числа.*

В зависимости от вида функции *F* выделим два типа систем счисления: *аддитивные* и *мультипликативные.* В системах первого типа F — функция *сложения,* а в системах второго типа — функция *умножения.* Другие виды *F* рассматривать не будем.

Если для любых цифр A и любого j имеет место равенство $\varphi(a_i, j) = S^j \cdot \varphi(a_i, 0)$, то говорят о системе счисления с *основанием S.* Если $\varphi(a_i, j) = p_j \varphi(a_i, 0)$ и p_j не совпадают при различных j, то система *весомозначного типа,* а p_j — *веса* разрядов.

Если в системе с основанием S множество цифр состоит из $0, 1, \ldots, S-1$, то система имеет *естественное множество цифр (естественная система счисления);* если $S = m + k + 1$ и множество цифр — $\{-m, -m+1, \ldots, -1, 0, 1, \ldots, k\}$, то при $m = k$ система имеет *симметрическое множество цифр (симметрическая система счисления),* при $k > m$ или $m > k$ система имеет *асимметрическое* соответственно *в положительную* или *отрицательную сторону множество цифр (асимметрическая система счисления).*

Наиболее часто используют естественные системы счисления с натуральным основанием. Любое число x в системе с основанием S представимо в следующем виде:

$$x = \sum_{i=-\infty}^{\infty} (a_i) S^i, \qquad (1)$$

где (a_i) — количественный эквивалент цифры a_i в нулевом разряде.

Сдвиг целого числа x в естественной системе счисления с натуральным основанием S на один разряд вправо означает целочисленное деление x на S. Полученный при этом остаток является цифрой, находящейся в нулевом разряде до сдвига. Алгоритм перевода целых чисел из системы счисления с натуральным основанием R в систему счисления с натуральным основанием Q заключается в следующем. Число x, записанное в системе счисления с основанием R, делят на Q по правилам деления в системе счисления с основанием R до получения остатка. Если частное от деления не 0, то оно становится делимым и процесс деления на Q продолжается. Как только очередное частное будет равно нулю, то процесс деления на Q прекращают, запись остатков в порядке, обратном их получению («от последнего к первому»),, дает запись числа x в системе счисления с основанием Q.

Сдвиг дробного числа x в естественной системе счисления с натуральным основанием S на один разряд влево означает умножение x на S. Полученная при этом целая часть, быть может равная нулю, является цифрой, находящейся в — 1-м разряде до сдвига. Отсюда алгоритм перевода дробных чисел (R-ичных дробей) из системы счисления с натуральным основанием R в систему счисления с основанием Q состоит в следующем. Дробную запись числа x умножают на Q по правилам умножения в системе счисления с основанием R. В полученном произведении отделяют целую часть

(может быть, нулевую). Дробную часть произведения вновь умножают на *Q*, далее выделяют целую часть произведения и т. д. Умножение продолжают *k* раз при переводе с точностью до Q^{-k}. Для получения записи числа *x* в системе счисления с основанием *Q* (с точностью до Q^{-k}) выписывают после запятой все целые части произведений в порядке их получения.

Рассмотрим пример использования этих алгоритмов при переводе десятичного числа 11.87 в пятиричную систему. Произведем целочисленное деление числа 11 на 5: 11=2·5+1, 2 = 0·5 + 2. Число 11 в пятиричной системе счисления записывается как 21. Перевод дробной части 0.87 осуществим с точностью до 5^{-2}: 0.87·5=4.35; 0.35 • 5 = 1.75. Следовательно, число 0.87 в пятиричной системе запишется как 0.41. Окончательно имеем $<11.87>_{10} = <21.41>_5$.

В естественных системах счисления, получивших наибольшее распространение, можно записывать числа (при натуральном основании) только одного знака. Для изображения чисел противоположного знака в таких системах используют специальный знак « + » или «—». Более целесообразно кодировать знак числа с помощью цифр, используемых при записи числа. Для этого в записи числа выделяют один специальный разряд, называемый *знаковым.* Цифра, стоящая в нем, не кодирует количественного эквивалента и не участвует в вычислении эквивалента с помощью функции *F*.

Будем кодировать знак « + » нулем, знак « — » — цифрой (S — 1) в нулевом разряде при задании чисел *x,* | *x* | < 1. Для этого определим прямой код числа *x,* |*x* | < 1, следующим образом:

$$[x]_{\text{п}} = \begin{cases} 0.\ x_1x_2\ldots x_n \text{ при } x \geqslant 0, \\ S-1.\ x_1x_2\ldots x_n \text{ при } x \leqslant 0. \end{cases}$$

Учитывая, что | *x* | < 1, это соотношение можно переписать в виде

$$[x]_{\text{п}} = \begin{cases} x \text{ при } x \geqslant 0, \\ S-1+|x| \text{ при } x \leqslant 0. \end{cases}$$

Широко используют *полулогарифмическую* форму представления чисел или представление с *плавающей точкой.*

Любое число *x* в системе счисления с натуральным основанием *S* можно представить (неоднозначно) как $S^{p(x)}m(x)$, где | *m* (*x*) | < 1. Будем называть *m(x) мантиссой* числа *x,* а *p(x) — порядком* числа *x* в системе с основанием *S.* Если при фиксированном *S* задать *p(x)* и *m(x),* то число *x* определяется однозначно. Пара *p(x)* и *m(x)* хранится в памяти ЭВМ. Для однозначного представления чисел в полулогарифмической форме обычно требуется, чтобы мантисса удовлетворяла неравенству $S^{-1} \leqslant m(x) < 1$. Такая мантисса называется *нормализованной.*

В машинном представлении числа в самом левом разряде кодируется знак порядка, затем в n_1 разрядах записывается величина порядка, далее имеется разряд для записи знака мантиссы и n_2 разрядов для записи величины мантиссы.

Рассмотрим реализацию операций сложения, вычитания, умножения и деления в позиционных арифметиках с естественным множеством цифр. При любом основании S операции сложения и умножения определяются таблицами выполнения этих операций в одном разряде и правилами образования переносов в старшие разряды. При этом необходимо, чтобы в операции участвовали соответствующие разряды слагаемых. Поэтому при сложении необходимо еще выровнять порядки слагаемых (сдвинуть мантиссы, так, чтобы суммировались разряды с одинаковыми номерами).

При алгебраическом сложении выравнивание порядков происходит в результате увеличения меньшего порядка до большего. Для того чтобы количественный эквивалент числа при этом не изменялся, необходимо каждое увеличение порядка на единицу компенсировать сдвигом мантиссы вправо на один разряд.

Полученному результату приписывается выровненный порядок. При нормализации мантиссы результата необходимо сдвигать ее вправо (на один разряд) или влево до тех пор, пока не будут выполняться условия нормализации. Для сохранения количественного эквивалента числа при этом необходимо увеличивать (при сдвиге мантиссы вправо) или уменьшать (при сдвиге мантиссы влево) порядок результата на число единиц, совпадающее с числом сдвигов.

При вычитании двух чисел возникают трудности, связанные с заемом единиц в старших разрядах. Эта операция плохо реализуется в ЭВМ для систем счисления с естественным множеством цифр и натуральным основанием. Для систем с отрицательным основанием или систем с натуральным основанием и симметрической (асимметрической) совокупностью цифр эта операция осуществляется просто: вычитаемое инвертируется (т. е. вместо записи числа x берется запись числа $-x$ в этой системе), и полученные коды суммируются. Для систем с натуральным основанием и естественным множеством цифр операции алгебраического сложения осуществляются с помощью *дополнительного* и *обратного кодов* этих чисел.

Заменим операцию вычитания $y - x$ операцией сложения y и $S - x$ с последующим уменьшением результата на величину, равную S:

$$y - x = y + (S - x) - S. \quad (2\text{ а})$$

Введем понятие *дополнительного кода* числа x:

$$[x]_д = \begin{cases} x, & x \geqslant 0, \\ S + x, & x < 0. \end{cases} \quad (2б)$$

Операцию вычитания можно заменить операцией сложения и на основании следующего соотношения:

$$y - x = y + (S - S^{-n} - x) - S + S^{-n}. \quad (2в)$$

Отсюда получаем определение *обратного кода* числа *x:*

$$[x]_о = \begin{cases} x \text{ при } x \geqslant 0, \\ S + x - S^{-n} \text{ при } x \leqslant 0. \end{cases} \quad (2г)$$

Из соотношений (2б) и (2г) получаем

$$[x]_д = [x]_о + S^{-n}, \ x < 0. \quad (2д)$$

Согласно формуле (2б), дополнительный код **отрицательного** числа $x \leftrightarrow x_1 x_2 \ldots x_n$, $|x| < 1$, имеет следующий вид:

$$[x]_д = (S - 1).\ (S - 1 - x_1)(S - 1 - x_2)\ldots(S - x_n).$$

Согласно соотношению (2д), обратный код этого числа имеет вид

$$[x]_о = (S - 1).\ (S - 1 - x_1)(S - 1 - x_2)\ldots(S - 1 - x_n).$$

Таким образом, имеем следующие *правила образования обратного и дополнительного кодов для отрицательных чисел.*

1. Для получения обратного кода отрицательного числа необходимо в каждом разряде S-ичной записи числа заменить цифру этого разряда на цифру, дополняющую ее до S — 1. В знаковом разряде следует записать цифру S — 1.

2. Для получения дополнительного кода отрицательного числа необходимо прибавить единицу к младшему разряду его обратного кода.

На основании формул (2а) и (2в) имеем соответственно следующие *правила алгебраического сложения.*

1. Для алгебраического сложения двух чисел x и y произвольного знака в системе счисления с натуральным основанием S и естественным множеством цифр достаточно перевести запись этих чисел в дополнительный код, просуммировать полученные коды по правилам сложения чисел в системе с основанием S и отбросить единицу переноса из знакового разряда, если такая возникнет. Полученный результат представляет собой дополнительный код алгебраической суммы чисел x и y.

2. Для алгебраического сложения чисел x *и* y произвольного знака в системе счисления с натуральным основанием S и естественным множеством цифр достаточно перевести запись этих чисел в обратный код, просуммировать полученные коды по правилам сложения чисел в системе с основанием S и добавить единицу в младший разряд полученного выражения, если при суммировании появляется единица

переноса из знакового разряда. Полученный результат представляет собой обратный код истинной (алгебраической) суммы чисел *x* и *y*.
При умножении необходимо просуммировать порядки сомножителей, произвести умножение мантисс по правилам умножения чисел, нормализовать результат и, если произошел сдвиг мантиссы произведения, изменить соответственно порядок произведения. При делении из порядка делимого вычитают порядок делителя. Деление мантисс заменяют вычитанием делителя из делимого до тех пор, пока в результате очередного вычитания не будет получена отрицательная разность. Затем прибавляют делитель к отрицательной разности (эта операция называется восстановлением остатка) и в качестве цифры частного записывают число вычитаний делителя без учета последнего вычитания. После этого роль делимого начинает играть остаток, а роль делителя — прежний делитель, сдвинутый на один разряд вправо, и т. д. Мантиссу частного нормализуют и соответственно изменяют порядок частного.
Как уже отмечалось, система счисления является весомозначной, если для каждого разряда с номером *j* можно указать такое число p_j, что количественный эквивалент, который сопоставляется цифре a_i, записанной в этом разряде, равен $p_j \cdot (a_i)$, где (a_i) — количественный эквивалент, сопоставляемый этой же цифре в нулевом разряде.
Аналогом соотношения (1) является следующее выражение:

$$x = \sum_{j=-\infty}^{\infty} (a_i) \cdot p_j.$$

В вычислительной технике представляют интерес *двоично-десятичные* системы счисления, в которых каждая десятичная цифра кодируется четырьмя двоичными цифрами и, следовательно, каждый разряд десятичной записи замещается четырьмя разрядами. Если этим четырем разрядам соответствуют некоторые веса, то имеет место весомозначная система для этих четырех двоичных разрядов.
При использовании двоично-десятичных систем в ЭВМ желательно, чтобы кодирование (оно неоднозначно, так как для кодирования десяти цифр можно использовать любые из 16 тетрад-четверок из нулей и единиц) удовлетворяло некоторым ограничениям. Приведем пять основных требований, которые сформулированы Рутисхаузером.
1. *Единственность.* Необходимо однозначное соответствие цифр и тетрад. Если это требование не выполнено, то невозможно кодирование и декодирование чисел. Другими словами, различные десятичные цифры должны кодироваться различными тетрадами.
2. Упорядоченность. Большим десятичным цифрам должны соответствовать большие (по количественному эквиваленту)

тетрады. Выполнение этого требования необходимо при сравнении кодированных чисел.

3. *Четность.* Четным десятичным цифрам должны соответствовать четные тетрады (тетрады, у которых в крайнем правом разряде стоит нуль), а нечетным цифрам — нечетные тетрады.

4. *Дополнительность.* Если цифры десятичной системы таковы, что сумма их равна девяти, то им должны сопоставляться тетрады, которые взаимно инвертированы, т. е. получаются друг из друга заменой единиц на нули, а нулей на единицы. Выполнение этого требования необходимо для того, чтобы ввести в двоично-десятичной системе дополнительный или обратный код.

5. *Весомозначность.* Должны существовать четыре веса p_1, p_2, p_3 и p_4 такие, что если десятичной цифре x сопоставлена тетрада $\alpha_1\alpha_2\alpha_3\alpha_4$, то имеет место равенство $x = \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 + \alpha_4 p_4$.

Кодирование, удовлетворяющее всем пяти требованиям, называется *совершенным.*

Условимся через $q(x)$ обозначать тетраду, сопоставляемую десятичной цифре x. Рассмотрим правила сложения для чисел, записанных в двоично-десятичной системе. Пусть в десятичной системе взяты цифры x и y. Тогда $x + y$ — либо некоторая новая цифра этой системы (если $x + y < 10$), либо результатом суммирования этих цифр является цифра, соответствующая $x + y$ — 10, и возникнет единица переноса в следующий разряд. Тогда правила сложения в двоично-десятичной системе имеют вид

$$q(x) + q(y) = \begin{cases} q(x+y) \text{ при } x + y < 10, \\ q(x+y-10) + 16 \text{ при } x + y \geqslant 10. \end{cases} \quad (3)$$

Здесь прибавление 16 соответствует переносу единицы в следующий разряд (перенос в правый разряд тетрады, стоящей слева от данной). Как следует из соотношения (3), при суммировании чисел в такой системе необходимо вносить поправки.

Пусть, например, кодируется каждая десятичная цифра ее записью в двоичной системе с использованием четырех двоичных разрядов. В этом случае цифра 5 будет закодирована тетрадой 0101. Такой способ кодирования называется *кодом прямого замещения* (8421). В данном случае соотношение (3), учитывая, что для кода прямого замещения $q(x) = x$, можно записать в виде

$$q(x) + q(y) = \begin{cases} x + y \text{ при } x + y < 10, \\ x + y + 6 \text{ при } x + y \geqslant 10. \end{cases}$$

Таким образом, при суммировании в коде прямого замещения необходимо произвести суммирование в каждой паре тетрад по

правилам двоичного сложения, учитывая перенос между тетрадами, если последний возникает. После этого в те разряды, где сумма кодированных цифр превысила 10, следует добавить поправку 0110. Проиллюстрируем это на следующем примере. Пусть требуется в коде прямого замещения найти сумму чисел 205 и 768. Выполним необходимые операции:

```
  205 – 0010 0000 0101 }
+                      } кодирование,
  768 – 0111 0110 1000 }
-------------------------
        1001 0110 1101   первое суммирование
      + 0000 0000 0110   поправки,
-------------------------
973 – 1001 0111 0011   результат.
```

Код прямого замещения удовлетворяет всем требованиям Рутисхаузера, кроме четвертого. Нарушение этого требования не позволяет вводить дополнительный или обратный код, что, в свою очередь, не позволяет заменить вычитание операцией сложения. Для выполнения свойства дополнительности будем кодировать десятичную цифру x тетрадой $q(x)$, равной $x + 3$. Полученный код называется *кодом с избытком три* или *кодом Штибитца.* В этом случае соотношение (3) принимает следующий вид:

$$q(x) + q(y) = \begin{cases} x + y + 3 \text{ при } x + y < 10, \\ (x + y - 10) + 3 + 16 \text{ при } x + y \geqslant 10. \end{cases}$$

Таким образом, в коде с избытком три требуется поправка + 3(0011), если x + y≥10, и поправка -3(-0011: 1100 в обратном и 1101 в дополнительном коде), если $x + y < 10$. Приведем пример суммирования чисел -471 и 607 с использованием дополнительного кода:

```
-471: 1000 0101 1100 }
                     } кодирование,
+607: 1001 0011 1010 }
-------------------------
      10001 1001 0110   первое суммирование
  +    0001 1001 0110   отбрасывание переноса из левого разряда,
       0011 1101 0011   поправки,
-------------------------
136 – 0100 0110 1001   получение результата после поправок.
```

Кодирование с избытком три не обладает свойством весомозначимости. Единственным кодированием, которое обладает всеми пятью свойствами, является *кодирование Айкена - Эмери.* Это кодирование с весами 2 4 2 1.

В качестве упражнения доказать, что совершенным является кодирование с весами

0 – 0 0 0 0, 1 – 0 0 0 1, 2 – 0 0 1 0, 3 – 0 0 1 1, 4 – 0 1 0 0, 5 – 1 0 1 1, 6 – 1 1 0 0, 7 – 1 1 0 1, 8 – 1 1 1 0, 9 – 1 1 1 1.

Анализируя коды, замечаем, что

$$q(x) = \begin{cases} x & \text{при } x < 5, \\ x + 6 & \text{при } x \geqslant 5; \end{cases}$$

для определения правил суммирования тетрад в этом коде необходимо рассмотреть следующие случаи:

1. $x < 5,\ y < 5,\ x + y < 5,$
   $q(x) + q(y) = x + y;\ q(x + y) = x + y;\ \Delta = 0.$

2. $x < 5,\ y < 5,\ 5 \leqslant x + y < 10,$
   $q(x) + q(y) = x + y;\ q(x + y) = x + y + 6;\ \Delta = 6.$

3. $x < 5,\ y \geqslant 5,\ 5 \leqslant x + y < 10,$
   $q(x) + q(y) = x + (y + 6),\ q(x + y) = x + y + 6;\ \Delta = 0.$

4. $x < 5,\ y \geqslant 5,\ 10 \leqslant x + y < 15,$
   $q(x) + q(y) = x + (y + 6),\ q(x + y) + 6$ (перенос, см. (4.10)) $= x + y + 6,\ \Delta = 0.$

5. $x \geqslant 5,\ y \geqslant 5,\ 10 \leqslant x + y < 15,$
   $q(x) + q(y) = (x + 6) + (y + 6),\ q(x + y) + 6 =$
   $= (x + y) + 6;\ \Delta = -6.$

6. $x \geqslant 5,\ y \geqslant 5,\ x + y \geqslant 15,$
   $q(x) + q(y) = (x + 6) + (y + 6),\ q(x + y) + 6 =$
   $= (x + y + 6) + 6;\ \Delta = 0.$

Таким образом, при суммировании $x + y$ в коде Айкена — Эмери необходима поправка +6 в тетрадах, где $x < 5,\ y < 5,\ 5 \leqslant x + y < 10,$ и поправка —6, когда $x \geqslant 5,\ y \geqslant 5,\ 10 \leqslant x + y < 15.$

Просуммируем в этом коде рассмотренную выше пару чисел — 471 и 607, используя при этом дополнительный код. Имеем

−471	:	1011	0010	1111	кодирование,
+607	:	1100	0000	1101	
		0111	0011	1100	суммирование,
		0111	0011	1100	отбрасывание переноса из левого разряда,
		1010	0000	0000	поправка,
+		5-й случай	1-й случай	8-й случай	
136	:	0001	0011	1100.	

Системой счисления, позволяющей производить вычисления в каждом разряде независимо от результатов, полученных в других разрядах, является код в остатках. *Множеством модулей* кода в остатках называется совокупность l взаимно простых натуральных чисел: $q_1, q_2, \ldots, q_l$.

Обозначим через $\mathrm{res}\,\frac{x}{q_i}$ остаток от деления числа x на q_i. *Кодом*

в остатках числа *x* по множеству модулей $q_1, q_2, \ldots, q_l$ называется выражение вида $\text{res}\,\frac{x}{q_1}\,\text{res}\,\frac{x}{q_2}\ldots\text{res}\,\frac{x}{q_l}$. Если

$$\text{res}\,\frac{x}{q_i}=\alpha \text{ и } \text{res}\,\frac{y}{q_i}=\beta,$$

то $x = nq_i + \alpha$ и $y = mq_i + \beta$. Просуммируем эти два равенства. В результате получаем $x + y = (n + m)q_i + (\alpha + \beta)$. Теперь перемножим соответственно правые и левые части исходных равенств: $xy = (mnq_i + n\beta + m\alpha)q_i + \alpha\beta$. Если $\alpha + \beta$ и $\alpha\beta$ меньше q_i, то можно утверждать, что при суммировании и умножении двух чисел их остатки от деления на q_i также складываются или перемножаются. Если же $\alpha + \beta$ или $\alpha\beta$ больше q_i, то, разделив их на q_i и определив целое частное, получим, что утверждение о суммировании и перемножении остатков совпадает с ранее высказанным, если считать, что после выполнения этих операций происходит деление на модуль данного разряда и выделение истинного остатка. Подобная операция может иметь место для каждого q_i независимо, поэтому код в остатках позволяет суммировать и перемножать поразрядно, что позволяет увеличить скорость выполнения операций ЭВМ.

Пусть, например, имеется система из трех модулей: 7, 8, 9. Тогда коды в остатках для чисел $x = 11$ и $y = 6$ имеют соответственно виды 432 и 666. Просуммируем и перемножим поразрядно коды *x* и *y*; тогда получаем 10 9 8 и 24 18 12. После деления в каждом разряде результата на модуль этого разряда и выделения истинного остатка получим соответственно 318 и 323. Сумма и произведение *x* и *y,* равные 17 и 66, имеют коды в остатках 318 и 323.

Для получения взаимно однозначного представления чисел в коде в остатках следует учитывать, что если произведение всех модулей q_i, используемых для кодирования, есть *N*, то взаимно однозначно в коде в остатках с этими модулями можно кодировать лишь *N* различных чисел (от 0 до *N* - 1, от *N* до 2N - 1 и т. д.). Число *N* называется *мощностью системы модулей.*

7.3. Алгоритмический этап проектирования

Проектирование автоматного оператора заключается в построении автоматного отображения по заданному словесному описанию, включающему цель проектирования, назначение синтезируемого автомата и свойства поведения (функционирования) управляемого объекта, в контуре с которым проектируемый автомат реализует поставленную цель. Автоматное отображение, как правило, задают в виде графа

переходов. *Граф переходов* - это граф $G = \langle V, (X, Y) \rangle$, каждая вершина которого взаимно однозначно соответствует внутреннему состоянию автомата, и если из состояния S, автомат переходит в состояние S_j, то соответствующие им вершины v_i, v_j соединяются дугой $(v_i, v_j) \in U$ взвешенной парой векторов *(X, Y),* при которых этот переход осуществляется.

На алгоритмическом этапе неформально заданная информация преобразуется в формальную систему в виде автоматного оператора. При этом используется принцип аналогий. Формально этот переход может быть основан на применении грамматик, в которых правила подстановки формализуют данные свойства управляемого объекта. В результате применения грамматик порождаются композиции графов переходов, удовлетворяющие заданному словесному описанию. Эта композиция часто представляет собой двухуровневую иерархию. Первый уровень представляет собой графы переходов, реакцией которых является непосредственно управляющее воздействие на управляемый объект. Второй уровень представляет собой граф переходов, реакция которого соответствует возбуждению инициальных вершин графов первого уровня.

Рассмотрим построение графов переходов автоматов, реализующих устройства вычислительной техники и промышленной автоматики.

Пусть задан операционный автомат в виде арифметического устройства последовательного действия, изображенного на рис. 8.

