

Федеральное агентство по образованию
Российской Федерации
Омский государственный университет им. Ф.М. Достоевского
Факультет компьютерных наук
Кафедра информационной безопасности

Н.Ф. Богаченко, Р.Т. Файзуллин

СИНТЕЗ ДИСКРЕТНЫХ АВТОМАТОВ

Омск 2006

УДК 519.713
Б 733

Богаченко Н.Ф., Файзуллин Р.Т. Синтез дискретных автоматов: Учебное пособие. Омск: Издательство На-
следие. Диалог-Сибирь, 2006. 150 с.: ил.

В пособии изложены основные принципы и алгоритмы абстрактного и структурного синтеза дискретных автоматов. Теоретический материал сопровождается примерами решения типовых задач с подробными пояснениями.

Для студентов, обучающихся по специальности 075200 – «Компьютерная безопасность» и по специальности 220100 – «Вычислительные машины, комплексы, системы и сети».

Рекомендовано к изданию учебно-методической комиссией факультета компьютерных наук ОмГУ

© Омский госуниверситет, 2006

Глава