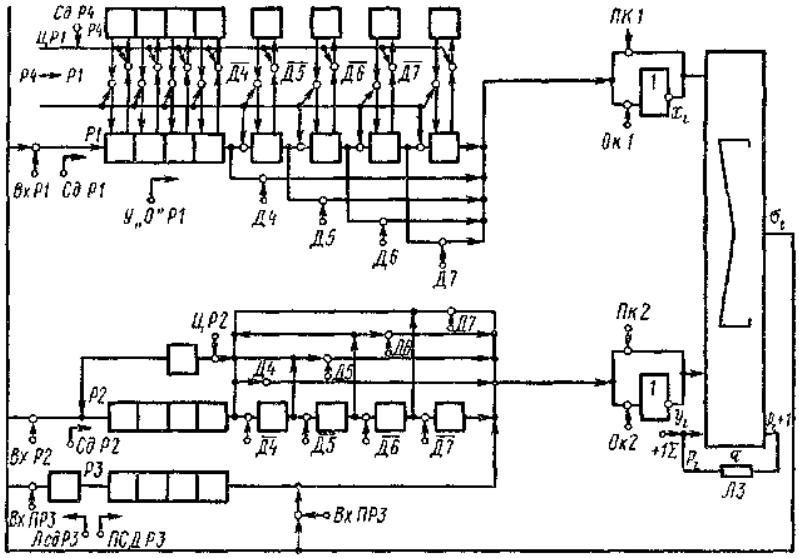


Рис. 8

Это устройство содержит четыре регистра ***Р1, Р2, Р3, Р4*** и одноразрядный сумматор комбинационного типа, которые связаны между собой соединительными каналами и управляемыми вентилями. Четырехразрядные числа представляются в устройстве в двоичном коде 8421.

Синтезируем автомат управляющий вычислением мантиссы частного. До выполнения операции делимое *a* находится в регистре *Р1,* делитель *b* -в регистре *Р2,* числа *a* и *b* нормализованы, а частное формируется в регистре *Р3.*

При выполнении операции деления вычитание заменяем сложением в дополнительном коде и используем следующий алгоритм.

1. Вычитаем в дополнительном коде содержимое регистра *Р2(b)* из содержимого регистра *Р1 (a),* одновременно переписывая содержимое регистра *Р1* в регистр *Р4.*
2. Если $a - b \geq 0$, то выполняем п. 3, в противном случае — п. 4.
3. Разность $a - b$ записываем в регистр *Р1,* регистр *Р4* устанавливаем в «нуль», в регистр *Р3* записываем 1, сдвигаем на один разряд вправо делитель, регистр результата влево и переходим к п. 5.
4. Записываем содержимое регистра *Р4* в регистр *Р1,* в регистр *Р3* записываем 0, сдвигаем на один разряд делитель вправо, регистр результата — влево и переходим к п. 5.

5. Если п. 1 был выполнен менее пяти раз, то переходим к п. 1, в противном случае — к п. 6.
6. Конец (в общем случае здесь осуществляется передача управления в блок нормализации частного).

Для простоты считаем, что в регистр *Р3* предварительно записан код нуля.

Зная операционный автомат и реализуемый алгоритм, составляем временную диаграмму функционирования управляющего автомата, которая представляет собой алгоритм выполнения данной операции в терминах управляющих точек — *микроопераций.* Каждой микрооперации соответствует выходной канал управляющего автомата. Отсюда число выходных каналов синтезируемого автомата равно числу всех микроопераций.

Множество микроопераций рассматриваемого устройства таково: *СдР1* — сдвиг регистра *Р1; СдР2* — сдвиг регистра *Р2; ПСдП3* — правый сдвиг регистра *Р3; ЛСдР3* — левый сдвиг регистра Р3; *СдР4* - сдвиг регистра *Р4* (наличие регистра *Р4* позволяет не производить восстановления остатка); *ПК1* — прямой код содержимого регистра *Р1; ПК2* — прямой код содержимого регистра *Р2; ОК1* — обратный код содержимого регистра *Р1*; *ОК2* — обратный код содержимого регистра *Р2; ВхР1*— вход регистра *Р1; ВхР2* — вход регистра *Р2; У«О»Р1* - установка регистра *Р1* в «нуль»; *ВхЛР3* - вход регистра Р3 слева; *ВхПР3* — вход регистра *Р3* справа; *ЦР1* — цикл регистра *Р1*; *ЦР2*— цикл регистра Р2; +1Σ — подача единицы в цепь переноса сумматора; *Д4* — работа с векторами длины 4; *Д5* — работа с векторами длины 5; *Д6* — работа с векторами длины 6; Д7 — работа с векторами длины 7; *Р4 →Р1* — передача содержимого регистра *Р4* в регистр *Р1*

Строка временной диаграммы взаимно однозначно соответствует микрооперации.

Множество микроопераций, выполняемых одновременно, называется микрокомандой, а множество последовательностей микрокоманд, соответствующее выполняемой операции, — *микропрограммой.* Каждая микропрограмма взаимно однозначно соответствует значению входного вектора *X*. Следовательно, число различных значений входного вектора *X* равно числу выполняемых операций, а число входных каналов равно $[\log_2 | \{X_i\} |]$, где [ ] — знак ближайшего целого числа.

При синтезе управляющего автомата, реализующего заданный алгоритм, предварительно составляют временные диаграммы, детализирующие каждый пункт алгоритма. Временная диаграмма, соответствующая вычислению нулевого разряда частного на

операционном автомате (рис. 8) по приведенному алгоритму деления, приведена на рис. 9.

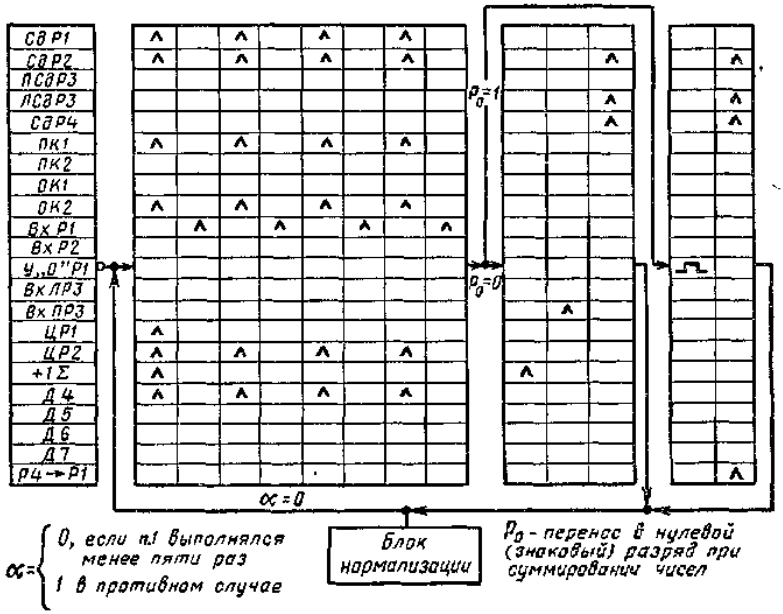


Рис. 9

Временные диаграммы, соответствующие вычислениям последующих разрядов частного, отличаются от диаграмм, изображенных на рис. 9, только в первом блоке, а именно: при вычислении *i*-го разряда частного вместо четырехкратного выполнения микрооперации *Д4* имеет место соответственно (4 — *i*)-кратное выполнение микрооперации Д (4 — *i*), т. е при вычислении первого разряда выполняется пятикратное возбуждение *Д5* и т. д. Следовательно, чтобы использовать первый временной блок, надо перед вычислением соответствующего разряда частного производить его «настройку». Настройка первого временного блока заключается в следующем: $Д_j$ рассматривают как входной параметр этого блока и перед вычислением нулевого разряда *j* присваивают значение 4, а после вычисления каждого разряда частного значение *j* увеличивают на 1 Осуществим настройку первого временного блока введением дополнительного четырехразрядного регистра *P5,* в который перед началом выполнения операции деления записывается 1, а после вычисления каждого разряда частного производится сдвиг регистра *P5* на один разряд. Выход *i*-го разряда регистра *P5* подается на клапан

$Д(|i| \pm 3)$.

Для нахождения числа вычисленных разрядов частного введем трехразрядный счетчик *(СчЦ),* состояние 101 которого укажет на то, что вычислено пять разрядов частного и надо передавать управление в блок нормализации.

Основываясь на проведенном анализе, целесообразно изменить множество микроопераций, а именно: вместо микроопераций *Д4, Д5. Д6, Д7* ввести микрооперации *СдР5* (сдвиг регистра *Р5)* и *1вР5* (запись единицы в регистр *Р5)* и дополнительно ввести микрооперацию *+1СчЦ* (прибавление единицы в счетчик циклов СчЦ) и *СчЦв«0»* (установление счетчика циклов в «нуль»).

В результате рассмотренных преобразований временная диаграмма, приведенная на рис. 9, преобразуется во временную диаграмму, изображенную на рис. 10.

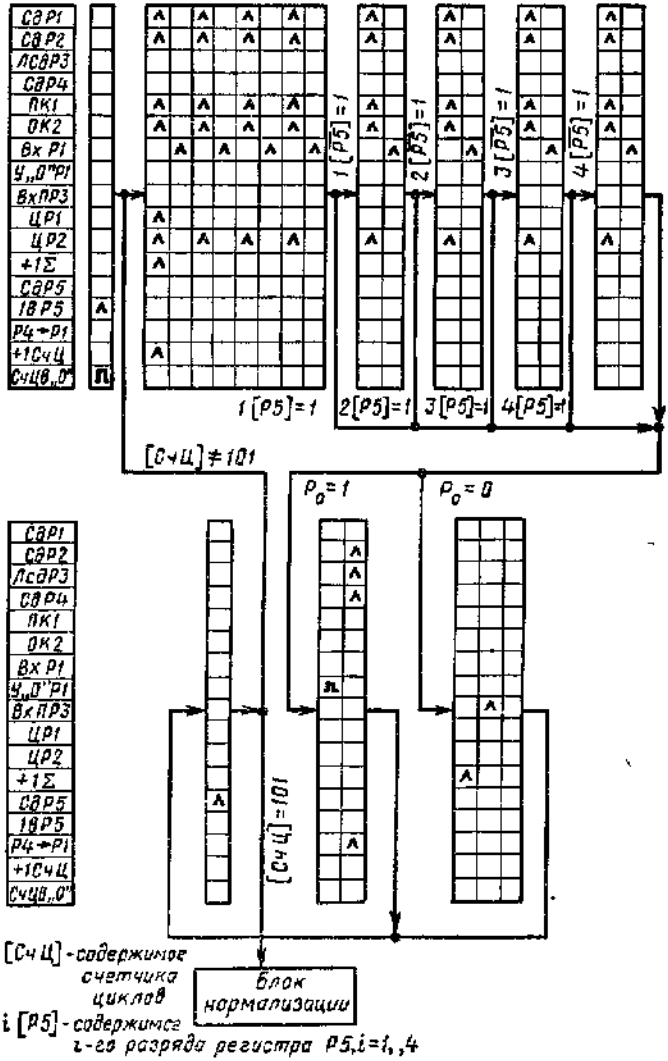


Рис. 10

Временна́я диаграмма еще не дает автоматного отображения, Действительно, одному значению входного вектора, определяющего выполняемую операцию, соответствуют несколько значений выходного вектора (микрокоманд), т. е. временна́я диаграмма задает неоднозначное преобразование входного вектора в выходной. Следовательно, это преобразование не является автоматным

отображением. **Для получения автоматного отображения необходимо ввести в синтезируемый автомат <u>память</u>**. При этом заведомо достаточен объем памяти, равный числу всех микрокоманд. Совокупность входного вектора и состояния памяти однозначно определяет микрокоманду.

В соответствии с временной диаграммой построим граф переходов автомата. Внутреннее состояние автомата взаимно однозначно сопоставляется столбцу во временной диаграмме.

Граф переходов автомата, реализующий временную диаграмму (рас. 10), изображен на рис. 11.

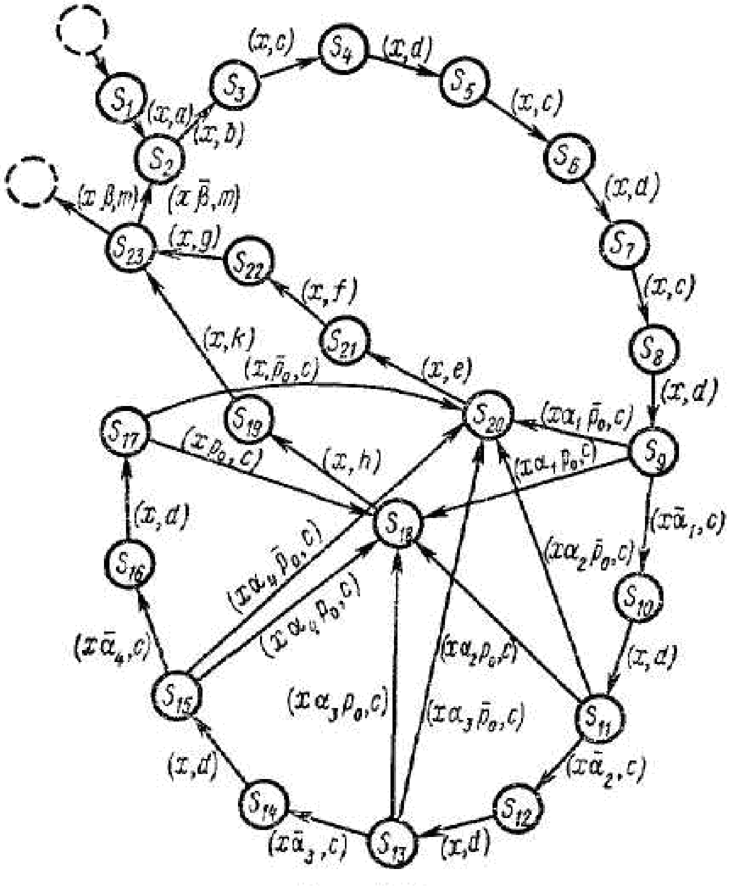


Рис. 11.

Микрокоманды на этом рисунке обозначены латинскими буквами и имеют следующий вид:

$a = \{1вР5,\ СчЦв«О»\};$

$b = \{СдР1,\ СдР2,\ ПК1,\ ОК2,\ ЦР1,\ ЦР2,\ +1\sum,\ +1СчЦ\};$

$c = \{ВхР1\};\ d = \{СдР1,\ СдР2,\ ПК1,\ ОК2,\ ЦР2\};$

$e = \{+1\sum\};\ f = \{ВхПР3\};\ g = \{СдР2,\ ЛСдР3,\ СдР4\};$

$h = \{У«О»вР1\};\ k = \{СдР2,\ ЛСдР3,\ СдР4,\ Р4 \to Р1\},\ m = \{СдР5\}.$

Считаем, что если на вход автомата управления поступила единица ($x = 1$), то автомат выполняет управление делением.

Логические переменные (признаки), определяющие ход управления, обозначим как α_i — содержимое *i*-го разряда регистра *Р5*; β - условное состояние счетчика *СчЦ;*

$$\beta = \begin{cases} 1, \text{ если состояние счетчика есть 101,} \\ 0 \text{ в противном случае;} \end{cases}$$

P_0 — перенос в нулевой разряд при суммировании чисел.

Рассмотрим большую систему промышленной автоматики - систему сжигания твердого топлива в плотном слое. Система содержит четыре полугазовые топки, установки подачи топлива, воздуха и установку удаления шлака. Схема полугазовой топки изображена на рис. 4.12, где *1* — конвейер, 2 — плужок, *3* — входной бункер, *4* — топливо, 5 — питатель, б — забрасыватель, 7 — полугаз, 8 — подача воздуха, 9 — рабочая камера, *10* — шлак, *11* — слой свежезаброшенного топлива. *12* — зона восстановления, *13* — зона горения, *14* — зона шлака, *15* — колосниковая решетка.

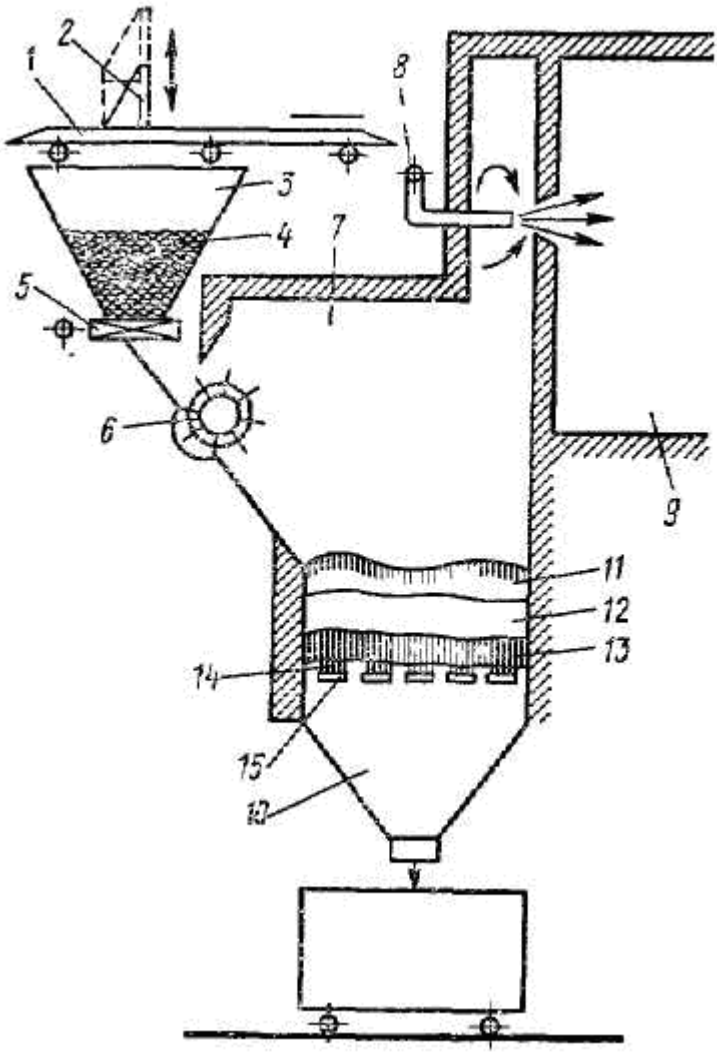
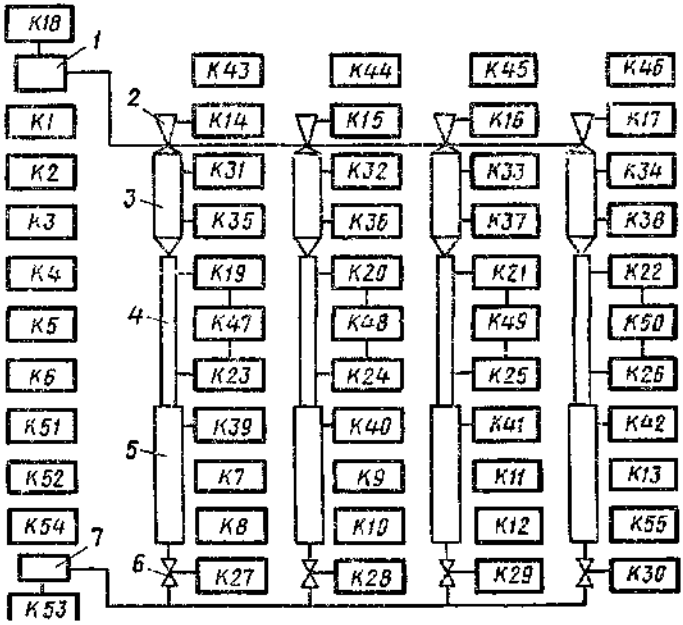


Рис. 12.

Полугазовые топки, являющиеся одним из видов промышленных печей, входят в состав газового хозяйства завода, и автоматизация управления процессами сжигания в топках — одна из задач автоматизации всего технологическою процесса. Полнота газификации определяется процентным содержанием в полугазе двуоксида углерода. С его повышением падает теплота сгорания полугаза и

увеличивается температура в топке. Полугаз перед выходом из топки смешивается с вторичным воздухом и подается в рабочую камеру. Топка снабжается топливом через систему, состоящую из конвейера, плужковых сбрасывателей, входного бункера, питателя и забрасывателя. Шлак удаляется с помощью встряхивателей, спускового клапана и вагонов. Воздух, предназначенный для сжигания топлива, подается с помощью клапана подачи первичного воздуха. Управление процессом газификация осуществляется выдерживанием определенной толщины слоя сгораемого топлива (200 — 400 мм для антрацита, 400 — 800 мм для каменных и бурых углей), т. е. определяется интенсивностью подачи топлива, первичного воздуха и удаления шлака из зоны горения

Система управления сжиганием имеет следующие каналы (рис. 13), представляющие собой микрооперации при проектировании управляющего автомата: *К1* — пуск и останов процесса газификации; *К2* — индикация работы процесса газификации; *К3* — пуск и останов заполнения входных бункеров; *К4* — индикации заполнения входных бункеров; *К5* — пуск и останов подачи



Рнс. 13

топлива в топки; *К6* — индикация подачи топлива в топки; *К7* — приостанов газификации в первой топке; *К8* — индикация состояния

канала *К7; К9* — приостанов газификации во второй топке; *К10* — индикация состояния канала *К9; К11* — приостанов газификации в третьей топке; *К12* — индикация состояния канала *К11; К13* - приостанов газификации в четвертой топке; *К14* — индикация состояния канала *К13;* К15 — плужок первой топки; К16 —плужок второй топки; *К17* — плужок третьей топки; *К18* — плужок четвертой топки; *К19* — конвейер; *К20 — К23* — питатели первой,...,четвертой топки соответственно ;К24—К27 — забрасыватели; *К28 — К31* — клапаны подачи вторичного воздуха; *К32—К35* — датчики верхнего уровня во входных бункерах; *К40 — К43* — датчики подачи топлива; *К44 -К47* - индикация аварии установки заполнения входных бункеров; *К48 — К51* — индикация аварии установки подачи топлива в печи; *К52* — пуск и останов процесса управления; *К53* — индикация управления; *К54* - начальные установки исполнительных элементов.

На рис. 13: *1* - конвейер, *2* — плужок, *3* — входной бункер, *4* — система подачи топлива, 5 — топка, *6* — клапан подачи воздуха, 7 — общий клапан.

Перечисленное множество каналов включается в носитель модели Ψ_a, формализующей функционирование управляемого объекта. Кроме этого множества носитель модели Ψ_a может включать множество дополнительных элементов, являющихся идентификаторами входных, внутренних и выходных каналов, которые введены для получения эффективного автоматного управления. Исходя из информации, полученной от технолога, в носителе устанавливаются причинно-следственные связи вида $A \rightarrow B$, которые определяют сигнатуру модели Ψ_a. Очевидно, что модель обладает свойством симметричности и ее можно задать в виде графа $G^M(\Psi_a)$. Причинно-следственные связи могут быть заданы и в виде временных диаграмм. Переход от графа или временных диаграмм к графу переходов аналогичен переходу от временных диаграмм (которые могут быть заданы в виде графа) к графу переходов устройства микропрограммного управления.

Таким образом, на этом этапе проектирования автоматов осуществляется переход от отображения $A \rightarrow$ B к автоматному отображению $XS^+ \rightarrow S^- Y$, где X — входной вектор, Y — выходной вектор, или реакция автомата, S^+ — идентификатор внутреннего состояния, в котором автомат находится в рассматриваемый момент, S^- — идентификатор внутреннего состояния, в которое автомат переходит из состояния S^+ под действием входного вектора X.

7.4. Абстрактное проектирование автоматов

Введенный на этапе алгоритмического проектирования объем памяти $\{S_i\}$ автомата может быть избыточным. Эта **избыточность устраняется склеиванием эквивалентных состояний**. Данное преобразование относится к этапу абстрактного проектирования.
Эквивалентными **называются состояния, находясь в которых автомат вырабатывает для любой входной последовательности одну и ту же выходную последовательность.**
Из определения эквивалентных состояний следует, что если каждый класс эквивалентных состояний заменить одним: состоянием, склеивая соответствующие вершины, то полученный граф переходов будет задавать то же отображение $X \rightarrow Y$, что и первоначальный.
Метод *абстрактной минимизации автомата,* основанный на склеивании эквивалентных состояний, был предложен Хафменом и состоит в последовательном выделении классов эквивалентных состояний с помощью таблиц выходов и переходов. Рассмотрим метод Хафмена на примере.
Пример 1. Пусть в результате проведения алгоритмического этапа синтеза был получен граф переходов, изображенный на рис. 14, *а*.

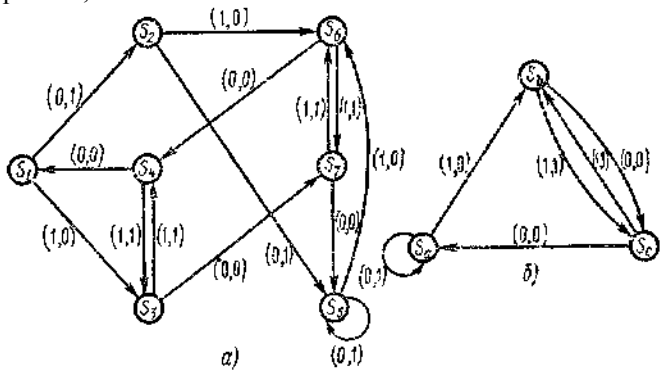


Рис. 14.

Строим *таблицу выходов,* представляющую собой двумерную таблицу (табл. 3), каждой строке которой взаимно однозначно соответствует значение входного вектора *X,* столбцу - внутреннее состояние автомата и на пересечении *i*-й строки с *j*-м столбцом находится значение выходного вектора Y, который вырабатывается на выходе, когда автомат находится в *j*-м внутреннем состоянии и на входе подан *i*-й вектор.

Таблица 3

X_i	S_j						
	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	1	1	0	0	1	0	0
1	0	0	1	1	0	1	1

Если двум внутренним состояниям автомата соответствуют различные значения столбцов в таблице выходов, то состояния не эквивалентны, так как в этих состояниях отображения $X \to Y$ различные.

Предварительно разобьем все множество состояний на классы *условно эквивалентных состояний,* а именно: в один и тот же класс попадают те состояния, которым соответствуют одинаковые значения столбцов в таблице выходов: $K_1 = \{S_1, S_2, S_5\}; K_2 = \{S_3, S_4, S_6, S_7\}$.

Если таблицу выходов рассматривать как матрицу инцидентности некоторой модели, то внутренние состояния попадают в один и тот же класс, если в соответствующей ей частотной матрице отношений собственные и взаимные частоты внутренних состояний равны друг другу, т. е.

$$\frac{\partial G^M}{\partial S}(S_a, S_b) = \frac{f_{aa} - 2f_{ab} + f_{bb}}{f_{ab}} = 0.$$

Для того чтобы внутренние состояния автомата были эквивалентными, одного и того же соответствия $X \to Y$ только в этих состояниях недостаточно, нужно, чтобы и при любом другом возможном переходе из этих состояний отображение $X \to Y$ было одним и тем же.

Для проверки этого условия строим *таблицу переходов* (табл. 4). Каждой строке, столбцу в этой таблице соответствуют те же значения, что в таблице выходов, и на пересечении i-й строки с j-м столбцом находится класс условно эквивалентных состояний, в который переходит автомат из состояния S_j под воздействием X_i.

Таблица 4

X_i	S_j						
	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	K_1	K_1	K_2	K_1	K_1	K_2	K_1
1	K_2	K_2	K_2	K_2	K_2	K_2	K_2

Если класс условно эквивалентных состояний, впделенный на предыдущем шаге, не является классом эквивалентных состояний, то состояниям этого класса соответствуют различные значения столбцов; это означает, что при дальнейших переходах отображения $X \to Y$ различны для данных состояний.

Каждый класс K_i разбивается на новые классы условно эквивалентных состояний, причем в один и тот же класс входят все состояния класса

K_i с одинаковыми значениями столбцов:
$K_1 = \{S_1,\ S_2,\ S_5\}, K_2' = \{S_3,\ S_6\}, K_2'' = \{S_4,\ S_7\}.$

Составим по таблице переходов частотную матрицу отношений, определив при этом умножение как $K_i \times K_i = 1,\ K_i \times K_j = 0\ (i \neq j)$. Тогда внутренние состояния попадают в один и тот же класс при разбиении классов, полученных на предыдущем шаге, если собственные и взаимные частоты этих состояний равны друг другу, т. е.

$$\frac{\partial G^M}{\partial S}(S_a,\ S_b) = 0.$$

После получения классов $K_1,\ K_2',\ K_2''$ спять строим таблицу переходов и т. д. до тех пор, пока каждый класс условно эквивалентных состояний, выделенный на предыдущем шаге, не станет неизменным.

Построим таблицу переходов (табл. 5), учитывая разбиение класса K_2.

Таблица 5

X_t	S_j						
	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	K_1	K_1	K_2''	K_1	K_1	K_2''	K_1
1	K_2'	K_2'	K_2''	K_2'	K_2'	K_2''	K_2'

Анализируя эту таблицу, делаем вывод, что все состояния каждого из выделенных классов «ведут себя дружно»: переходят под воздействием X в один и тот же класс, что обусловливает получение на выходе одного и того же значения Y для всех состояний класса при одном переходе. Следовательно, все состояния одного класса ведут себя как одно состояние, которым его и заменяют.

Заменим классы $K_1,\ K_2',\ K_2''$ соответственно внутренними состояниями $S_a,\ S_b,\ S_c$. В результате получим минимизированный граф переходов (рис. 14,б), задающий то же отображение $X \rightarrow Y$, что и первоначальной граф переходов (рис. 14, *а)*. Для иллюстрации проведем три эксперимента.

Пусть на вход автомата подается временная последовательность вида $X(t) = 011010$. В первом и во втором случаях автомат задан первоначальным трафом переходов (рис. 14, а) соответственно с начальными состояниями S_1 и S_2. В третьем случае автомат задан минимизированным графом (рис. 14,б) с начальным состоянием S_a. Определим временные выходные последовательности $Y(t)$ для каждого случая.

Результаты всех трех экспериментов сведем в таблицу (табл. 6).

Таблица 6

t		t_1	t_2	t_3	t_4	t_5	t_6
$X(t)$		0	1	1	0	1	0
$S_n = S_1$	$S(t+1)$	S_2	S_6	S_7	S_5	S_6	S_4
	$Y(t)$	1	0	1	0	0	0
$S_n = S_2$	$S(t+1)$	S_5	S_6	S_7	S_5	S_6	S_4
	$Y(t)$	1	0	1	0	0	0
$S_n = S_a$	$S(t+1)$	S_a	S_b	S_c	S_a	S_b	S_c
	$Y(t)$	1	0	1	0	0	0

Если размер таблицы выходов и переходов велик в случае, когда необходимо проводить минимизацию графа переходов с помощью ЭВМ, то целесообразно использовать частотные матрицы отношений, соответствующие этим таблицам, что упрощает вычисления. При этом минимизацию графов переходов проводят, основываясь на следующем утверждении.

Внутренние состояния S_a, S_b эквивалентны, если и только если производные $\frac{\partial G}{\partial S}(S_a, S_b)$ от модельных графов, соответствующих таблицам выходов и переходов, равны нулю

$$\frac{\partial G}{\partial S}(S_a, S_b) = \frac{f_{aa} - 2f_{ab} + f_{bb}}{f_{ab}} = 0$$

на каждом шаге разбиения множества внутренних состояний на классы.

Минимизация объема памяти, например в четыре раза, экономит только два элемента памяти. Более актуальным и мощным средством оптимизации всей структуры автомата является *абстрактная декомпозиция автоматов,* особенно *параллельная декомпозиция автоматов.* Необходимость поиска ее, с одной стороны, диктуется требованиями, предъявляемыми к быстродействию управления (обусловливающими использование параллельного управления объектом). С другой стороны, **повышение надежности автоматного управления требует разбиения автомата на ряд функционально не связанных друг с другом автоматов меньшей размерности, обеспечивающих реализацию полученного автоматного оператора**.

Поиск параллельной декомпозиции абстрактных автоматов сводится к *разложению* графа переходов в *частичное декартово произведение* более простых по числу вершин графов. Разложить граф G в частичное декартово произведение - значит найти графы G_i, $i = 1, \ldots, n$, такие, что $G \subset G_1 \times G_2 \times \ldots \times G_n$.

Для важного в вычислительной технике класса автоматов *микропрограммного управления* задачу построения параллельной декомпозиции можно решить, используя характерные свойства автоматов этого класса.

При рассмотрении *микропрограммных* автоматов элементы, с которых снимается информация, характеризующая ход вычислений, будем относить к цепям обратной связи, включенным в блок памяти автомата. **Микропрограммные автоматы обладают специфическим свойством: входной вектор *X* остается неизменным от начала до конца работы автомата, пока не выполнится заданная операция.** Используя это свойство, уточним понятие микропрограммного автомата.

Предварительно введем несколько понятий. *Цунгом* $G_ц$ называется взвешенный граф, в котором найдется хотя бы одна вершина, через которую проходят все контуры графа, и не существует ни одного пути, не являющегося частью одного из таких контуров.

Граф называется *приводимым к цунгу,* если при введении в него не более одной вершины и инцидентных ей дуг он преобразуется в цунг.

Автомат, граф переходов которого приводим к цунгу с инициальной вершиной, соответствующей началу и концу работы автомата при любом входном векторе *X,* определяющем реализуемую операцию и не изменяющемся в промежуточных состояниях, называется *микропрограммным.* В общем случае микропрограммный автомат реализует несколько операций, каждой из которых соответствует микропрограмма. При построении графа переходов, реализующего эти микропрограммы, будем производить склеивание совместимых состояний.

Внутренние состояния S_i, S_j называются *совместимыми* $S_i \equiv S_j$, если любой входной вектор, подаваемый на вход автомата, который находится в состоянии S_i не совпадает и не является частью ни одного входного вектора, подаваемого на вход автомата, который находится в состоянии S_j: $X_{S_i} \not\subset X_{S_j}$.

Подграф $G' = \langle V', U' \rangle$ графа $G = \langle V, U \rangle$ называется *двухинцидентным,* если каждой его вершине инцидентны две и только две группы параллельных дуг, входящих и исходящих, за исключением, быть может, минимального элемента $v^+ \in V'$ и максимального элемента $v^- \in V'$ множества V', причем в вершину v^+ могут входить, а из вершины v^- выходить несколько непараллельных дуг.

Последовательности микрокоманд, взвешивающие двухинцидентный подграф, называются *микролучом,* а число микрокоманд, входящих в последовательность,— его *длиной.*

При синтезе автомата управления, реализующего несколько микропрограмм, производим склеивание совместимых внутренних состояний, учитывая при этом два ограничения. Во-первых, склеивание следует начинать с конечных состояний, соответствующих различным операциям. Во-вторых, склеивают те совместимые состояния, в которых на выходе автомата вырабатывается одна и та же микрокоманда.

Рассмотрим микропрограммный автомат управления процессором ЭВМ для выполнения операций «Суммирование» (*a + b)*, «Вычитание» (*a — b)* и «Поразрядное сложение». Графы переходов, построенные по временным диаграммам, изображены на рис. 15.

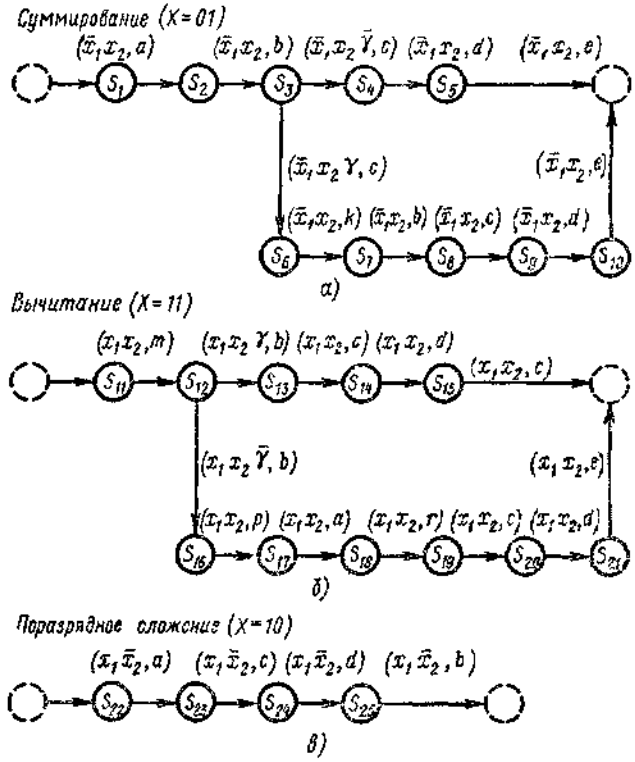


Рис. 15

Микрокоманды на этом рисунке обозначены латинскими буквами:
a = {ГД]; b = {HP}; c= {BΣ}; d= {СВ, ГС, B→C}; e = {CC, O}; k = {ГД, СВ, ГА, +1Т30Д}; m={ДА, ГД}; p={ДА, ДВ}; r={ HP, СВ}.

Микрооперации, являющиеся элементами микрокоманд, расшифровываются следующим образом: *ДА* — дополнение регистра *А; ГД* — гашение регистра Д; *НП* — начало пробега; *ДВ* — дополнение регистра *В; СВ* - счет регистра В; ВΣ — выдача суммы Σ; *ГА* — гашение регистра *А; + 1Т30Д* — «+1» в триггер *30* регистра Д; *ГС* - гашение регистра С; *В*→ С — передача из регистра *В* в регистр *С; СС* — счет регистра *С; О* — ответ о выполнении операции. Результат проводимых вычислений характеризуется логической переменной *ТОД* — значения триггера нуля регистра Д (на рис. 15 и в дальнейшем *ТОД* будем обозначать *γ).*

Произведем склеивание совместимых состояний. В результате получаем объединенный граф переходов *G* (рис. 16, а).

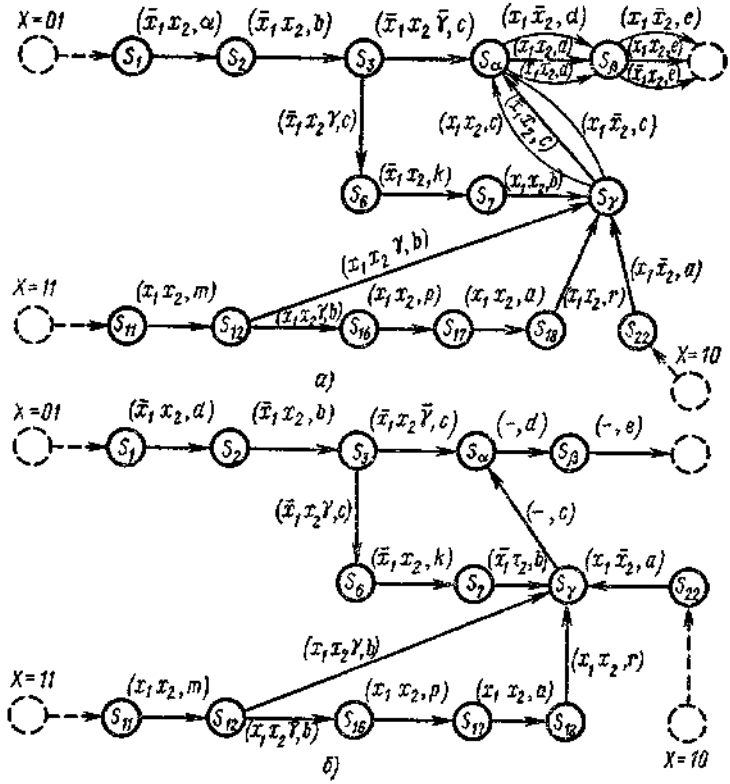


Рис. 16

На этом графе в состояниях S_γ, S_ω и S_β доопределено отображение $X \to Y$ добавлением $(\bar{x}_1, \bar{x}_2, c)$, $(\bar{x}_1, \bar{x}_2, d)$ и $(\bar{x}_1, \bar{x}_2, e)$ соответственно (рис. 16, б). После такого склеивашм совместимых состояний произведем свертывание двухинцидентных подграфов. Под *свертыванием* двухинцидентных подграфов понимаем замену этого подграфа на вершилу, взвешенную соответствующим микролучом. Микролучи имеют следующий вид:
$A = abc$; $B = de$; $D = kb$; $E = mb$; $M = par$.

В результате свертывания двухинцидентных подграфов получаем граф автомата управления (рис. 17, а), в котором нарушена детерминированность: одному состоянию, соответствующему вершине, в которую свернут двухинцидентный подграф, соответствует несколько микрокоманд.

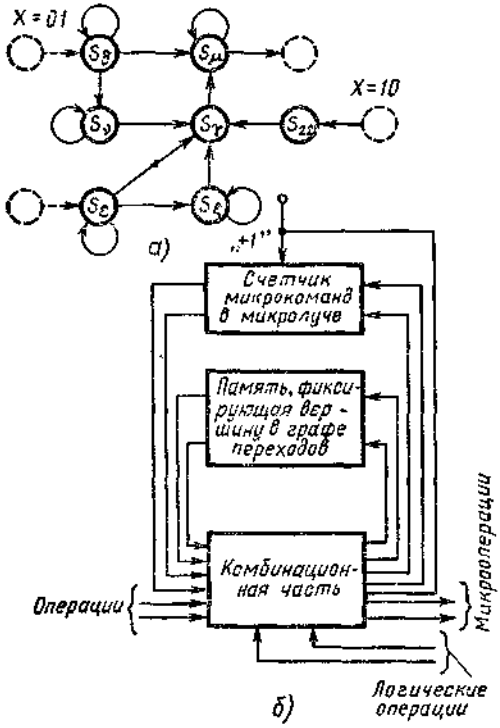


Рис. 17

Исходный граф переходов (рис. 16,б) получается как частичное декартово произведение графа *G* (рис. *17, а)* и двухинцидентного графа, конец которого соединен дугой с его началом и число вершин равно максимальной длине рассматриваемых микролучей (в данном

случае трем). Двухинцидентный граф, конец которого соединен дугой с началом, реализуется в виде счетчика.

Таким образом, детерминированность автомата и эквивалентность его функционирования сохраняем, вводя в обратную связь автомата счетчик микрокоманд в микролуче, увеличивающий свое значение до числа, равного максимальной длине микролуча $l_{макс}$ (рис. 17, *б).* При переходе в вершину v_{Gi}, соответствующую свернутому двухинцидентному подграфу G_i, на счетчике устанавливается число, равное $l_{макс} - l_i$, где l_i - длина микролуча, взвешивающего этот подграф. При этом l_i состояний счетчика взаимно однозначно сопоставляются микрокомандам, соответствующим вершине v_{Gi} . При каждом переходе в двух-инцидентном подграфе к содержимому счетчика прибавляется 1. Следовательно, совокупность кода вершин v_{Gi} и показания счетчика взаимно однозначно определяют выполняемую микрокоманду. Переполнение счетчика указывает на то, что автомат вышел из состояния, соответствующего вершине v_{G_i}.

Такая организация переходов позволяет не возбуждать выходы комбинационной части автомата, идущие в обратную связь в процессе выполнения микролуча, так как состояние памяти, запоминающей вершину v_{Gi}, остается неизменным, а переход счетчика из состояния в состояние происходит автоматически прибавлением единицы либо извне, либо с помощью микрооперации $+1 Сч\mu$ (прибавление единицы в счетчик микрокоманд), расширяющей множества микроопераций. При использовании последнего способа возбуждается только один выход, идущий в обратную связь.

Дальнейшие преобразования графа переходов произведем, стягивая незацепленные состояния. Два внутренних состояния (две вершины) графа переходов называются *незацепленными,* если они не входят ни в один из простых путей графа переходов при функционировании синтезируемого автомата.

Для предлагаемого преобразования графа переходов построим граф G_3 следующим образом. Каждой вершине графа переходов взаимно однозначно сопоставим вершину графа G_3; пара вершин $v_a, v_b (v_a \neq v_b)$ графа G_3 смежна, если в графе переходов существует простой путь, проходящий через вершины, соответствующие v_a и v_b, при функционировании синтезируемого автомата. Построенный таким образом граф G_3 будем называть *графом зацепления.*

С помощью операции раскраски вершин графа разобьем все множество вершин графа зацепления на подмножества, каждое из которых состоит из вершин, соответствующих незацепленным внутренним состояниям синтезируемого автомата.

Граф зацепления и его раскраска для автомата, изображенного на рис. 17, *а,* приведены на рис. 18, *а.*

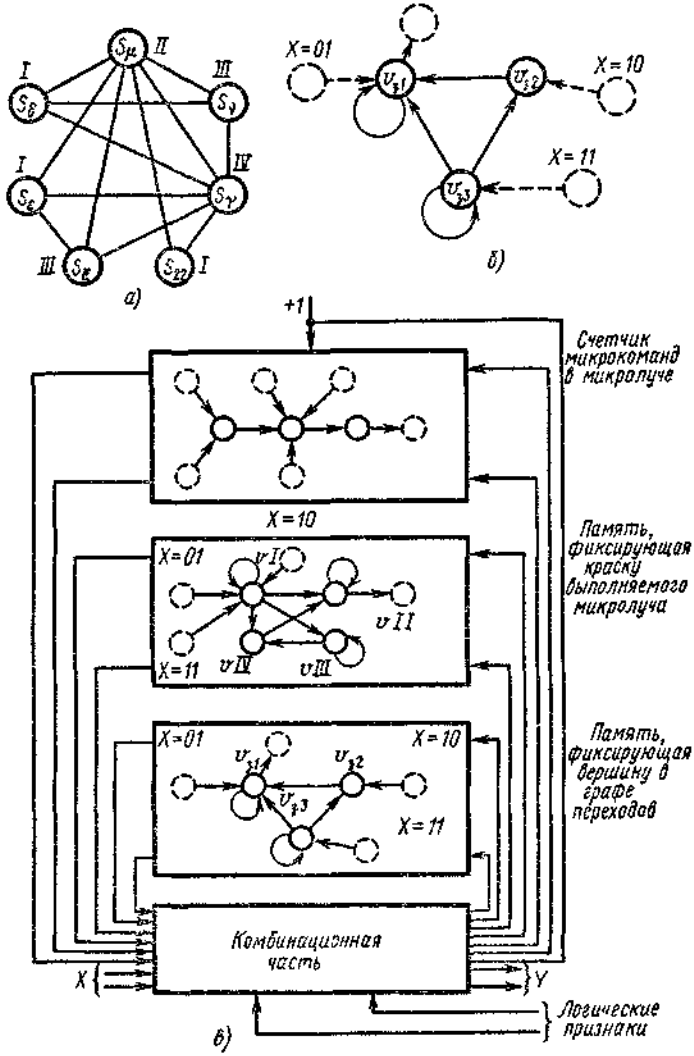


Рис. 18

В результате раскраски графа зацепления имеем следующие подмножества состояний, каждое из которых состоит из не зацепленных между собой состояний:

$$V_{\text{I}} = \{S_\delta,\ S_{22},\ S_\varepsilon\},\ V_{\text{II}} = \{S_\nu,\ S_\xi\},\ V_{\text{III}} = \{S_\mu\},\ V_{\text{IV}} = \{S_\gamma\}.$$

Из каждого подмножества $V_i(i = \text{I},\ldots,\text{IV})$ выбираем по одному элементу, например $S_\delta,\ S_\mu,\ S_\gamma,\ S_\nu$, из выбранных элементов образуем подмножество $V_{31} = \{S_\delta,\ S_\mu,\ S_\nu,\ S_\gamma\}$. Далее, снова из каждого подмножества V_i выбираем по одному из оставшихся элементов и снова образуем из них подмножество. Выборку осуществляем до тех пор, пока каждое из подмножеств V_i не преобразуется в пустое множество. Образованные в результате этой выборки подмножества и являются подмножествами, состоящими из зацепленных состояний. В данном случае они имеют следующий вид:

$$V_{31} = \{S_\delta,\ S_\mu,\ S_\nu,\ S_\gamma\},\ V_{32} = \{S_{22},\ S_\xi\},\ V_{33} = \{S_\varepsilon\}.$$

Каждое подмножество, состоящее из зацепленных состояний, заменяем одним состоянием, вес которого является совокупностью микролучей, соответствующих объединяемым вершинам (в частном случае микролуч может состоять из одной микрокоманды). Такую замену будем называть *стягиванием зацепленных состояний.* В результате стягивания зацепленных состояний графа переходов (рис. 17, *а)* получаем граф переходов $G_{\text{п}}$ (рис. 18,б).

Стягивания зацепленных состояний также нарушают детерминированность автомата. Детерминированность синтезируемого автомата и его эквивалентность сохраняем, введя в обратную связь элемент памяти, в котором будет фиксироваться краска, соответствующая микролучу, выполняемому в данный момент (рис. 18, в). Тогда совокупность кода вершины окончательного графа переходов, краска и показания счетчика микрокоманд в микролуче взаимно однозначно определяют выполняемую микрокоманду. В результате такого преобразования граф переходов можно представить в виде частичного декартова произведения соответствующих графов (рис. 18, в), что упрощает процесс кодирования внутренних состояний и практически уменьшает аппаратурные затраты при синтезе схемы возбуждения обратных каналов автомата.

Если контуры в графе переходов, соответствующем реализуемым микропрограммам, отсутствуют, то целесообразно разложить его только на два графа, исключив граф, показывающий смену красок. При этом проводят свертывание двухинцидентных подграфов в каждом графе переходов, соответствующем реализуемой операции, с последующим склеиванием совместимых состояний.

Если граф переходов является деревом, то в обратной связи автомата оставляют только счетчик. При этом граф переходов реализуется счетчиком при условии запоминания или наличия значений логических переменных, характеризующих вычисления. Начальному внутреннему состоянию автомата сопоставляется начальный код на счетчике, например код 0. Внутренние состояния, в которые автомат переходит из состояния S_i, имеющего код A, кодируются как $A + 1$. Смена кодов происходит прибавлением единицы в счетчик при каждом переходе. Тогда при наличии разветвления некоторые внутренние состояния будут иметь одинаковые коды. Детерминированность автомата при этом не нарушается благодаря хранению логических признаков, характеризующих проводимые вычисления. Показания счетчика и значения логических переменных однозначно определяют нужную микрокоманду. Данное утверждение справедливо, так как **граф переходов является деревом, т. е. не содержит циклов**. В конце выполнения операции происходит гашение счетчиков.

При представлении графа переходов, соответствующего автомату управления одной операцией, в виде частичного декартова произведения производят следующие преобразования:

1) склеивание псевдоэквивалентных состояний;
2) свертывание двухинцидентных подграфов, причем в результате изменения последовательности применения этих преобразований возникает несколько эквивалентных вариантов.

Рассмотрим граф переходов управления операцией деления (см. рис. 12). Стягивая двухинсйдентные подграфы, получаем граф переходов G (рис. 19, а).

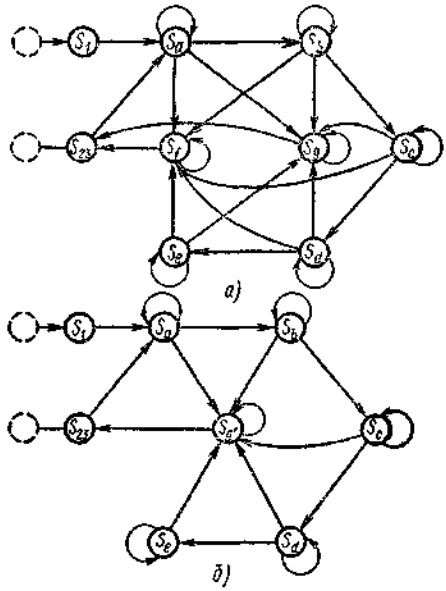


Рис. 19

Вершинам этого графа соответствуют следующие микролучи: $S_1 - a$;

$S_a - bcdcdcdc$; $S_b - dc$; $S_d - dc$; $S_e - dc$; $S_f - hk$; $S_g - efg$, $S_{23} - m$.

Следовательно, для восстановления эквивалентного функционирования необходим счетчик на восемь состояний. Вершины, соответствующие состояниям S_f и S_g в графе переходов (рис. 19, а), можно склеить при наличии запоминания признака p_0, детерминированность автомата при этом не нарушается. Окончательно имеем граф переходов G, изображенный на рис. 19.б. **Состояния счетчика, признак p_0 и код вершины графа переходов (рис. 19,б) однозначно определяют выполняемую микрокоманду.**

Внутренние состояния называются *псевдоэквивалентными,* если в них реализуется одна и та же микрокоманда. Для графа переходов $G_п$ (см. рис. 11) имеем следующее множества псевдоэквивалентных состояний: $M_c = \{S_3, S_5, S_7, S_9, S_{11}, S_{13}, S_{15}, S_{17}\}$; $M_d = \{S_4, S_6, S_8, S_{10}, S_{12}, S_{14}, S_{16}\}$.

Врезультате склеивания псевдоэквивалентных состояний (рис. 20, *а*) нарушается детерминированность автомата.

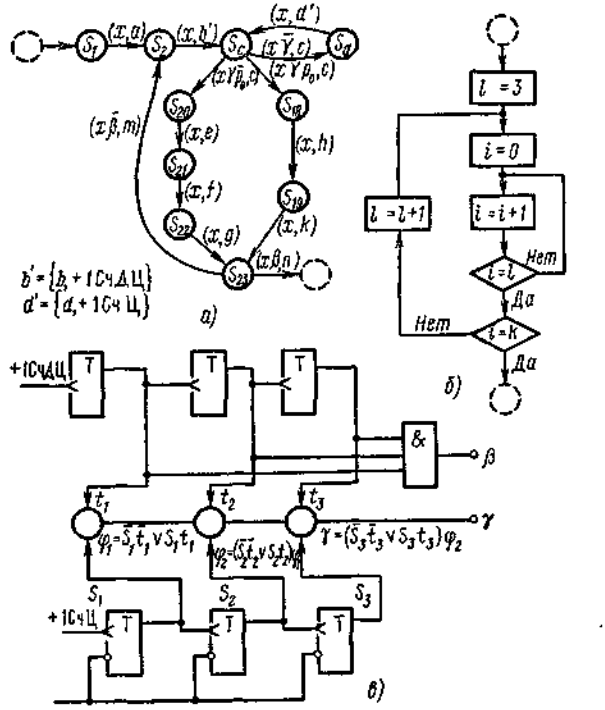


Рис. 20

Детерминированность и эквивалентность функционирования автомата восстанавливаем введением счетчика при организации цикла переменной длины. На рис. 20,б приведена программа работы счетчиков; в рассматриваемом автомате $k = 7$. Внешний цикл реализуется счетчиком длины цикла *СчДЦ.* Внутренний цикл реализуется счетчиком числа циклов *СчЦ.* Признак γ, который восстанавливает детерминированность автомата, вычисляется в блоке, включающем эти счетчики (рис. 20, в). В том же блоке вычисляется и признак окончания операции β (как сигнал переполнения счетчика *СчДЦ,* причем начальное состояние этого счетчика равно двум).

После восстановления детерминированности автомата производим склеивание двухинцидентных подграфов (рис. 21, а, б). Для восстановления детерминированности функционирования автомата после стягивания двухинцидентных подграфов введем еще два счетчика, имеющих три и два состояния соответственно.

При восстановлении детерминированности автомата с помощью введения счетчиков возникает задача минимизации числа счетчиков, которая сводится к задаче раскраски *графа затирания* $G_{ЗТ}$. Каждому счетчику взаимно однозначно сопоставляется вершина графа $G_{ЗТ}$: две вершины графа $G_{ЗТ}$ смежны, если подграфы работы соответствующих счетчикоз имеют хотя бы одну общую вершину (подграф G_α является подграфом работы счетчика α, если для реализации в этом подграфе правильных переходов необходимо знать состояние счетчика α).
Для рассматриваемого случая счетчикам *СчЦ, СчДЦ,* $Сч(S_\alpha)$ и $Сч(S_\beta)$ соответствуют подграфы, носители которых имеют соответственно следующий вид:

$$\{S_c,\ S_d\},\ \{S_2,\ S_c,\ S_d,\ S_\alpha,\ S_\beta,\ S_{23}\},\ \{S_\alpha\}\ \text{и}\ \{S_\beta\}.$$

Граф затирания $G_{ЗТ}$ для рассматриваемого случая изображен на рис. 21, в.

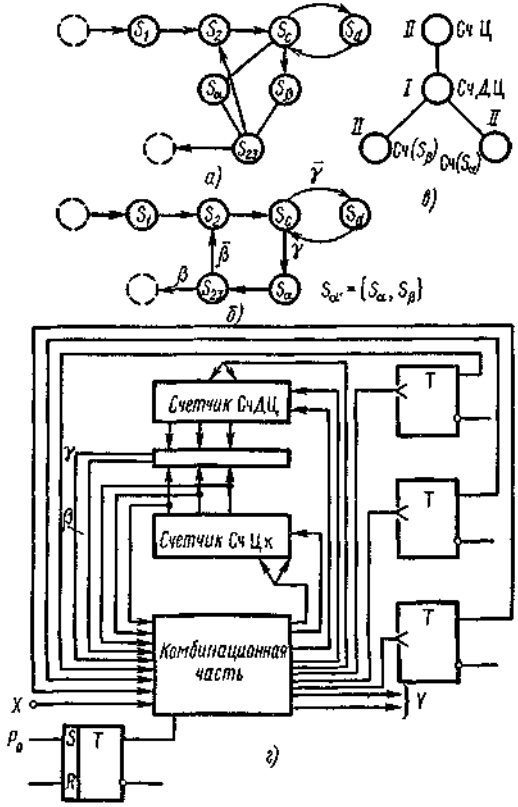


Рис. 21

Очевидно, два «счетчика», соответствующие несмежным вершинам в графе затирания, можно рассматривать как один физический счетчик, так как не найдется ни одного состояния, в котором для правильного функционирования автомата необходимо было бы знать состояние обоих счетчиков. Таким образом, минимизация числа счетчиков сводится к раскраске вершин графов затирания.

В данном случае, согласно минимальной раскраске вершин графа затирания (рис. 21,в), достаточно наличия двух счетчиков: счетчика *СчДЦ* и счетчика *СчЦК,* который включается в моменты работы счетчиков *СчЦ,* $Сч(S_\alpha)$ и $Сч(S_\beta)$ (рис. 21, г).

Примечание счетчиков в цепи обратной связи автоматов управления позволяет использовать стандартные узлы ЭВМ, выполненные в виде интегральных микросхем.

В результате поиска оптимальной декомпозиции автоматов получаем несколько эквивалентных вариантов, из которых выбираем окончательный вариант, имеющий минимальное значение $p(G^M)$:

$$p(G^M) = \frac{1}{|U|}\sum_{k=1}^{|U|}\left(\frac{1}{l(l-1)}\sum_{i=1}^{l-1}\sum_{j=i+1}^{l}\frac{f_i - 2f_{ij} + f_j}{j_{ij}}\right)_k, \quad (4)$$

где $|U|$ - количество слов в мографе, каждое из которых соответствует автоматному переходу и представляет собой XS^+S^-Y; l — число букв (термов), образующих слово XS^+S^-Y; под внешним знаком Σ находится выражение, являющееся средним значением производной $\partial G^M/\partial S$, вычисленной на парах букв, образующих слово XS^+S^-Y. Этому варианту соответствует более простая структурная реализация.

7.5. Кодирование внутренних состояний

На этапе *кодирования внутренних состояний* отображение $XS^+ \to S^-Y$, полученное на этапе абстрактного проектирования, преобразуется в отображение $XZ^+ \to Z^-Y$, где S^+, S^- — идентификаторы внутренних состояний, рассматриваемые соответственно в моменты времени t и $t+\tau$; Z^+, Z^- — коды этих состояний, элементами которых являются буквы используемого структурного алфавита; для случая булевой логики это 0 и 1.

Кодирование внутренних состояний автомата можно осуществлять, исходя из требований уменьшения аппаратурных затрат, либо увеличения надежности функционирования автомата, либо удовлетворения обоих требований одновременно.

Рассмотрим метод кодирования, удовлетворяющий первому требованию. Число способов кодирования N вершин графа переходов $G = \langle Y, V \rangle$ увеличивается с ростом $|V|$ как

$$N = \frac{(2^{[\log_2 |V|]} - 1)!}{(2^{[\log_2 |V|]} - |V|)!([\log_2 |V|])!}, \quad (5)$$

где [ ] — ближайшее целое число. Каждый способ кодирования определяет свои аппаратурные затраты при реализации автомата.

Наиболее интересным является *метод кодирования с использованием подстановочных разбиений,* предложенный Хартманисом, Стирнсом. Этот метод основан на уменьшении функциональной зависимости функций возбуждения. К сожалению, при кодировании состояний автомата с большим объемом памяти метод подстановочных разбиений очень трудоемок, что не позволяет применять его при кодировании внутренних состояний управляющих автоматов.

Рассмотрим *частотно-матричный метод кодирования* состояний. Пусть в результате абстрактного синтеза был построен мораф, определяющий отображение $XS^+ \to S^- Y$. После кодирования получим мораф, задающий отображение $XZ^+ \to Z^- Y$, т. е. в первоначальном мографе вершины, соответствующие идентификаторам внутренних состояний S_i, будут заменены полными подграфами, соответствующими кодам Z_i внутренних состояний. Данный метод кодирования можно реализовать с помощью следующего алгоритма синтеза кодирующего дерева.

1. Строим двумерную матрицу $Q = [q_{ij}]$, каждой строке которой взаимно однозначно соответствует микрооперация или значение входного канала (элементы вектора Y или *X),* столбцу — внутреннее состояние:

$$q_{ij} = \begin{cases} 1, \text{ если первичный терм, соответствующий } i\text{-й строке, входит} \\ \quad \text{в каждую пару векторов } (X, Y)(X \to Y), \text{ которые взвеши-} \\ \quad \text{вают дуги, исходящие из вершины } j; \\ 0 \text{ в противном случае.} \end{cases}$$

Каждому внутреннему состоянию (столбцу матрицы) взаимно однозначно сопоставляем произвольным образом максимальную вершину синтезируемого кодирующего дерева. Полагаем $Q = \tilde{Q}$.

2. По матрице $\tilde{Q}$ находим частотную матрицу отношений $\tilde{F} = \tilde{Q}^T \cdot \tilde{Q}$.

3. Вычислим значения производной от модели, заданной матрицей $\tilde{Q}$.

4. Выбираем пару внутренних состояний, для которой вычисленное значение производной минимально.

5. Выбранную пару состояний вычеркиваем и ставим ей в соответствие в кодирующем дереве вершину, являющуюся началом дуг, концами которых являются выбранные состояния. Построенной вершине взаимно однозначно сопоставляем столбец строящейся матрицы Q_c, равный векторному произведению столбцов матрицы $\ddot{Q}$, соответствующих выбранным вершинам.
6. Проверяем, остались или нет нерассмотренные внутренние состояния, для которых вычислялись значения производной. Если «да», то переходим к п. 4, в противном случае — к п. 7.
7. Проверяем, образована ли матрица Q_c. Если да, то полагаем $Q_c = \ddot{Q}$ и переходим к п. 2, в противном случае — к п. 8.
8. Каждым двум дугам, исходящим из вершины построенного дерева, начиная с дуг, исходящих из корня дерева, ставим в соответствие 0 и 1.

Путь, соединяющий корень дерева и максимальный элемент, взвешен кодом, который сопоставляется внутреннему состоянию, соответствующему этому максимальному элементу дерева.
9. Конец.

Проиллюстрируем предлагаемый алгоритм на следующем примере.

Пример 2. Пусть в результате абстрактного синтеза был получен граф переходов *G* (рис. 22, а).

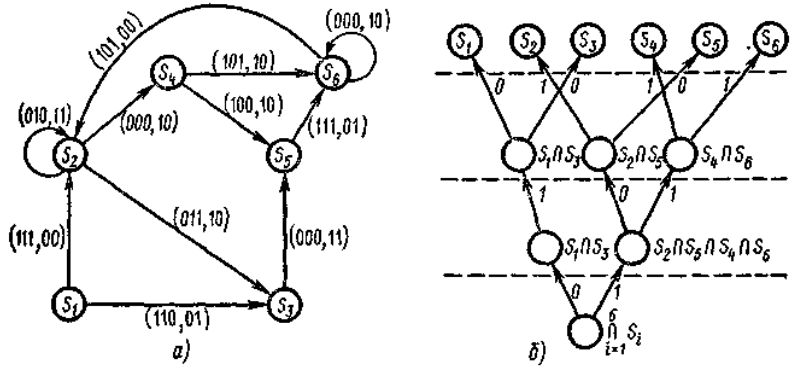


Рис. 22

Используя предлагаемый алгоритм, закодируем внутренние состояния автомата. Матрицу *Q*, соответствующую данному графу, предлагаем построить самостоятельно. Частотная матрица отношений соответствующая матрице *Q*, имеет следующий вид:

$$F = \begin{array}{c} \begin{matrix} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 \end{matrix} \\ \left\| \begin{matrix} 2 & 0 & 2 & 1 & 0 & 0 \\ 0 & 3 & 0 & 1 & 3 & 1 \\ 2 & 0 & 5 & 2 & 1 & 1 \\ 1 & 1 & 2 & 4 & 1 & 3 \\ 0 & 3 & 1 & 1 & 5 & 1 \\ 0 & 1 & 1 & 3 & 1 & 3 \end{matrix} \right\| \begin{matrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{matrix} \end{array}$$

Вычисляем значения производной для каждой пары состояний:

$$\frac{\partial G^M}{\partial S}(S_1, S_3) = \frac{f_{11} - 2f_{13} + f_{33}}{f_{13}} = \frac{2 - 2 \cdot 2 + 5}{2} = 1{,}5;$$

$$\frac{\partial G^M}{\partial S}(S_1, S_4) = \frac{f_{11} - 2f_{14} + f_{44}}{f_{14}} = \frac{2 - 2 \cdot 1 + 4}{1} = 4;$$

$$\frac{\partial G^M}{\partial S}(S_2, S_4) = 5; \quad \frac{\partial G^M}{\partial S}(S_3, S_6) = 6; \quad \frac{\partial G^M}{\partial S}(S_2, S_5) = 0{,}67;$$

$$\frac{\partial G^M}{\partial S}(S_4, S_5) = 7; \quad \frac{\partial G^M}{\partial S}(S_2, S_6) = 4; \quad \frac{\partial G^M}{\partial S}(S_4, S_6) = 0{,}33;$$

$$\frac{\partial G^M}{\partial S}(S_3, S_4) = 2{,}5; \quad \frac{\partial G^M}{\partial S}(S_5, S_6) = 6; \quad \frac{\partial G^M}{\partial S}(S_3, S_5) = 8.$$

Остальные значения производной равны ∞.

Производная $\frac{\partial G^M}{\partial S}(S_i, S_j)$ имеет минимальное значение на паре (S_4, S_6). Этой паре состояний ставим в соответствие «пересечение» соответствующих вершин строящегося дерева. Согласно алгоритму, объединяем в пары состояния S_2, S_5 и S_1, S_3. В результате построим следующий ярус дерева.

Матрица Q_c, соответствующая вершинам построенного яруса, имеет вид:

$$Q_c = \begin{array}{c} S_1 \cap S_3 \quad S_2 \cap S_5 \quad S_4 \cap S_6 \\ \left\| \begin{array}{ccc} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{array} \right\| \end{array} \begin{array}{l} x_1 \\ \bar{x}_1 \\ x_2 \\ \bar{x}_2 \\ x_3 \\ \bar{x}_3 \\ y_1 \\ \bar{y}_1 \\ y_2 \\ \bar{y}_2 \end{array}$$

Частотная матрица отношений, соответствующая матрице Q_c, имеет следующий вид:

$$F_c = \left\| \begin{array}{ccc} 2 & 0 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{array} \right\|.$$

Производная от модели $\Psi(Q_c)$ на парах состояний

$$\frac{\partial G}{\partial S}(S_2 \cap S_5,\ S_4 \cap S_6) = \frac{3 - 2 \cdot 1 + 3}{1} = 6.$$

Остальные значения производной равны ∞.

Искомое кодирующее дерево изображено на рис. 22, б. Согласно построенному кодирующему дереву, имеем следующие коды внутренних состояний автомата:

$S_1 - 010,\ S_2 - 100,\ S_3 - 011,\ S_4 - 110,\ S_5 - 101,\ S_6 - 111.$

Рассмотрим кодирование внутренних состояний автомата, исходя из удовлетворения требований надежности. Функционирование автомата может нарушаться в результате неодинаковой задержки в реальной схеме, реализующей автомат из-за явления гонок, сущность которого можно проиллюстрировав на следующем примере (рис. 23).

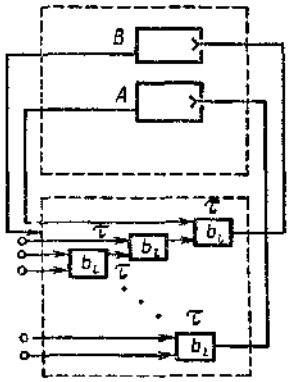


Рис. 23

Пусть в рассматриваемый момент переключаются два триггера *A* и *B* (первое условие), причем функция возбуждения φ одного из триггеров, например *B,* содержит в качестве одной из переменных значение триггера *A*, $\varphi_B = \varphi(\ldots, z^+_A, \ldots)$ (второе условие) и время задержки в схемах возбуждения триггеров *A* и *B* определяется следующим неравенством: $\Delta_{BA} > t_A$ (Δ_{BA} — время, в течение которого необходимо наличие «старого» значения триггера *A* в схеме возбуждения триггера *B;* t_A — время задержьи сигнала возбуждения триггера *A)* (третье условие). При выполнении этих трех условий значение триггера будет вычислено неправильно (так как для правильного вычисления значения триггера *B* необходимо, чтобы триггер *A* сохраняв свое «старое» значение по крайней мере в течение времени 2τ (после начала перехода автомата), а триггер *A* «обновляет» свое значение через τ). Это так назызаемое *явление гонок.* При наличии гонок автомат переходит не в то состояние, которое указано при данном переходе, что приводит к нарушению автоматного соответствия.

Гонки можно устранить, нарушив одно из трех условий гонок. Для того чтобы нарушить третье условие, т. е. чтобы выполнялось неравенство $\Delta_{BA} < t_A$,необходимо иметь схемы возбуждения триггеров *A* и *B* для определения Δ_{BA} и t_A. Эти схемы будут получены только на этапе структурного синтеза, для проведения которого необходимы результат кодирования внутренних состояний. Следовательно, при логическом устранении гонок нарушение третьего условия неосуществимо. Технически третье условие можно нарушить, вводя второй каскад элементов обратной связи или с помощью синхронного выполнения схемы автомата.

Второе условие наличия гонок — условие $\varphi_B = \varphi(\ldots, z^+_A, \ldots)$— можно

нарушить функциональным развязыванием элементов памяти, например, применяя подстановочные разбиения.

Первое условие (переключение двух (и более) элементов памяти при одном переходе автомата) нарушается, если при любом переходе автомата переключается только один элемент памяти. Это означает, что каждому внутреннему состоянию автомата сопоставляется такой код, что если существует переход из состояния S_i в состояние S_j $(S_i \neq S_j)$, то соответствующие им коды отличаются только в одном разряде. Другими словами, при реализации автомата гонки отсутствуют, если соответствующий граф переходов можно расположить в n-мерном гиперкубе так, чтобы переходы из состояния S_i в состояние S_j $(S_i \neq S_j)$ осуществлялись только по ребрам гиперкуба и коды соответствующих вершин гиперкуба были сопоставлены вершинам графа переходов. Такое кодирование называется *соседним.*

Будем говорить, что для графа переходов возможно соседнее кодирование кодами длины n, если он вложим в n-мерный куб. Приведем условие вложимости графа переходов в гиперкуб.

Из анализа гиперкуба следует, что если хроматическое число $h(G)$ графа переходов G (без учета дуг, являющихся петлями) больше двух, то для такого графа переходов не существует соседнего кодирования. Таким образом, для того чтобы граф переходов G имел соседнее кодирование, требуется, чтобы он не содержал циклы нечетной длины. Если же он их содержит, то необходимо устранить все циклы нечетной длины, вводя дополнительные внутренние состояния, в которых не реализуется отображение $X \rightarrow Y$. Эти состояния называют *неустойчивыми* внутренними состояниями.

Рассмотрим точный алгоритм соседнего кодирования, основанный на учете запрещенных фигур, характеризующих кубируемость кодируемого графа.

Пример 3. Пусть задан граф переходов автомата (рис. 24), который изображен без петель, кратных дуг и ориентации (поскольку это несущественно при решении задачи соседнего кодирования).

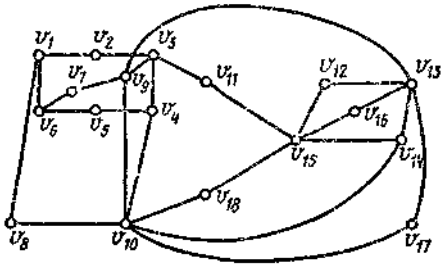


Рис. 24

В соответствии с выведенными ранее условиями, для того чтобы соседнее кодирование было возможно, необходимо и достаточно, чтобы граф не содержал запрещенных критических подграфов. При проектировании автомата необходимо так преобразовать эти графы, чтобы все они стали кубируемыми. Данное преобразование не нарушает функционирование автомата только в том случае, если каждый новый граф гомеоморфен предшествующему. Практически преобразование сводится к добавлению новых вершин в некоторые ребра каждого исходного критического графа, являющегося подграфом данного графа переходов. Этим самым вводим неустойчивые внутренние состояния автомата. При оптимальном проектировании число вводимых неустойчивых состояний должно быть минимальным. Двумерная таблица (табл. 7) распределения ребер по запрещенным фигурам (рис. 25) покрывается двумя строками $\{l_{13}, l_{21}\}$.

Таблица 7

	R_1	R_2	R_3	R_4	R_5	R_6	R_7
$l_1 = \{v_1, v_2\}$	1	0	0	0	0	0	0
$l_2 = \{v_2, v_3\}$	1	0	0	0	0	0	0
$l_3 = \{v_3, v_4\}$	1	1	0	0	0	0	0
$l_4 = \{v_4, v_5\}$	1	0	0	0	0	0	0
$l_5 = \{v_5, v_6\}$	1	0	0	0	0	0	0
$l_6 = \{v_1, v_6\}$	1	0	0	0	0	0	0
$l_7 = \{v_6, v_7\}$	1	0	0	0	0	0	0
$l_8 = \{v_7, v_9\}$	1	0	0	0	0	0	0
$l_9 = \{v_3, v_9\}$	1	1	0	0	0	0	0
$l_{10} = \{v_1, v_8\}$	1	0	0	0	0	0	0
$l_{11} = \{v_8, v_{10}\}$	1	0	0	0	0	0	0
$l_{12} = \{v_4, v_{10}\}$	1	1	0	0	0	0	0
$l_{13} = \{v_9, v_{10}\}$	1	1	1	0	1	1	1
$l_{14} = \{v_9, v_{13}\}$	0	1	1	0	1	1	1
$l_{15} = \{v_{10}, v_{18}\}$	0	0	1	0	0	1	1
$l_{16} = \{v_{10}, v_{14}\}$	0	1	0	0	1	0	0
$l_{17} = \{v_{10}, v_{17}\}$	0	0	1	0	1	1	1
$l_{18} = \{v_{15}, v_{18}\}$	0	0	1	0	0	1	1
$l_{19} = \{v_3, v_{11}\}$	0	1	0	0	0	0	0
$l_{20} = \{v_{11}, v_{15}\}$	0	1	0	0	0	0	0
$l_{21} = \{v_{12}, v_{13}\}$	0	1	1	1	0	1	0
$l_{22} = \{v_{12}, v_{15}\}$	0	1	1	1	0	1	0
$l_{23} = \{v_{13}, v_{14}\}$	0	1	1	1	1	0	1
$l_{24} = \{v_{14}, v_{15}\}$	0	1	1	1	0	0	1
$l_{25} = \{v_{15}, v_{16}\}$	0	0	0	1	0	1	1
$l_{26} = \{v_{13}, v_{10}\}$	0	0	0	1	0	1	1
$l_{27} = \{v_{13}, v_{17}\}$	0	0	1	0	1	1	1

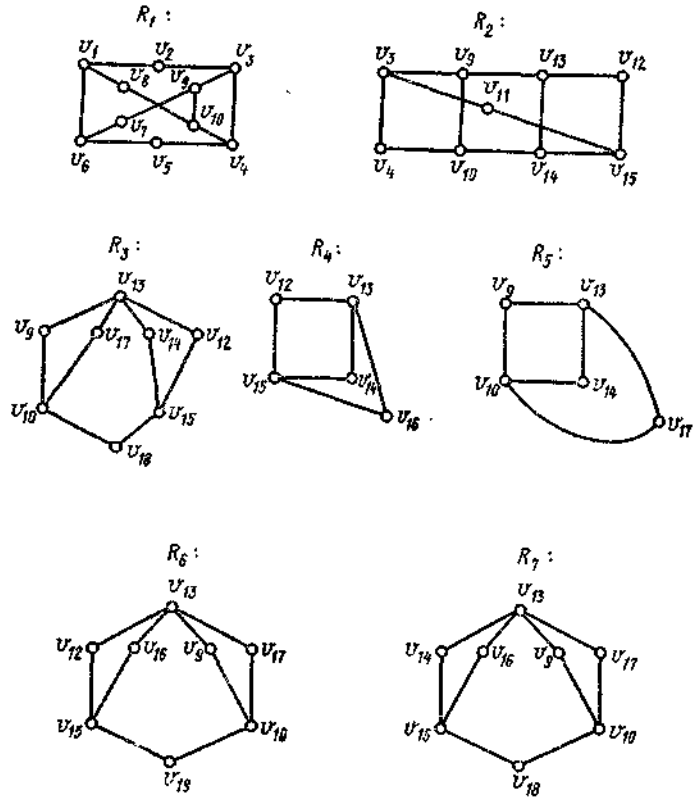


Рис.25

Добавляя между вершинами v_9, v_{10} и v_{12}, v_{13} вершины, соответствующие неустойчивым состояниям, вкладываем полученный граф в булево пространство (рис. 26).

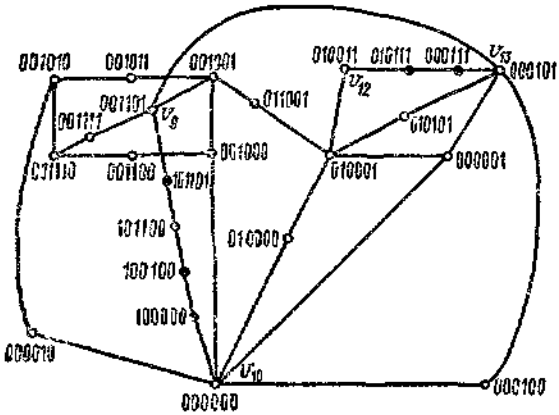


Рис. 26

При введении дополнительных вершин в граф разрешенные фигуры (циклы четной длины) не были нарушены.

7. 6. Структурное проектирование автоматов

Рассмотрим проектирование логической схемы как синтез соответствующего булева графа. Разобьем все множество базисов на два класса: *топологические* и *функциональные.*

В топологических базисах полнота используемых при синтезе функций достигается соответствующим физическим моделированием двоичных переменных и определенным соединением заданных элементов. Такие базисы образуют все ключевые элементы, токовые ключи, криотроны, переключатели света, спейсисторы, деплисторы, унитроны и другие вентильные элементы, которые используются в вычислительной технике как элементы с высоким быстродействием.

В функциональных базисах элементы, независимо от их соединения и физического моделирования двоичных переменных, реализуют функции, образующие полную систему булевых функций. К таким базисам относятся, например, элементы, которые реализуют функции Шеффера, Вебба, импликации и др.

Множество топологических базисов, в свою очередь, разобьем на четыре класса. Пераый и второй классы образуют элементы, пропускающие информационный сигнал в одном направлении при совпадении значения управляющего сигнала с буквой, взвешивающей этот элемент. Но в элементах первого класса прошедший сигнал распро-

страняется по каналам с односторонней проводимостью, а в элементах второго класса — по каналам с двусторонней проводимостью. К первому классу относятся диодно-вентильные элементы, в случае оптической обработки информации с использованием волоконной техники — переключатели света, ко второму - вентильные элементы (например, спейсистор-триод).

На входах схемы, составленной из рассматриваемых элементов, как и на входах схемы, построенной из элементов любого топологического базиса, используется парафазное представление двоичной информации.

Третий и четвертый классы образуют элементы, пропускающие информационный сигнал в обоих направлениях при совпадении значения управляющего сигнала с буквой, взвешивающей этот элемент. В элементах третьего класса прошедший сигнал распространяется по каналам с односторонней проводимостью, а в элементах четвертого класса - по каналам с двусторонней проводимостью.

К элементам третьего класса относятся, например, туннель-троны и диодно-контактные элементы; к элементам четвертого — ключевые элементы с двусторонней проводимостью (криотроны, контакты).

Таким образом, **все множество топологических базисов разбито на четыре класса по двум признакам: проводимости элемента и проводимости соединительного канала.**

В результате абстрактной оптимизации и кодирования внутренних состояний автомата автоматный оператор задается системой функций возбуждения и функций выхода. Построение оптимальной по сложности схемной реализации этих функций — трудоемкий процесс, что обусловлено комбинаторной природой задачи. Один из эффективных путей снижения трудоемкости схемной реализации автоматного оператора - сокращение размерности задач, достигаемое с помощью **декомпозиции**. Для полученных в результате декомпозиции функций меньшей размерности можно построить оптимальные схемы. Оптимальная декомпозиция направлена на достижение глобальной оптимальности решения задачи схемной реализации автоматного оператора. Декомпозиция функции $f(X)$, $X = \{x_1, \ldots, x_n\}$- это представление ее в виде суперпозиции нескольких функций меньшей размерности: $f(X) = F(\varphi_1(X_1), \varphi_2(X_2), \ldots, \varphi_k(X_k))$, где $X_i \subseteq X$.

Функция F называется *внешней,* функции $\varphi_1, \ldots, \varphi_k$ - *сопряженными.*

Критерием качества декомпозиции является минимум числа сопряженных функций.

Декомпозиция должна удовлетворять следующим практическим требованиям:

1) учитывать особенности функций, встречающихся в практике проектирования ;
2) повышать надежность проектируемых автоматных устройств за счет снижения вероятности возникновения риска.

Первое требование предполагает, что метод должен быть хорошо применим к наиболее часто встречающимся на практике слабоопределенным функциям, т. е. функциям, мощность определенной области которых намного меньше мощности неопределенной области. Второе требование накладывает ограничение на длину путей в схемной реализации функции, которые определяются глубиной вхождения соответствующих аргументов в функцию. Например, вероятность возникновения риска для функции $f(X) = F(\varphi_1(\varphi_{11}(x_1\ldots), \varphi_{12}(\varphi_{121}(x_2\ldots), \varphi_{122}(x_3\ldots), \ldots), \varphi_{13}\ldots), \varphi_2(\varphi_{21}(x_4\ldots), \ldots, x_5), \ldots, x_6)$ пропорциональна сумме модулей разностей глубин вхождения аргументов в подфункции. В приведенной функции глубина вхождения переменной x_6 равна нулю, а глубина вхождения переменной x_3 - трем. Справедливость такого утверждения основывается на следующих предположениях: события возникновения риска в сопряженных функциях являются независимыми; вероятность возникновения риска в функции пропорциональна модулю разности длин путей, проходимых сигналом, т. е. вероятность возникновения риска в представлении функции $F(\varphi_1(x_1, \ldots), x_2)$ в два раза меньше, чем в представлении функции $F(\varphi_1(\varphi_2(x_3, \ldots), x_1), x_2)$. Следовательно, вероятность возникновения риска для функции *f(X)* пропорциональна дебалансу длин путей d_l в схеме, реализующей *f(X)*, $d_l = \sum |l_{cp} - l_i|$, где $l_{ср}$ — средняя длина пути, l_i — длина *t*-го пути. Дебаланс d_l можно оценить дебалансом глубин вхождения d_v, получающимся из d_l заменой l_i на v_i, l_{cp} на $v_{ср}$, где $v_{ср}$ — средняя глубина вхождения переменной в функцию, v_i — глубина вхождения в функцию переменной x_i.

Второе из приведенных выше требований автоматически выполняется в декомпозиции *f(X)* вида

$$F(\varphi_1(X_A), \ldots, \varphi_p(X_A), \xi_1(X_B), \ldots, \xi_s(X_B)), \quad (13)$$

где $X_A \bigcup X_B = X$, $X_A \bigcap X_B = \varnothing$.

Рассмотрим метод структурно-функциональной декомпозиции вида (13), ориентированный на слабоопределенные функции. В его основе лежит представление функции К-графом. К-графом называется двудольный граф $G^K = \langle V^A, V^B, U_1, U_0 \rangle$, в котором множества V^A и V^B определяют множество вершин G^K, а U_1, U_0 - множества, задающие ребра двух видов, которые соединяют вершины из V^A и V^B.

Разбиение множества *X* на X_A и X_B влечет разбиение каждого набора m_i единичной области определения функции *f(X)* на два поднабора

m_i^A и m_i^B, которые состоят из переменных, входящих соответственно в X_A и X_B. Нулевая и единичная области определяются совокупностями пар поднаборов $(m_i^A,\ m_i^B)$. Таким образом, функция *f(X)* с заданным разбиением *X* на X_A и X_B представляется К-графом $G^K = \langle V^A,\ V^B,\ U_1,\ U_0 \rangle$, где V^A - множество поднаборов m_i^A, V^B — множество поднаборов m_i^B; U_1, U_0 — бинарные отношения, определяющие единичную и нулевую области функции.

Рассмотрим слабоопределенную булеву функцию $f(X)$, $X = \{x_1,\ x_2,\ x_3,\ x_4\}$. Функция *f(x)* принимает значение 1 на множестве наборов $M_1 = \{1010,\ 0111,\ 1101,\ 1100\}$; значение 0 — на множестве наборов $M_0 = \{0001,\ 1000,\ 1011\}$. Пусть $X_A = \{x_1,\ x_2\}$, $X_B = \{x_3,\ x_4\}$.

К-граф G^K, представляющий *f(X)* с данным разбиением *X*, изображен на рис. 27, *а*. Перечеркнутые ребра образуют множество U_0, неперечеркнутые — множество U_1.

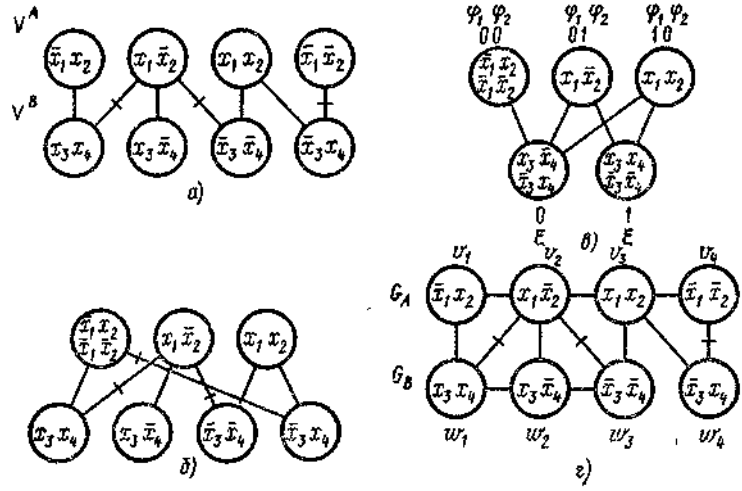


Рис. 27

Определим операцию склеивания вершин в К-графе. Две вершины одного класса v_1 и v_2 могут быть склеены, если не существует вершины *w* другого класса, связанной с одной из вершин ребром, а с другой — перечеркнутым ребром. В результате склеивания вершины заменяются на одну, а их окрестности объединяются. Например, после склеивания вершин, соответствующих наборам $\bar{x}_1 x_2$ и $\bar{x}_1 \bar{x}_2$, К-граф имеет вид, приведенный на рис. 27, *б*. Заметим, что склеивание вершин одного

класса изменяет условия склеивания вершин другого класса. Будем производить склеивание вершин К-графа до тех пор, пока это возможно. Результирующий К-граф изображен на рис. 27, *в*.

Теорема 2. *Булева функция* $f(X) = f(x_1, x_2, \ldots, x_n)$ *представима в виде* $f(X) = F(\varphi_1(X_A), \ldots, \varphi_p(X_A), \xi_1(X_B), \ldots, \xi_s(X_B))$, $X_A \cap X_B = \varnothing$, $X_A \cup X_B = X$, $p < |X_A|$, $s < |X_B|$, *тогда и только тогда, когда существует последовательность операций склеивания, приводящая К-граф* $G^K = \langle V^A, V^B, U_1, U_0 \rangle$, *представляющий функцию, к К-графу* $G^K = \langle V_1^A, V_1^B, U_1', U_0' \rangle$, *у которого* $|V_1^A| \leqslant 2^p$, $|V_1^B| \leqslant 2^s$.

Критерием качества декомпозиции является минимум числа функций φ_i и ξ_i, что достигается минимизацией количества вершин в классах К-графа после склеивания. Процесс склеивания вершин равносилен раскраске графов несовместимости $G_A = \langle V^A, U^A \rangle$ и $G_B = \langle V^B, U^B \rangle$. Носителями этих графов являются классы вершин V^A и V^B. Две вершины v_1 и v_2 графа несовместимости смежны, если одна из них связана с некоторой вершиной из другого класса ребрам, а другая — перечеркнутым ребром. Графы несовместимости для К-графа, изображенного на рис. 27, *а,* приведены на рис. 27, *г*.

Фиксация соцветности двух вершин графа несовместимости равносильна их склеиванию и поэтому в общем случае приводит к введению новых ребер в другой граф несовместимости. Если соцветными становятся вершины v_1 и v_2 графа G_i $(i = A, B)$, то в G_j $(j = A, B, j \neq i)$ вводятся ребра (w_1, w_2) такие, что в конфигурации G_A, G_B v_1 и w_1 связаны ребром, а v_2 и w_2 — перечеркнутым ребром. Отсюда следует, что раскраска конфигурации графов несовместимости в общем случае не определяется их независимой раскраской. Процесс склеивания равносилен связной раскраске графов несовместимости. Под *связной раскраской* конфигурации графов G_A, G_B понимается такое разбиение вершин графов на классы (и присвоение вершинам каждого класса одного цвета), при котором вершины в каждом классе попарно не смежны и, кроме того, вершины любого класса в G_A не могут быть связаны одновременно ребрами и перечеркнутыми ребрами с вершинами некоторого класса G_B. Таким образом, декомпозиция слабоопределенной булевой функции сводится к связной раскраске графов несовместимости.

Хотя связная раскраска в общем случае не определяется независимой раскраской каждого из графов, выявление условий, при которых независимая раскраска графов несовместимости дает связную

раскраску, представляет большой интерес, поскольку задача декомпозиции в этом случае значительно упрощается.

Теорема 3. *Связная раскраска графов G_A, G_B определяется независимой их раскраской тогда и только тогда, когда конфигурация G_A, G_B не содержит в качестве подграфа конфигурацию G_{AB}* (рис. 28).

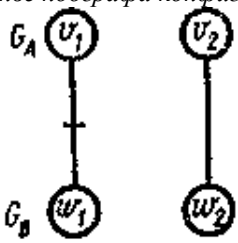


Рис. 28

Необходимость. Пусть связная раскраска графов G_A, G_B сводится к их независимой раскраске. Это означает, что связная раскраска любого подграфа конфигурации также определяется независимой раскраской входящих в нее графов. Связная раскраска конфигурации G_{AB} не определяется независимой раскраской, поэтому конфигурация G_A, G_B не может содержать G_{AB} в качестве подграфа.

Достаточность. Пусть связная раскраска графов G_A, G_B не определяется их независимой раскраской. Пусть также фиксация соцветности несмежных вершин v_1, v_2 графа G_A приводит к введению ребра (w_1, w_2) в графе G_B. Это возможно только в случае, когда одна из вершин v_1, v_2 связана с одной из вершин w_1, w_2 ребром, а другая с оставшейся — перечеркнутым ребром. Других ребер между v_1, v_2 и w_1; w_2 быть не может, поскольку в противном случае они были бы смежны v_1, v_2 или w_1, w_2. Но описанная конфигурация v_1, v_2, w_1, w_2 является с точностью до переобозначения конфигурацией G_{AB}. Следовательно, если G_{AB} не присутствует в качестве порожденного подграфа в конфигурации G_A, G_B, то связная раскраска последней определяется независимой раскраской G_A и G_B.

Знание конфигурации G_{AB} позволяет предложить конструктивный аппарат сведения связной раскраски к независимой. Для каждого подграфа конфигурации, изоморфного G_{AB}, с вершинами v_1, v_2, w_1, w_2, введем вершины v'_1, v'_2, w'_1, w'_2. Соединим v'_1 с вершинами $\Gamma(v_1)$, v'_2 — с вершинами $\Gamma(v_2)$, w'_1 — с вершинами $\Gamma(w_1)$, w'_2 — с вершинами $\Gamma(w_2)$, а также введем ребра (v'_1, v'_2) и (w'_1, w'_2). Полученную конфигурацию назовем *расширением* и обозначим G^*_A, G^*_B. Найдем независимую раскраску G^*_A и G^*_B. Проведем теперь операцию приведения: для каждой конфигурации G_{AB} в случае, если v_1 соцветна с v_2, а w_1 соцветна с w_2, удаляем вершины v_1, v_2, w_1, w_2; если же хотя бы одна пара из

v_1, v_2 и w_1, w_2 оказалась несоцветна, то удаляем v'_1, v'_2, w'_1, w'_2. В результате получена связная раскраска конфигурации G_A, G_B.

Пример 4. Расширение G_A, G_B для приведенной на рис. 27, а конфигурации G_A, G_B и независимая раскраска G_A и G_B показаны на рис. 29.

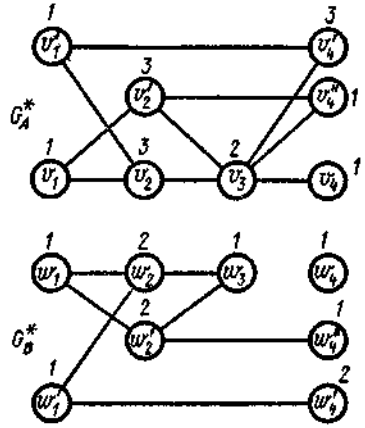


Рис. 29

В результате проведения G_A, G_B получаем склеивание К-графа, изображенного на рис. 27, в.

Закодируем вершины каждого класса. Длина кода для каждого класса равна $\lceil \log_2 s \rceil$, где s - число вершин в классе. Каждый код определяет набор, определенный на новых переменных, которые получены в результате кодирования: φ_i — для одного класса, ξ_i — для другого. Таким образом, К-граф определяет новую функцию F, являющуюся внешней в декомпозиции (13). Декомпозиция рассматриваемой функции определяется следующим образом:

$$f(x_1, x_2, x_3, x_4) = F(\varphi_1(x_1, x_2), \varphi_2(x_1, x_2), \xi(x_3, x_4)).$$

Функция $F(\varphi_1, \varphi_2, \xi)$ принимает единичное значение на множестве наборов $M_1 = \{\bar{\varphi}_1\bar{\varphi}_2\bar{\xi}, \bar{\varphi}_1\varphi_2\bar{\xi}, \bar{\varphi}_1\varphi_2\xi\}$ и нулевое значение на множестве наборов $M_0 = \{\bar{\varphi}_1\varphi_2\xi, \bar{\varphi}_1\bar{\varphi}_2\xi\}$. При этом φ_1, φ_2 и ξ также слабоопределенные функции.

Запишем для них множества единичных и нулевых наборов:

$M_1(\varphi_1) = \{x_1x_2\}$, $M_0(\varphi_1) = \{\bar{x}_1x_2, \bar{x}_1\bar{x}_2, x_1\bar{x}_2\}$;

$M_1(\varphi_2) = \{x_1\bar{x}_2\}$, $M_0(\varphi_2) = \{\bar{x}_1x_2, \bar{x}_1\bar{x}_2, x_1x_2\}$;

$M_1(\xi) = \{x_3x_4, \bar{x}_3\bar{x}_4\}$, $M_0 = (\xi) = \{x_3\bar{x}_4, \bar{x}_3x_4\}$.

Сигнатура алгебр определяет *законы композиции,* свойства которых задаются тождествами алгебр. Законы композиции позволяют определять функционирование целого по функционированию его частей. При синтезе систем функционирование целого задано и

необходимо определить функционирование его частей, т. е. построить структуру, реализующую заданное функционирование. При синтезе представляют интерес не законы композиции, а *законы декомпозиции.*
Рассмотрим модель

$$\Psi = \langle M, P_1^f, P_2^f, \ldots, P_n^f \rangle,$$

где

$$P_i^f(m_1, m_2, \ldots, m_k) = \begin{cases} 1 \text{ при } m_k = f_i(m_1, m_2, \ldots, m_{k-1}), \\ 0 \text{ в противном случае.} \end{cases}$$

Аналогично понятию *алгебра*

$$A \langle M, f_1, f_2, \ldots, f_n \rangle,$$

введем понятие *коалгебра*

$$K = \langle M, \varkappa_1^f, \varkappa_2^f, \ldots, \varkappa_n^f \rangle,$$

где M - носитель коалгебры, $\varkappa_1^f, \varkappa_2^f, \ldots, \varkappa_n^f$ - ее сигнатура. Процедура определения по m_k множества $\{m_1, m_2, \ldots, m_{k-1}\}$ такого, что $m_k = f_i(m_1, m_2, \ldots, m_{k-1})$, называется *ко-операцией* $\varkappa_i^f(m_k)$,

$$\varkappa_i^f(m_k) = \{m_1, m_2, \ldots, m_{k-1}\}.$$

Очевидно, что если результат операций f_i строго однозначен, то результат ко-операции $\varkappa_i^f$ не однозначен. Алгебра определяет законы композиции, коалгебра - законы декомпозиции.
Аналогично существованию, например, алгебры Буля, алгебры Вебба, импликативной алгебры, алгебры Жегалкина, при развитии теории коалгебр следует ожидать появления коалгебры Буля, коалгебры Вебба, импликативной коалгебры, коалгебры Жегалкина.
В настоящее время разработана только коалгебра графов, являющаяся изоморфной коалгебре Буля, в которой носитель задан диаграммами Хассе или структурными графами.
В структурном графе каждая вершина взвешена первичным термом $x_i^{\sigma_i}$ и путь взаимно однозначно соответствует максимальному интервалу (простой импликанте) булевой функции $f(x_1, x_2, \ldots, x_n)$.
Коалгеброй графов называется совокупность вида

$$K = \langle M, \varkappa^{\vee}, \varkappa^{-} \rangle,$$

где носителем M является множество всевозможных структурных графов, а сигнатурой - *ко-операция дизъюнкции* $\varkappa^{\vee}$ и *ко-операция отрицания* $\varkappa^{-}$ структурных графов. В дальнейшем эти две ко-операции будем называть соответственно *операцией разложения* $\varkappa^{\vee}$ и *операцией инверсии* $\varkappa^{-}$ графов.
Операция разложения $\varkappa^{\vee}$ графов. При выполнении этой операции указывают направления, по которым происходит разложение графа Н. Направление задается вершинами, являющимися максимальными элементами подграфов графа *Н.*

Операция разложения $\varkappa^{\vee}(H)$ графа H по направлениям V_i ($i = 1, ..., k$) есть выделение соответственно подграфов H_i ($i = 1, ..., k$), состоящих из всех вершин $\{v_j / j = 1, \ldots, l\}$ и соединяющих их дуг, для которых в графе H найдется путь, соединяющий $v_a \in \{v_j / j = 1, \ldots, l\}$ и $v_b \in V_i$.

Например, результат операции разложения структурного графа H по направлениям $\{x_3\}$ и $\{x_2, \bar{x}_3\}$ произведен на рис. 30. Повторяющиеся буквы отмечены штрихом, чтобы они идентифицировали вершины графа.

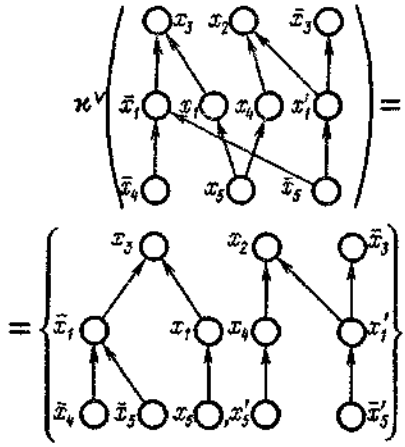


Рис. 30

Операция инверсии $\varkappa^{-}$ графов. Структурный граф, представленный в виде суперпозиции структур типов π и σ, называется *графом типа* πσ.

Операция инверсии $\varkappa^{-}(H)$ структурного графа H — это приведение данного графа с помощью его разложения к графу типа πσ , замена каждого подграфа типа σ подграфом типа π и каждого подграфа типа π - подграфом типа σ , при этом каждый вес $x_i^{\sigma_i}$ первоначального графа заменяется весом $x^{(\sigma_i + 1) \bmod 2}$ на соответствующих вершинах полученного графа.

Запрещенными фигурами свойства структурных графов быть графами типа πσ являются фигуры $H_г$ (рис. 31).

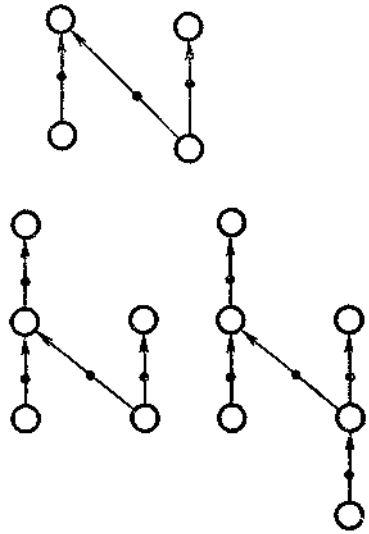


Рис. 31

Следовательно, минимальное расщепление структурного графа при приведении его к типу πσ-графов определяется минимальным покрытием таблицы, в которой запрещенными фигурами являются подграфы $H_г$, а их компонентами — вершины со степенью, равной двум (минимальные и максимальные вершины диагонали), одна из которых при преобразовании расщепляется.

В рассматриваемом примере (см. рис. 30) описанную таблицу можно представить в виде табл. 8.

Таблица 8

Компоненты		$H_{Г_1}$	$H_{Г_2}$	$H_{Г_3}$	$H_{Г_4}$
a	$(\bar{x}_1, x_3)$	1	0	0	0
b	$\bar{x}_5$	1	0	0	0
c	x_2	0	1	0	1
d	$(\bar{x}_5, x_1)$	0	1	0	0
e	x_3	0	0	1	0
f	x_3	0	0	1	1

Имеем шесть покрытий этой таблицы:

$$(a \vee b)(c \vee d)(e \vee f)(c \vee f) = ace \vee bec \vee bcf \vee adf \vee bdf \vee afc.$$

Для определенности выбираем покрытие $\{b,\ c,\ f\}$. Тогда рассматриваемый структурный граф H представим декомпозицией подграфов π, σ:

$$H = \sigma(\pi(x_3, \sigma(\pi(x_1, x_5), \pi(\bar{x}_1\sigma(\bar{x}_4, \bar{x}_5)))),$$
$$\pi(x_2, x_4, x'_5), \pi(\sigma(x'_2, \bar{x}_3), x'_1, \bar{x}'_5)).$$

В этом случае инверсия графа $\varkappa^{-}(H) = \overline{H}$ (рис. 32) представляет собой выражение вида

$$\bar{H} = \pi(\sigma(\bar{x}_3, \pi(\sigma(\bar{x}_1, \bar{x}_5), \sigma(x_1, \pi(x_4, x_5)))),$$
$$\sigma(\bar{x}_2, \bar{x}_4, \bar{x}'_5), \sigma(\pi(\bar{x}'_2, x_3), \bar{x}'_1, x'_5)).$$

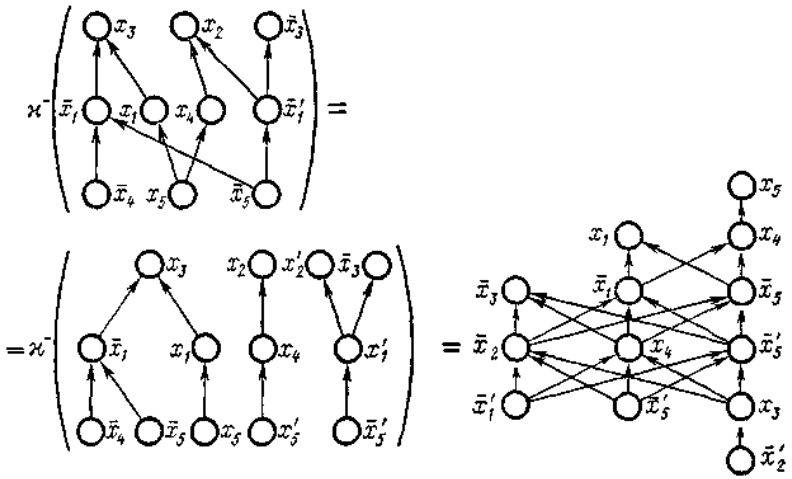


Рис. 32

При синтезе логических схем в функциональном базисе *B* необходимо определять результаты ко-операций вида $\varkappa^{f_{b_i}}$ $(b_i \in B)$, т. е.

$$\varkappa^{f_{b_i}}(H) = \{H_1, H_2, \ldots, H_{k_{вх}}\}, \quad (14)$$

где $k_{вх}$ - входной коэффициент элемента b_i.

Результат ко-операции $\varkappa^{f_{b_i}}$ можно получить, применяя соответственно ряд ко-операций вида $\varkappa^{\vee}$ и $\varkappa^{-}$ в порядке, обратном порядку следования операций $\vee$ и $^{-}$ в разложении операции f_{b_i} на $\vee$ и $^{-}$.

Например, результаты *ко-операции Шеффера* $\varkappa'(H)$ определяем с помощью коалгебры графов следующим образом.

Пусть необходимо найти результат $(H_1, H_2\}$ ко-операции $\varkappa'(H)$, т. е.

$$\varkappa'(H) = \{H_1, H_2\}. \quad (15)$$

Графы H_1, H_2 являются результатом $\varkappa'(H)$, если и только если

соответствующие им функции $f_1(H_1)$ и $f_2(H_2)$ связаны, согласно определению ко-операции, как

$$f(H) = f_1(H_1) | f_2(H_2).$$

Но, с другой стороны, $f(H) = \overline{f_1(H_1)} \vee \overline{f_2(H_2)}$. Отсюда, согласно определению *ко-операции* и уравнениям,

$$\varkappa^{\vee}(H) = \{H'_1, H'_2\},\ \varkappa^{-}(H'_1) = H_1,\ \varkappa^{-}(H'_2) = H_2,$$

т. е. в случае базиса Шеффера уравнение (15) можно свести к системе

$$\begin{cases} H_1 = \varkappa^{-}(H'_1 \in \varkappa^{\vee}(H)), \\ H_2 = \varkappa^{-}(H'_2 \in \varkappa^{\vee}(H)). \end{cases}$$

В дальнейшем будем обозначать *j*-ю компоненту результата ко-операции $\varkappa^{f_{b_i}}(H) = \{H_1, H_2, \ldots, H_j, \ldots, H_{k_{вх}}\}$ как $\varkappa^{f_{b_i}}(H)|_j$, т. е. $H_j = \varkappa^{f_{b_i}}(H)|_j$.

В рассматриваемом случае уравнение (15) сведено к системе вида

$$\begin{cases} H_1 = \varkappa^{-}(\varkappa^{\vee}(H)|_1), \\ H_2 = \varkappa^{-}(\varkappa^{\vee}(H)|_2). \end{cases}$$

При сведении задачи определения результата $\{H_1, H_2, \ldots, H_{k_{вх}}\}$ ко-операции $\varkappa^{f_{b_i}}(H)$ к задаче определения результатов ко-операций $\varkappa^{\vee}$ и $\varkappa^{-}$, в сущности, сводят структурное уравнение (14), разрешенное относительно $\{H_1, H_2, \ldots, H_{k_{вх}}\}$, к системе, состоящей из $k_{вх}$ структурных уравнений, причем каждое *j*-е ($j = 1, \ldots, k_{вх}$) структурное уравнение системы разрешено относительно H_j, т. е.

$$\begin{cases} H_1 = \varkappa^{\alpha_{11}}(\varkappa^{\alpha_{12}} \ldots (\varkappa^{\alpha_{1n_1}}(H)) \ldots), \\ H_2 = \varkappa^{\alpha_{21}}(\varkappa^{\alpha_{22}} \ldots (\varkappa^{\alpha_{2n_2}}(H)) \ldots), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ H_j = \varkappa^{\alpha_{j1}}(\varkappa^{\alpha_{j2}} \ldots (\varkappa^{\alpha_{jn_j}}(H)) \ldots), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ H_{k_{вх}} = \varkappa^{\alpha_{k_{вх}1}}(\varkappa^{\alpha_{k_{вх}2}} \ldots (\varkappa^{\alpha_{k_{вх}n_k}}(H)) \ldots), \end{cases} \qquad (16)$$

где $\varkappa^{\alpha_{ij}} \in \{\varkappa^{\vee}|_k, \varkappa^{-}\}$; $i = 1, \ldots, k_{вх}$; $j = 1, \ldots, n_j$; n_j - целое число, определяемое базисным элементом b_i для любого *j*.

Система (16) представляет собой $k_{вх}$ независимых друг от друга структурных уравнений. Не для каждого базисного элемента $b_i \in B$ можно свести решение структурного уравнения (14) к решению системы уравнений вида (16).

Для определения вида базисного элемента, для которого решение (14) сводится к решению (16), разобьем все множество функциональных базисов на два класса: *несвязных* и *связных базисов.*

Предварительно сопоставим каждому базисному элементу $b_i \in B$ граф

H_{b_i}, построенный следующим образом. Выразим булеву функцию f_{b_i}, реализуемую элементом b_i, через связки v и $^-$. Полученное выражение, в котором имеются только операции «дизъюнкция v» и «отрицание $^-$», представим в виде графа H_{b_i} согласно геометрической интерпретации l-местной операции дизъюнкции и операции отрицания (рис. 33).

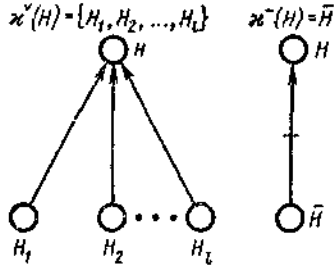


Рис. 33

Базис $B = \{b_i / i = 1, \ldots, n\}$ называется *несвязным,* если ни один из графов H_{b_i}, которые соответствуют функциям реализуемым f_{b_i}, элементами базиса, не содержит цикла.

Несвязными являются, например, базисы Вебба, Шеффера, импликативные, коимшшкативные (операцией *коимпликация* называем отрицание импликации).

Базис $B = \{b_i / i = 1, \ldots, n\}$ называется *связным,* если хотя бы один граф $H_{b_i} (i \in \{1, 2, \ldots, n\})$, который соответствует функции $f_{b_i} (i \in \{1, 2, \ldots, n\})$, реализуемой одним из элементов базиса, содержит цикл.

Например, решение уравнения $x^{\Delta}(H) = \{H_a, H_b, H_c, H_d\}$, соответствующего основному элементу УСЭППА (рис. 34), сводится к решению системы вида

$$\begin{cases} H_a = x^{-}(x^{\vee}(x^{-}(x^{\vee}(H)|_1))|_1), \\ H_b = x^{\vee}(x^{-}(x^{\vee}(x^{-}(x^{\vee}(H)|_1))|_2))|_1), \\ H_c = x^{-}(x^{\vee}(x^{-}(x^{\vee}(x^{-}(x^{\vee}(H)|_1))|_2))|_1, \\ H_d = x^{-}(x^{\vee}(x^{-}(x^{\vee}(H)|_2))|_2), \\ x^{-}(x^{\vee}(x^{-}(x^{\vee}(H)|_1))|_2) = x^{\vee}(x^{-}(x^{\vee}(H)|_2))|_1. \end{cases}$$

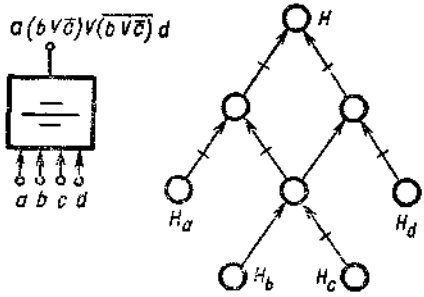


Рис. 34

Здесь Δ — знак операции, реализуемой основным элементом УСЭППА.

В данном случае определение результатов ко-операций при синтезе в связных базисах также сводится к решению системы структурных уравнений, но уже зависимых друг от друга.

При синтезе логических схем в функциональных базисах строим по реализуемой функции *f* структурный граф, который затем с помощью коалгебры графов преобразуем в функциональный. Полученный функциональный граф и является искомой логической схемой.

Предложим следующую процедуру преобразования структурного графа в функциональный. Предлагаемый ниже алгоритм рассмотрим на примере синтеза функционального графа, реализующего булеву функцию от трех переменных счетчика четности в базисе $\boldsymbol{B} = \{\rightarrow, 0\}$.

Алгоритм состоит из следующих шагов:

1. Структурный граф H_f, реализующий булеву функцию *f*, сопоставляется максимальному элементу графа H_b, соответствующего базисному элементу $b \in \boldsymbol{B}$.
2. Согласно графу H_b, определяющему функционально-структурные свойства элемента $b \in \boldsymbol{B}$, выполняются соответствующие операции разложения и инверсии над графом H_f.
3. В результате выполнения операций п. 2 определяются структурные графы H_{f_i}, соответствующие минимальным элементам графа H_b.

Заметим, что максимальный элемент графа H_b соответствует выходу базисного элемента *b*, минимальные — входам этого элемента.

Выполнение первых трех шагов для рассматриваемого примера иллюстрируется на рис. 35.

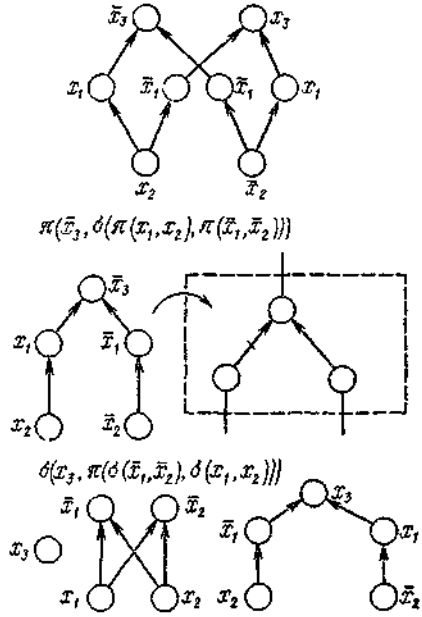


Рис.35

Для каждого из найденных графов H_{f_i} п. 1 — 3 выполняют до тех пор, пока не получат структурные графы, реализующие булевы функции, которые допустимы на входах логической схемы. Обычно такими функциями являются переменные x_i или переменные и их отрицания.

Если на i -м шаге преобразования структурного графа в функциональный было получено N графов, реализующих одну и ту же функцию с точностью до конъюнкций, тождественно равных нулю, то эти графы объединяются в $\left[N/k_{\text{вых}}\right]$ групп ([ ] —знак ближайшего целого числа;

$k_{\text{вых}}$ — выходной коэффициент (коэффициент разветвления) базисного элемента). Таким образом, при построении функционального графа учитывают коэффициент разветвления $k_{\text{вых}}$ базисных элементов. Окончательный результат преобразования структурного графа в функциональный в базисе $\{\rightarrow, 0\}$ показан на рис. 36.

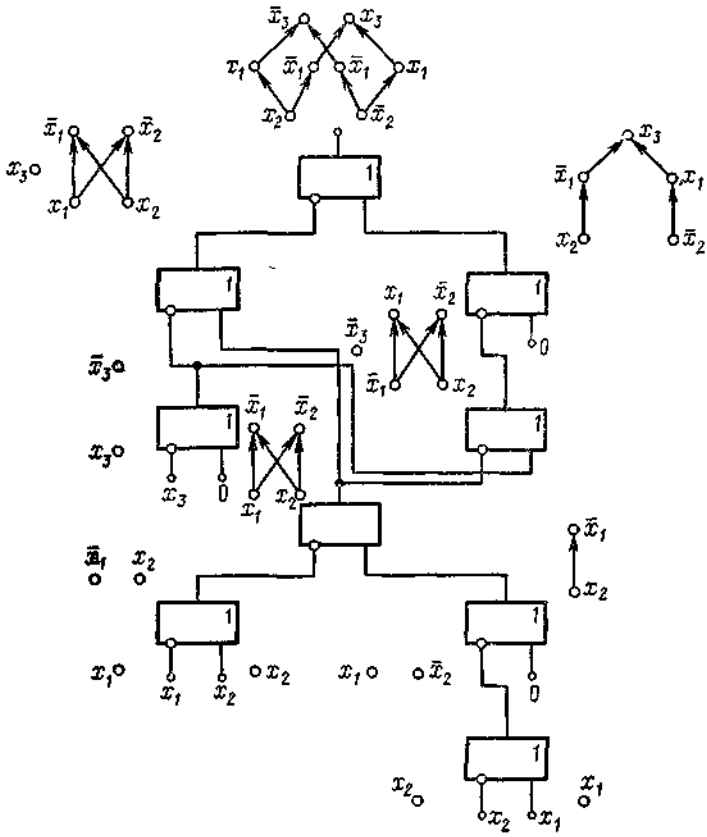


Рис. 36

Переход Н →*S* от структурного графа Н к переключательной схеме S в базисах первого и третьего топологических классов осуществляется тривиально: заменой вершин на ключевые элементы и дуг — на соединительные каналы с односторонней проводимостью.

При преобразовании *H*→ *S* в базисах второго топологического класса возможно появление лишних путей из-за двусторонней проводимости соединительных каналов, если структурный граф *H* содержит подграф Q_B, изображенный на рис. 37, а.

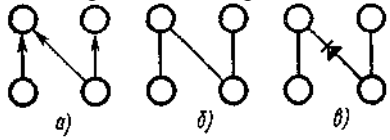


Рис. 37

При снятии ориентации дуг (рис. 37,б) несравнимые элементы становятся сравнимыми и появляется лишний путь, что выводит граф из класса эквивалентных, в смысле реализации заданной функции, графов. Для ликвидации лишних путей необходимо ориентировать диагональное ребро, помещая на него развязывающий диод (рис.37, в). Граф Q_B (рис. 37, *а)* является запрещенной фигурой преобразования H→ *S.* Следовательно, минимизация развязывающих диодов сводится к покрытию семантической таблицы, в которой запрещенными фигурами являются подграфы $Q_в$, а их компонентами — диагональные ребра, в которые помещаются развязывающие диоды.

Переход H→ S в базисах второго топологического класса осуществляется, как и в базисах первого класса, ориентацией диагональных ребер в запрещенных фигурах $O_в$. При преобразовании H →*S* в базисах этого класса абстракцией переключательной схемы (рис. 38, *а*) является *линейный граф* (рис. 38,б), дуги которого взвешены первичными термами или единицами, соответствующими развязывающим диодам. Линейный граф получается заменой вершин структурного графа H дугами с теми же весами при сохранении исходных путей.

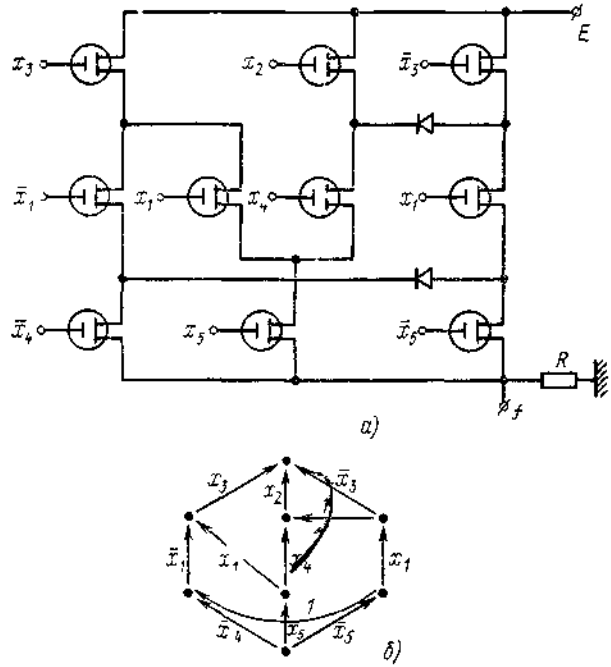


Рис. 38

При преобразовании $H \rightarrow S$ в базисах четвертого топологического класса возможно появление лишних путей не только из-за двусторонней проводимости соединительных каналов, но и в результате двусторонней проводимости элементов. При этом лишние пути возникают, если структурный граф содержит подхраф $Q_г$, гомеоморфный подграфу Q_B. Следовательно, дополнительной запрещенной фигурой преобразования H→S в базисах этого класса является подграф $Q_г$. Для ликвидации лишних путей, вызванных двусторонней проводимостью самих элементов, один из элементов, который находится на диагонали запрещенной фигуры, ориентируют, последовательно соединяя с ним развязывающий диод.

Переход H→S в базисах рассматриваемого класса осуществляется (как и в базисах первого класса) ориентацией диагональных ребер в запрещенных фигурах Q_B, $Q_г$.

При преобразовании H $\rightarrow S$ в базисах четвертого топологического класса абстракцией переключательных схем является линейный граф, ребра которого взвешены первичными термами или единицами, соответствующими развязывающим диодам, которые получаются заменой

вершин структурного графа H ребрами с теми же весами при сохранении исходных путей.

Сложность переключательных схем в базисах первого и третьего топологических классов равна сложности структурного графа. Во втором топологическом классе к этой сложности добавляется количество развязывающих диодов, определяемое распределением подграфов Q_B.

Сложность схем в четвертом классе равна сложности соответствующего структурного графа плюс количество развязывающих диодов, определяемое распределением фигур Q_B и $Q_г$, минус количество контуров длины 2, причем ребра каждого взвешены одним и тем же первичным термом. Каждый из таких контуров заменяется ключевым элементом с двусторонней проводимостью.

Таким образом, если считать только переключательные элементы, то сложность логических схем в топологических базисах равна сложности соответствующего структурного графа H. В функциональных базисах структурный граф H(*f*), определяющий реализуемую булеву функцию, преобразуется в логическую схему с помощью коалгебры графов.

Для описания функционирования автомат во времени с помощью булевых функций положим, что все поступающие сигналы могут изменяться во времени только дискретно. Выбрав достаточно короткие интервалы времени, считаем, что сигнал изменяется только на границе временных интервалов и не изменяется внутри самого интервала. Длительность интервала выбираем, исходя из следующих соображений. Разложим *временной булев ряд x(t)* в сумму *эталонных функций* вида

$$x(t) = \bigvee_i \alpha_i a_i(t).$$

Наиболее простые эталонные функции a_0, a_1, a_2, ... приведены на рис. 39, где T_0 — время, в течение которого анализируется работа автомата; при этом

$$T_i = T_0/2^i, \quad (17)$$

где T_i - период эталонной функции a_i. Согласно (7), длительность минимального интервала

$$T_k = T_0/2^{\gamma+1},$$

где γ — *глубина временного квантования.*

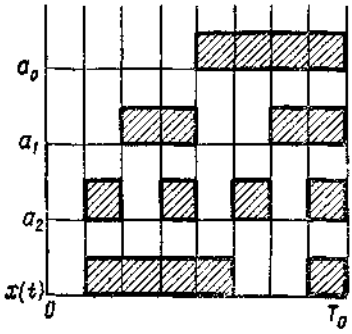


Рис. 39

Для любого автомата существует минимальное время $T_{мин}$ между двумя соседними переходами. Для того чтобы описать работу автомата во времени с достаточной степенью точности, необходимо, чтобы

$$T_{мин} \gg T_k \text{ или } \gamma \gg \log_2 \frac{T_0}{T_{мин}}. \quad (18)$$

Таким образом, для аналитического задания временных булевых рядов с данной степенью точности необходима глубина временного квантования

$$\gamma \gg \log_2 (T_0/T_{мин}).$$

Минимальный временной интервал $T_к$ будем называть *временным квантом.* Тогда каждому кванту соответствует конституента единицы булевой временной функции $\varphi(a_0, a_1, a_2, \ldots, a_\gamma)$, переменными которой являются эталонные функции $a_0, a_1, a_2, \ldots, a_\gamma$. Следовательно, любой временной булев ряд с заданной степенью точности можно задавать как булеву функцию от $a_0, a_1, \ldots, a_t$ и применять ее, как и обычную булеву функцию.

Например, временной булев ряд *x(t)* (рис. 39) можно представить в виде

$$x(t) = \bar{a}_0(a_1 \vee a_2) \vee a_0(\bar{a}_1\bar{a}_2 \vee a_1a_2).$$

Каждый реальный элемент, входящий в синтезируемую логическую схему, обладает постоянной времени. Следовательно, в практических логических схемах имеют место *переходные процессы,* которые необходимо учитывать. Рассмотрим применение понятия производной для исследования переходных процессов в схемах.

Временной производной $\frac{\partial x}{\partial t}$ от булевой переменной *x(t)* называется

$$\frac{\partial x}{\partial t} = x(t) \oplus x(t \mp 1), \quad (19)$$

где *x (t)* - временная булева последовательность, принимающая значения 0,1 в моменты времени 0, 1, 2, ... Для определенности возьмем

в (19) знак « —». Временная производная показывает изменение сигнала во времени. Будем рассматривать периодические булевы ряды. Проиллюстрируем понятие временной производной на следующем примере. Ести переменная $x(t)$ во времени $t = 0, 1, 2, 3, 4, 5$ изменяется соответственно $x(t) = 0, 1, 0, 1, 1, 1$, то временная производная изменяется как $\frac{\partial x}{\partial t} = 1, 1, 1, 1, 0, 0$.

Рассмотрим логическую схему с двумя входами, которые одновременно переключаются. Для определенности будем рассматривать переход (0, 0)→ (1, 1). Условие одновременного переключения входных каналов x_1 и x_2 является идеальным. На самом деле с вероятностью, почти равной единице, изменяется сначала один, а через некоторое время другой вход. В зависимости от порядка их переключения путь от (0, 0) к (1, 1) может пройти через (1, 0) или через (0, 1). Рассмотрим два случая:

1. Функции $f(0, 0)$ и $f(1, 1)$ не равны. Выходной сигнал до и после перехода различен. Тогда во время промежуточных состояний может иметь место либо «старое» значение сигнала, либо «новое». Если в схеме отсутствуют всплески, то никаких ложных сигналов принципиально не может быть.
2. Функции $f(0, 0)$ и $f(1, 1)$ равны. Выходной сигнал до и после перехода одинаков, т. е. переход на выходе «не чувствуется». Но если выходной сигнал во время промежуточных состояний отличается от $f(0, 0)$ ($f(1, 1)$), то во время переключения может появиться ложный сигнал, переход — критический. Это явление будем называть *риском* в логической схеме. Путь, по которому сигнал перейдет из состояния (0, 0) в состояние (1, 1), зависит от физических параметров сигналов схемы и от случая, поскольку параметры носят статистический характер.

Следуя Бохману, приведем необходимые и достаточные условия критического перехода. Для того чтобы переход из состояния (σ_1, σ_2) в состояние $(\bar{\sigma}_1, \bar{\sigma}_2)$ $(\sigma_1, \sigma_2 = 0, 1)$ был критическим, необходимо и достаточно, чтобы:

а) $f(\sigma_1, \sigma_2) = f(\bar{\sigma}_1, \bar{\sigma}_2)$;

б) $f(\bar{\sigma}_1, \sigma_2) \neq f(\sigma_1, \sigma_2)$ **или** $f(\sigma_1, \bar{\sigma}_2) \neq f(\sigma_1, \sigma_2)$.

Эти условия легко сформулировать с помощью производных. В случае критического перехода имеем:

а) одновременное переключение обоих входов: $\frac{\partial x_1}{\partial t} \frac{\partial x_2}{\partial t} = 1$;

б) значение выходного сигнала до и после перехода одинаковое: $\frac{\partial^2 f}{\partial(x_1 x_2)} = 0$;

в) переключение только одного из входов ведет к переключению выхода: $\frac{\partial f}{\partial x_1} \vee \frac{\partial f}{\partial x_2} = 1$.

Следовательно, ошибка выражается следующим образом:

$$\Delta f = \frac{\partial x_1}{\partial t}\frac{\partial x_2}{\partial t}\left(\frac{\partial f}{\partial x_1} \vee \frac{\partial f}{\partial x_2}\right)\frac{\overline{\partial^2 f}}{\partial(x_2 x_1)}, \quad \textbf{(20)}$$

где Δf— *функция* ошибки.

В выражении (20) конъюнкция $\frac{\partial x_1}{\partial t}\frac{\partial x_2}{\partial t}$ определяет свойства сигнала, в дальнейшем будем называть ее *сигнальной* частью. Конъюнкция $\left(\frac{\partial f}{\partial x_1} \vee \frac{\partial f}{\partial x_2}\right)\frac{\partial^2 f}{\partial(x_1 x_2)}$ определяет свойства реализуемой функции; далее будем называть ее *функциональной* частью формулы, определяющей ошибку.

Одним из способов повышения производительности ЭВМ является увеличение рабочей частоты применяемых элементов. Современные молектронические элементы имеют рабочую частоту тысячи мегагерц. При работе логических схем задержка в цепи элементов становится сравнимой с рабочим периодом работы схемы, что приводит к необходимости учитывать ложные сигналы, определяемые выражением (20).

Ошибки не возникает, если $\Delta f = 0$, для чего необходимо наложить ограничения на сигнальную или на функциональную часть формулы. Рассмотрим случай логических схем, у которых одновременно переключаются три входных канала. Согласно (20), ошибка имеет место при переключении произвольных двух входов. Следовательно, в функции ошибок при переключении трех входов x_a, x_b, x_c содержатся следующие члены:

$$\frac{\partial x_a}{\partial t}\frac{\partial x_b}{\partial t}\left(\frac{\partial f}{\partial x_a} \vee \frac{\partial f}{\partial x_b}\right)\frac{\overline{\partial^2 f}}{\partial(x_a x_b)};$$

$$\frac{\partial x_a}{\partial t}\frac{\partial x_c}{\partial t}\left(\frac{\partial f}{\partial x_a} \vee \frac{\partial f}{\partial x_c}\right)\frac{\overline{\partial^2 f}}{\partial(x_a x_c)};$$

$$\frac{\partial x_b}{\partial t}\frac{\partial x_c}{\partial t}\left(\frac{\partial f}{\partial x_b} \vee \frac{\partial f}{\partial x_c}\right)\frac{\overline{\partial^2 f}}{\partial(x_b x_c)}.$$

В формуле, определяющей условия возникновения ошибок при переключении трех входов, содержится еще один член, который и учитывает одновременное переключение трех входов:

$$\frac{\partial x_a}{\partial t}\frac{\partial x_b}{\partial t}\frac{\partial x_c}{\partial t}\left(\frac{\partial^2 f}{\partial (x_a x_b)} \vee \frac{\partial^2 f}{\partial (x_a x_c)} \vee \frac{\partial^2 f}{\partial (x_b x_c)}\right)\overline{\frac{\partial^3 f}{\partial (x_a x_b x_c)}}.$$

Следовательно, функция ошибки при переключении трех входов имеет вид

$$\Delta f = \frac{\partial x_a}{\partial t}\frac{\partial x_b}{\partial t}\left(\frac{\partial f}{\partial x_a} \vee \frac{\partial f}{\partial x_b}\right)\overline{\frac{\partial^2 f}{\partial (x_a x_b)}} \vee$$

$$\vee \frac{\partial x_a}{\partial t}\frac{\partial x_c}{\partial t}\left(\frac{\partial f}{\partial x_a} \vee \frac{\partial f}{\partial x_c}\right)\overline{\frac{\partial^2 f}{\partial (x_a x_c)}} \vee$$

$$\vee \frac{\partial x_b}{\partial t}\frac{\partial x_c}{\partial t}\left(\frac{\partial f}{\partial x_b} \vee \frac{\partial f}{\partial x_c}\right)\overline{\frac{\partial^2 f}{\partial (x_b x_c)}} \vee$$

$$\vee \frac{\partial x_a}{\partial t}\frac{\partial x_b}{\partial t}\frac{\partial x_c}{\partial t}\left(\frac{\partial^2 f}{\partial (x_a x_b)} \vee \frac{\partial^2 f}{\partial (x_a x_c)} \vee \frac{\partial^2 f}{\partial (x_b x_c)}\right)\overline{\frac{\partial^3 f}{\partial (x_a x_b x_c)}}. \quad (21)$$

Обозначим функциональную часть функции ошибок, которая входит в одну и ту же конъюнкцию с $\frac{\partial x_{a_1}}{\partial t}\frac{\partial x_{a_2}}{\partial t}\ldots\frac{\partial x_{a_k}}{\partial t}$, как $F(x_{a_1}, x_{a_2}, \ldots, x_{a_k})$.

Тогда (21) можно переписать в виде

$$\Delta f = \frac{\partial x_a}{\partial t}\frac{\partial x_b}{\partial t}F(x_a, x_b) \vee \frac{\partial x_a}{\partial t}\frac{\partial x_c}{\partial t}F(x_a, x_c) \vee$$

$$\vee \frac{\partial x_b}{\partial t}\frac{\partial x_c}{\partial t}F(x_b, x_c) \vee \frac{\partial x_a}{\partial t}\frac{\partial x_b}{\partial t}\frac{\partial x_c}{\partial t}F(x_a, x_b, x_c).$$

Обобщая функцию ошибок на случай переключения и входов в логической схеме, получаем

$$F(x_{a_1}, x_{a_2}, \ldots, x_{a_n}) = \left(\bigvee_{\substack{i_1, i_2, \ldots, i_{n-1} = x_{a_1}, x_{a_2}, \ldots, x_{a_n} \\ i_1 \neq i_2, \ldots, i_{n-2} \neq i_{n-1}}} \frac{\partial^{n-1} f}{\partial (i_1, i_2, \ldots, i_{n-1})}\right)\overline{\frac{\partial^n f}{\partial (x_{a_1}, x_{a_2}, \ldots, x_{a_n})}}. \quad (22)$$

и функция ошибок имеет вид

$$\Delta f = \bigvee \frac{\partial i_1}{\partial t}\frac{\partial i_2}{\partial t}F(i_1, i_2) \vee \bigvee \frac{\partial i_1}{\partial t}\frac{\partial i_2}{\partial t}\frac{\partial i_3}{\partial t}F(i_1, i_2, i_3) \vee \ldots \vee$$

$$\vee \frac{\partial x_{a_1}}{\partial t}\frac{\partial x_{a_2}}{\partial t}\ldots\frac{\partial x_{a_n}}{\partial t}F(x_{a_1}, x_{a_2}, \ldots, x_{a_n}), \; i_1, i_2, \ldots = a_1, a_2, \ldots, a_n. \quad (23)$$

Из формулы (23) следует, что при увеличении числа переключаемых входных каналов вероятность критического перехода увеличивается. При переключении трех каналов на входе схемы вероятность риска не меньше 75%. Функция ошибок Δf носит статистический характер.

7.7. Моделирование автоматных систем сетями Петри

В связи со все более широким использованием параллельных и распределенных вычислительных систем особую актуальность приобретают дискретные структуры, представляющие параллельные процессы. Аппаратом описания сложных систем взаимодействующих процессов являются формальные системы типа *сетей Петри,* моделирующие динамические свойства систем.

Формализм сетей Петри общего вида основан на понятии комплекта, являющемся в некотором роде обобщением понятия множества. Как и множество, *комплект* — это набор элементов, но всякий элемент может входить в него более одного раза. Иначе говоря, отношение включения, связывающее элементы и множества, заменяется на *фунщию числа экземпляров* элемента в комплекте, которая обозначается как $\#(x, B)$ (читается: «число *x* в комплекте *B*»).

Множество — частный случай комплекта.

Многие понятия теории множеств распространяются и на комплекты. Так, *пустой комплект* аналогичен пустому множеству. *Мощность комплекта* есть общее число экземпляров элементов в комплекте. Комплект *A* включен в комплект *B* (является подкомплектом), если для всякого $x\ \#(x, A) \leqslant \#(x, B)$. С помощью функции $\#$ легко определяются *операции* над комплектами:

для *объединения комплектов A* и

$$B\ \#(x, A \cup B) = \max(\#(x, A), \#(x, B));$$

для *пересечения комплектов*

$$A \text{ и } B\ \#(x, A \cap B) = \min(\#(x, A), \#(x, B));$$

для *суммы комплектов* A и B $\#(x, A + B) = \#(x, A) + \#(x, B)$;

для *разности комплектов*

$$A \text{ и } B\ \#(x, A - B) = \#(x, A) - \#(x, A \setminus B).$$

Если M — множество, то M^n — множество всех таких комплектов, построенных из элементов M, что $\#(x, B) \leqslant n$, $B \in M^n$; M^∞ — множество всех комплектов, построенных из элементов *M* без ограничения на число экземпляров элемента в комплекте.

Сеть Петри — это четверка $C = (P, T; I, O)$, где *P* — конечное множество *позиций, T*— конечное множество *переходов,* $I: T \to P^\infty$ —

входная функция, отображающая переходы в комплекты позиций, $O: T \to P^{\infty}$ — выходная функция, отображающая переходы в комплекты позиций. Графически сеть Петри изображают в виде мультиграфа с вершинами двух видов: **кружки соответствуют позициям, планки — переходам.** Функции *I* VL *O* представляются дугами (рис. 40).

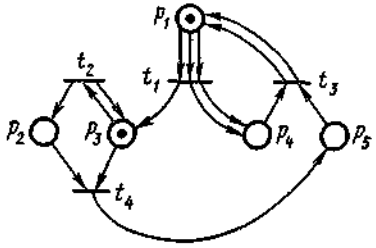


Рис.40

Позиции, дуги из которых ведут в переход t_j, называются *входными* для t_j; аналогично, позиции, в которые ведут дуги из перехода t_j, называются *выходными* для t_j. Множество входных позиций обозначают $I(t_j)$, а выходных — $O(t_j)$. В сети Петри, изображенной на рис. 40, $I(t_1) = \{p_1, p_1, p_1\}$, $O(t_1) = \{p_3, p_4, p_4\}$. Функции *I* и *O* удобно обобщить и на отображение из позиций в комплекты переходов $(P \to T^{\infty})$, что позволяет обозначать множества входных и выходных переходов позиции p_i, определяемые аналогично множествам входных и выходных позиций перехода, соответственно как $I(p_i)$ и $O(p_i)$. В сети Петри, изображенной на рис. 40, $I(p_3) = \{t_1, t_2\}$, $O(p_3) = \{t_2, t_4\}$.

Введенные понятия относятся к статической структуре сети Петри. Динамические свойства сети Петри определяются с помощью понятия **маркировки.** *Маркировка* μ сети Петри C = *(P, T, I, O)* — это функция, отображающая множество позиций *P* в множество неотрицательных целых чисел *N*. Маркировка изображается с помощью помещаемых внутрь позиций *фишек* (точек). Так, маркировка сети Петри, приведенной на рис. 40, определяется как $\mu(p_1) = \mu(p_3) = 1$, $\mu(p_2) = \mu(p_4) = \mu(p_5) = 0$.

Удобнопредставлять маркировку как *n*-вектор $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ (где $n = |P|$), каждый элемент которого μ_i есть $\mu(p_i)$, а также как комплект μ, в который входят позиции сети $p_i \in P$ и $\#(p_i, \mu) = \mu(p_i)$. Сеть Петри *C* с определенной в ней маркировкой μ называется *маркированной* сетью Петри.

Маркировка сети может изменяться в результате *запуска переходов.* Переход t_j маркированной сети Петри C с маркировкой μ называется

разрешенным, если $I(t_j) \subseteq \mu$, т. е. в каждой входной позиции t_j находится не меньше μ фишек, чем из этой позиции исходит дуг в t_j. Всякий разрешенный переход может запуститься. В результате *запуска* перехода t_j маркировка сети μ изменяется на новую: $\mu' = \mu - I(t_j) + O(t_j)$, т. е. из всякой входной позиции p_i перехода t_i удаляется столько фишек, сколько дуг ведет из p_i в t_j, а в каждую выходную позицию p_k помещается столько фишек, сколько дуг ведет из t_j в p_k. Последовательность запусков переходов называется *выполнением сети Петри.*

Рассмотрим выполнение сети Петри, изображенной на рис. 40. В *начальной маркировке* разрешен только переход t_2. При его запуске фишка удалится из p_3, а затем в позиции p_2 и p_3 добавится по фишке, т. е. в результате запуска в новой маркировке μ' появится фишка еще и в p_2. Теперь становятся разрешенными переходы t_2, t_4. Поскольку запуститься может любой разрешенный переход, предположим, что запускается переход t_4. После его запуска из позиции p_2 и p_3 фишки удалятся, а в позиции p_5 появится одна фишка. В получившейся маркировке μ" не разрешен ни один переход. На этом выполнение сети Петри заканчивается.

Рассмотрим маркировку μ сети Петри $C = (P, T, I, O)$. Маркировка μ' называется *непосредственно достижимой* из μ, если найдется такой переход $t_j \in T$, разрешенный в μ, что при его запуске получается маркировка μ'; в этом случае пара (μ, μ') принадлежит *отношению непосредственной достижимости,* определенному на P^∞. Транзитивное замыкание этого отношения называется *отношением достижимости.* Маркировки μ' такие, что (μ, μ') принадлежат отношению достижимости, называются *достижимыми* из μ. Множество достижимых из μ маркировок сети Петри С называется *множеством достижимости* и обозначается $R(C, \mu)$.

Интерпретация сетей Петри основана на понятиях *условия* и *события.* Состояние системы описывается совокупностью условий. Функционирование системы состоит в осуществлении последовательности определенных действий, т. е. событий. Для возникновения события необходимо выполнение некоторых условий, называемых *предусловиями.* Возникновение события может привести к нарушению предусловий и к выполнению некоторых других условий, называемых *постусловиями.* В сети Петри условия моделируются позициями, события — переходами. Предусловия события представляются входными позициями соответствующего перехода, постусловия — выходными позициями. Возникновение события моделируется запуском перехода. Выполнение условий представляется наличием фишек в соответствующих позициях, невыполнение — их отсутствием.

Рассмотрим, например, простую вычислительную систему, последовательно обрабатывающую задания, которые поступают во входную очередь. Когда процессор свободен и во входной очереди имеется задание, оно обрабатывается процессором, затем выводится. Эту систему можно промоделировать сетью Петри, изображенной на рис. 41.

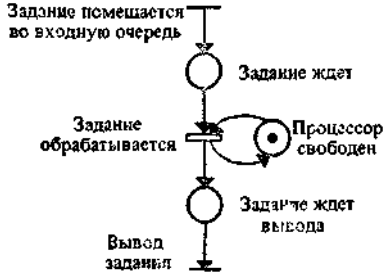


Рис. 41

Установим, какие особенности систем учитывают сети Петри. Это прежде всего *асинхронность*. В сети Петри отсутствует понятие времени. Время возникновения событий никак не указывается, но тем не менее структура сети Петри устанавливает частичный порядок возникновения событий. Далее, поскольку возникновение событий представляется запуском переходов, предполагается, что события происходят *мгновенно.* Если же моделируемое событие имеет отличную от нуля длительность, как, например, событие «задание обрабатывается» (рис. 41), и это существенно, то оно представляется в виде двух мгновенных событий типа «начало события», «конец события» и условия «событие происходит» (рис. 42).

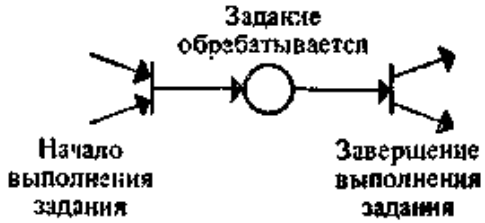


Рис.42

Кроме того, считается, что события происходят *неодновременно* (мгновенные события не могут происходить одновременно). Действительно, если допустить одновременное возникновение некоторых событий *i* и *j,* которым в сети Петри соответствуют

переходы t_i и t_j, то можно ввести дополнительный переход t_{ij} с $I(t_{ij}) = I(t_i) + I(t_j)$, $O(t_{ij}) = O(t_i) + O(t_j)$, интерпретирующийся как одновременное возникновение событий *i* и *j*. В этом случае переходы можно запускать последовательно.

Другим важным свойством сетей Петри как инструмента моделирования является их способность представлять *параллелизм* и *конфликтные ситуации.* Параллелизм двух событий представляется двумя разрешенными переходами, множества входных позиций которых не пересекаются (рис. 43), конфликт — переходами с общей входной позицией (рис. 44).

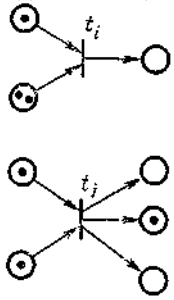


Рис. 43

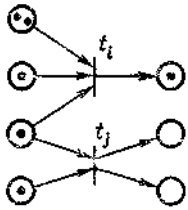


Рис. 44

Сети Петри используют в основном как формальный аппарат при моделировании систем, которым присущ параллелизм. Если рассматривать процесс проектирования в целом, то возможны два принципиально различных подхода к использованию сетей Петри. В первом система моделируется сетью Петри, которая преобразуется по определенным правилам к некоторому «оптимальному» виду. Полученная сеть Петри преобразуется в проект системы. Предполагается, что он также оптимальный. Здесь сети Петри применяют непосредственно при проектировании. Этот подход, однако, имеет трудности, связанные с неоднозначностью обратного преобразования — сети Петри в проект системы,— что подвергает сомнению оптимальность получаемого проекта. Во втором, более общепринятом, подходе сначала с помощью обычных средств создается проект системы и по нему строится модель в виде сети Петри. Затем исследуются свойства полученной сети и делаются выводы о свойствах и характеристиках проекта. Если они неудовлетворительны, то полученные в результате исследования сети Петри данные используют для модификации проекта. Модифицированный проект снова преобразуют в сеть Петри, и цикл повторяется. Этот

процесс заканчивается, когда сеть Петри будет обладать необходимыми свойствами.

Рассмотрим, какие свойства сети Петри как модели системы могут интересовать проектировщика. Одно из важнейших свойств — безопасность. Позиция сети Петри называется *безопасной,* если число фишек в ней никогда не превышает 1. Маркированная сеть Петри *безопасна,* если безопасны все ее позиции. Это свойство очень важно при интерпретации позиций как простых условий: если в позиции есть фишка, то условие выполняется, если нет, то не выполняется. Если интерпретация фишек более сложная, например количество фишек показывает число информационных единиц, то может быть интересен вопрос, ограничено ли число фишек в данной позиции, и если да, то какова граница. Так приходим к свойству ограниченности. Позиция называется *k-ограниченной,* если число фишек в ней в любой достижимой маркировке не превышает целого *k.* Маркированная сеть Петри называется *k-ограниченной,* если ее позиции являются *k*-ограниченными. В сети Петри, приведенной на рис. 45, позиции p_1, p_2 являются безопасными, позиция p_3 - 2-ограниченная, вся сеть — 2-ограниченная.

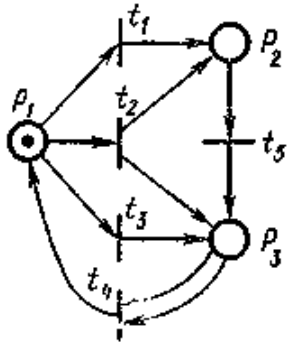


Рис. 45

В случае когда фишки интерпретируются как некоторые ресурсы, они не должны ни создаваться, ни уничтожаться. Иначе говоря, **в сети должен действовать закон сохранения**. Маркированная сеть Петри называется *строго сохраняющей,* если мощность маркировки (как комплекта позиций) постоянна. В общем случае фишка может интерпретироваться как некоторое число элементарных ресурсов, причем это число меняется от позиции к позиции. Введем понятие *взвешивания позиций: сектор* $\mathbf{W} = (w_1, w_2, \ldots, w_n)$, где w_i — вес

позиции p_i. Сеть Петри называется *сохраняющей по отношению к вектору взвешивания* **W**, если скалярное произведение вектора **W** и маркировки (рассматриваемой как вектор) постоянно; сеть Петри — *сохраняющая,* если она является сохраняющей по отношению к вектору взвешивания **W**, все элементы которого положительны. Рассмотренные до сих пор свойства относятся как к последовательным, так и к параллельным системам. Но при переходе от последовательных систем к параллельным возникают принципиально новые трудности: возможность *тупиковых ситуаций. Тупиком* в сети Петри называется множество переходов, которые в некоторой достижимой маркировке μ' и в последующих достижимых из μ' маркировках не разрешены. Свойство возможности возникновения тупиков в системе моделируется свойством активности в сетях Петри. Переход t_j называется *активным,* если он не входит ни в какой тупик. Переход называется *пассивным,* если он не разрешен ни в какой достижимой маркировке. При детальном исследовании активности сети Петри используют также понятие **уровней активности.** Переход t_j обладает *активностью: уровня 0,* если он не может быть запущен (пассивный); *уровня 1,* если он потенциально запустим, т. е. существует достижимая маркировка, в которой он разрешен; *уровня 2,* если для любого целого k существует последовательность запусков переходов, в которой он присутствует не менее k раз; *уровня 3,* если существует бесконечная последовательность запусков, в которой он присутствует неограниченно часто; *уровня 4*, если он потенциально запустим из всякой достижимой маркировки (т. е. активен). В сети Петри, изображенной на рис. 46, *а,* переход t_3 активен, переходы t_1, t_2, t_4, t_6 имеют уровень активности 3; переход t_6 пассивен.

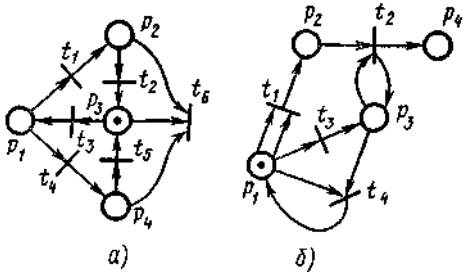


Рис. 46

В сети Петри, приведенной на рис. 46, *б,* переход t_4 пассивен, переход t_3 обладает активностью уровня *1*, переход t_2 — активностью уровня *2*, а t_1 — активностью уровня *3.*

Одной из наиболее важных задач анализа сетей Петри является *задача достижимости:* достижима ли маркировка μ' из начальной маркировки и данной сети Петри? Важность этой задачи обусловлена тем, что маркировка служит интерпретацией состояния системы. Решение задачи достижимости позволит определить, достижимо ли определенное состояние, будь оно «хорошим» или «плохим» для системы.

Описанные свойства и соответствующие задачи анализа сетей Петри — наиболее общие, хотя и не охватывают все множество вопросов, которые могут возникнуть при анализе сетей Петри. Для решения задач анализа имеется два основных подхода. Первый основан на построении дерева достижимости. *Дерево достижимости* — это ориентированное корневое дерево, вершинам которого соответствуют возможные маркировки, дугам — переходы. Корневой вершине соответствует начальная маркировка. Из каждой вершины исходят дуги, соответствующие разрешенным переходам. Построение дерева осуществляется последовательно начиная с корневой вершины; на каждом шаге строится очередной ярус дерева. Например, дерево достижимости для сети Петри, изображенной на рис. 46, *а,* после трех шагов имеет вид, приведенный на рис. 47, *а* (маркировки представлены векторами).

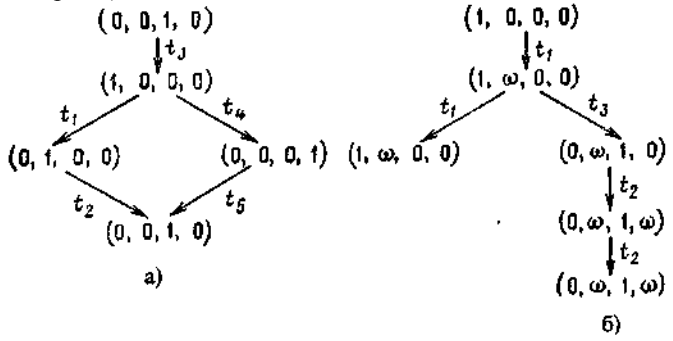


Рис. 47

Очевидно, что если не использовать при построении дерева определенные соглашения, то активные (даже ограниченные) сети Петри будут иметь бесконечное дерево достижимости.

Назовем вершины (и соответствующие маркировки), построенные на очередном шаге алгоритма, *граничными.* Если в какой-либо граничной маркировке нет разрешенных переходов, то будем называть такую маркировку *терминальной.* Если какая-либо граничная вершина имеет маркировку, уже существующую в дереве, то назовем ее *дублирующей.*

Для терминальных и дублирующих вершин не будем строить исходящих из них дуг. Это обеспечивает конечность дерева достижимости для ограниченной сети Петри (например, рис. 46, *а,* 47, *а).* Для неограниченных сетей требуется как-то обозначить неограниченное число фишек в позиции. Пусть ω обозначает такое число, причем $\omega + a = \omega$, $\omega - a = \omega$, $a < \omega$, $\omega \leqslant \omega$, где a — произвольное целое положительное число. Будем использовать при построении дерева достижимости следующее правило. Пусть граничная вершина μ не является терминальной или дублирующей. Для каждого разрешенного перехода t_j в маркировке μ построим дугу, исходящую из μ . Дугу будем помечать переходом t_j. Маркировка μ' новой вершины определяется следующим образом. Если $\mu(p_i) = \omega$, то $\mu'(p_i) = \omega$. Если на пути от корневой вершины к μ существует вершина μ" такая, что в результате запуска в μ перехода t_j число фишек в каждой позиции не меньше чем в μ", а в позиции p_i строго больше, то $\mu'(p_i) = \omega$. В противном случае $\mu'(p_i)$ — число фишек в позиции p_i, получающееся после запуска t_j из μ (рис. 47, б).

Теорема 4. *Дерево достижимости любой сети Петри конечно.*

Доказательство этого утверждения основано на свойствах ω и на правилах введения этого символа в маркировку граничных вершин.

Метод анализа, основанный на дереве достижимости, позволяет определить свойства безопасности, ограниченности, сохранения, исследовать свойства активности и достижимости.

Сеть Петри ограниченна тогда и только тогда, когда символ ω отсутствует в дереве достижимости. Кроме того, положение символа ω показывает, какие позиции неограниченны. Если символ ω отсутствует в дереве, то число достижимых маркировок конечно и все вопросы анализа можно решить простым перебором. В частности, для нахождения границы маркировки данной позиции p_i нужно найти наибольшее значение i-й компоненты среди всех вершин дерева. Если эта граница не превышает 1, то позиция безопасная.

Для установления того, является ли сеть Петри сохраняющей по отношению к некоторому вектору взвешивания $\mathbf{W} = (w_1, w_2, \ldots, w_n)$, необходимо решить следующую систему линейных уравнений с ограничениями:

$$\begin{cases} \sum\limits_{i=1}^{n} w_i \mu_j(p_i) = s \quad \text{при } j = (1, \ldots, k), \\ w_i \geqslant 0 \quad \text{при } i = (1, \ldots, n), \end{cases}$$

где k — число вершин дерева достижимости, которым соответствуют различные маркировки. (Очевидно, что если имеется $\mu_j(p_i) = \omega$, то $w_i = 0$.)

Возможность решения задач активности и достижимости ограничена существованием символа ω, скрывающего конкретную информацию о числе фишек. Например, если в сеть Петри, изображенную на рис. 46,б, ввести две дуги (t_1, p_2), (p_2, t_2), то полученная сеть Петри будет иметь то же дерево достижимости, что и первоначальная. При этом в новой сети Петри в позиции p_2 может быть только четное число фишек, тогда как в первоначальной сети — любое, т. е. множества достижимости этих сетей Петри не совпадают. Можно привести разные сети с различными свойствами активности, но имеющие одно дерево достижимости.

Тем не менее, хотя дерево достижимости и не дает полной информации о свойствах достижимости и активности, в некоторых случаях оно позволяет ответить на вопросы по достижимости и активности. Например, если в нем имеется терминальная вершина, то сеть Петри не активна. При решении задачи достижимости может оказаться, что маркировка μ' присутствует в дереве достижимости (положительный ответ) или что маркировка μ' не покрывается никакой вершиной дерева достижимости, т. е. $\mu'' \not\geqslant \mu$ для всех вершин μ'' (отрицательный ответ).

Другой подход к анализу сетей Петри называется *матричным* и основан на их матричном представлении. Введем матрицы D^- и D^+, столбцам которых соответствуют позиции, строкам — переходы, и $D^-(j, i) = \#(p_i, I(t_j))$, $D^+(j, i) = \#(p_i, O(t_j))$. Пусть $e(j)$ - вектор-строка, компоненты которого соответствуют переходам и все равны нулю, за исключением j-й, которая равна 1. Тогда переход t_j разрешен, когда $\mu \geqslant e(j) \cdot D^-$ (μ рассматривается как вектор), а результатом запуска t_j из μ является

$$\mu' = \mu - e(j) D^- + e(j) D^+ = \mu + e(j) \cdot (D^+ - D^-) = \mu + e(j) \cdot D,$$

где $D = D^+ - D^-$ — *составная матрица изменений.*

Маркировка μ', получающаяся из μ в результате запуска последовательности $\sigma = t_{j_1} t_{j_2} \ldots t_{j_k}$, определяется как

$$\mu' = \mu + e(j_1) D + e(j_2) D + \ldots + e(j_k) D =$$

$$= \mu + (e(j_1) + \ldots + e(j_k)) D = \mu + f(\sigma) D,$$

где $f(\sigma) = e(j_1) + \ldots + e(j_k)$ - *вектор запуска, i-я* компонента которого равна числу запусков t_{j_i} в σ.

Если маркированная сеть Петри является сохраняющей по отношению к некоторому вектору взвешивания **W** (**W** — вектор-столбец), то $\mu W = \mu' W$ для любой $\mu' = R(C, \mu)$. Так как $\mu' = \mu + f(\sigma) D$, то $f(\sigma) DW = 0$. Поскольку это верно для всех $f(\sigma)$, имеем $DW=0$.

Следовательно, сеть Петри является сохраняющей по отношению к

некоторому вектору взвешивания тогда и только тогда, когда имеется такой вектор **W**, что $DW = 0$. Это уравнение позволяет отыскать и сам вектор взвешивания **W**.

Если маркировка μ' достижима из начальной маркировки μ сети Петри, то должно существовать целое неотрицательное решение уравнения $\mu' = \mu + xD$, решением которого будет $x = f(\sigma)$.

Исследуем задачу достижимости для сети Петри, изображенной на рис. 46, *б,* с начальной маркировкой (1, 0, 0, 0) для маркировки μ' = (0, 2, 1, 2). Уравнение $\mu' = \mu + xD$ принимает вид

$$(0,\ 2,\ 1,\ 2) = (1,\ 0,\ 0,\ 0) + x \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}.$$

и имеет решение x = (4, 2, 1, 0), соответствующее последовательности запусков переходов $t_1t_1t_1t_1t_3t_2t_2$.

Матричный подход к анализу сетей Петри, как и подход, основанный на дереве достижимости, не позволяет в общем случае решить задачу активности и достижимости. Проблемы матричного анализа заключаются в том, что, во-первых, вектор запуска, получаемый при решении уравнения, не дает информации о порядке запуска переходов и, во-вторых, может соответствовать неразрешенной последовательности запусков. В настоящее время установлено, что задачи достижимости и активности *эквивалентны,* но неизвестно, разрешимы ли они вообще, т. е. нет ни алгоритма, позволяющего решить эти задачи, ни доказательства его отсутствия.

Рассмотрим применение методов анализа сетей Петри, моделирующих практические системы.

Специализированная параллельная система для реализации итерационных вычислительных процессов состоит из совокупности процессорных элементов (ПЭ) и модулей памяти (памяти данных (МПД) и памяти команд (МПК)), связанных в кольцо (рис. 48).

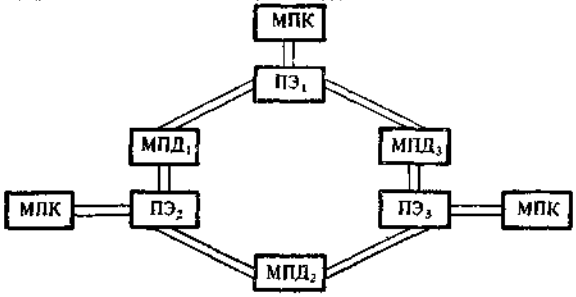


Рис. 48

ПЭ работает в двух режимах. В первом случае он захватывает оба смежных МПД, используя левый для извлечения исходных данных новой итерации, правый — для выборки результатов предыдущей итерации. По завершении итерации он помещает результат в правый МПД и освобождает оба МПД. В другом режиме ПЭ работает с внутренними данными. Режим указывается командой, считываемой из МПК. Рассмотрим два варианта реализации устройств управления ПЭ. В первом варианте захват МПД при осуществлении итерации производится последовательно. ПЭ может находиться в следующих состояниях: «обработка внутренних данных» (S_1), «захвачен левый МПД» (S_2), «захвачен правый МПД» (S_3), «захвачены оба МПД, обработка данных очередной итерации» (S_4). МПД может быть либо свободным, либо захваченным. Событиями будем считать захват и освобождение МПД. Этот вариант работы представляется сетью Петри, в которой каждому ПЭ соответствуют четыре позиции, реализующие описанные условия, а каждому МПД - позиция (фишка в которой означает, что МПД свободен) (рис. 49).

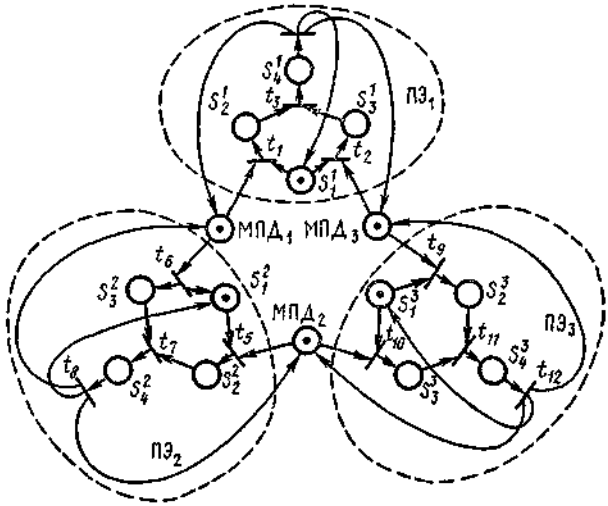


Рис. 49

Можно убедиться, что дерево достижимости этой сети Петри содержит две терминальные маркировки: $\mu_1 = \{S_2^1,\ S_2^2,\ S_2^3\}$ и $\mu_2 = \{S_3^1,\ S_3^2,\ S_3^3\}$, представляющие ситуации, когда каждый ПЭ захватывает левый и правый МПД соответственно. Это означает, что сеть Петри не активна, т. е. в системе возможны тупиковые ситуации.

При другом варианте работы системы ПЭ осуществляют только одновременный захват смежных МПД (если он возможен). В этом случае каждому ПЭ соответствуют только условия S_1 и S_4 (рис. 50).

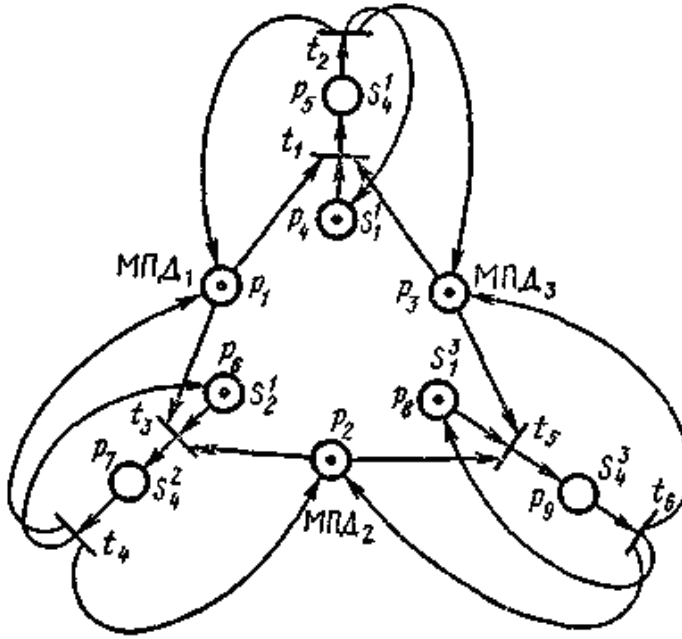


Рис. 50

Дерево достижимости (рис. 51) не содержит терминальных маркировок, сеть Петри активна.

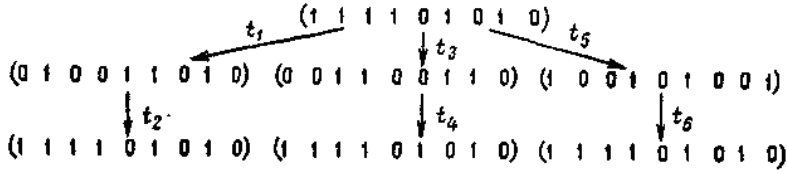


Рис. 51

Кроме того, из рассмотрения дерева достижимости очевидно, что сеть Петри безопасная (позиции интерпретируются как простые условия). В системе осуществляется распределение ресурсов, которые не появляются и не исчезают, т. е. выполняется закон сохранения.

Определим, является ли сеть Петри сохраняющей. Для этого решим уравнение $DW=0$, которое принимает вид

$$\begin{bmatrix} -1 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \end{bmatrix} = 0.$$

Решением его является $W = (1, 1, 1, 1, 3, 1, 3, 1, 3)$. Действительно, позиции p_5, p_7, p_9 представляют условия, связанные с тремя устройствами, остальные позиции — условия, которые связаны с одним устройством. Таким образом, сеть Петри обладает основными необходимыми свойствами, что обеспечивает работоспособность второго варианта.

Попытки моделирования реальных систем привели к различным доопределениям и модификациям сетей Петри. В основном эти модификации связаны с изменением правила запуска переходов.

Мощность моделирования обычных сетей Петри ограничена невозможностью проверки позиции на нуль (т. е. того, что маркировка позиции нулевая). Одним из способов преодоления этого недостатка является введение *сдерживающих дуг*. По новым правилам запуска переход разрешен, если фишки есть в его обычных входных позициях (из которых ведут обычные дуги) и отсутствуют в сдерживающих входных позициях (из которых ведут сдерживающие дуги). Сдерживающая дуга изображается как обычная, только на конце имеет вместо стрелки маленький кружок (это обозначение заимствовано из теории переключательных схем, где кружок означает «не») (рис. 52).

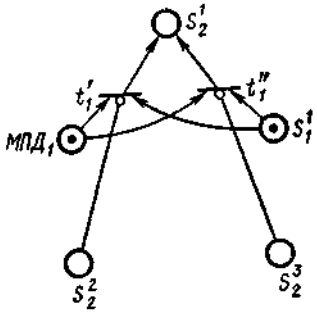


Рис. 52

Если в обычных сетях Петри переход запускается по *логике И,* то в *сетях Петри со сдерживающими дугами* логика расширена до включения отрицаний. Поскольку событие может представляться несколькими переходами, можно смоделировать событие, предусловие которого записывается как объединение нескольких конъюнкций условий и отрицаний условий, соответствующих позициям сети Петри со сдерживающими дугами. Таким образом, сети Петри позволяют моделировать предусловия в виде ДНФ, т. е. условия самого общего вида.

Рассмотренная задача организации работы специализированной вычислительной системы может быть решена и первым вариантом, в котором разрешен последовательный захват МПД (см. рис. 49), но с предотвращением тупиковых ситуаций. Очевидно, что возможны две тупиковые ситуации, описываемые маркировками с фишками в позициях S_2^1, S_2^2, S_2^3 и S_3^1, S_3^2, S_3^3 (см. рис. 49). Для предотвращения первой тупиковой ситуации необходимо в маркировках $\{S_2^1, S_2^2\}$, $\{S_2^1, S_2^3\}$, $\{S_2^2, S_2^3\}$ предотвратить проявление фишки в позициях S_2^3, S_2^2, S_2^1 соответственно. Следовательно, переход t_1 должен иметь в качестве предусловия конъюнкцию $МПД_1 \& S_1^1 \& (S_1^1 \& S_2^2)$, т. е. его нужно заменить на переходы t'_1 и t'' с предусловиями $МПД_1 \& \bar{S}_1' \& \bar{S}_2^1$ и $МПД_1 \& S_1^1 \& \bar{S}_2^2$ (рис. 52). Аналогично следует поступить с переходами t_5, t_9 (см. рис. 49). Подобные действия предпринимаются и для предотвращения второй тупиковой ситуации. Другие предложения по изменению правил запуска либо эквивалентны введению сдерживающих дуг, либо носят даже более частный характер. Например, в сетях Петри с областями ограничения имеются множества позиций (называемые областями ограничения), в которых фишки одновременно находиться не могут. Правила запуска модифицированы так, чтобы не нарушать это условие. Если в сеть Петри, изображенную на рис. 49, включить две области ограничения $\{S_2^1, S_2^2, S_2^3\}$ и $\{S_3^1, S_3^2, S_3^3\}$, то можно предотвратить возникновение тупиковых ситуаций.

Литература

1. Глушков В.М. Синтез цифровых автоматов. Физматгиз. М., 1962.
2. Мелихов А.Н. Ориентированные графы и конечные автоматы. М. Наука. 1971
3. Поспелов Д.А. Логические методы анализа и синтеза схем. М. Энергия. 1974.
4. Основы кибернетики. Под ред. К.А Пупкова. М. Высшая школа. 1976.
5. Кудряшов В.Б., Подколзин А.С., Ушчумлич Ш. Введение в теорию абстрактных автоматов. М. Издательство Московского университета. 1985.
6. Лазарев В.Г., Пийль Е.И. Синтез управляющих автоматов. М. Энергоатомиздат. 1989.
7. Кузнецов О.П, Адельсон-Вельский Г.М. Дискретная математика для инженера. М. Энергия. 1980.
8.Горбатов В.А. Основы дискретной математики. М. Высшая школа. 1986.
9. Ачасова С.М. Алгоритми синтеза автоматов на программируемых матрицах. М. Радио и связь. 1987.
10. Кук Д., Бейз Г. Компьютерная математика. М. Наука. 1990.
11. Алгебраическая теория автоматов, языков и полугрупп. Под редакцией Арбиба М.А. М. Статистика. 1975.
12. А. Е.Кононюк. Дискретно-непрерывная математика. Кн. Автоматы, ч.1. Детерминированные автоматы (начало). Киев, Освита Украины. 2017, 658 с.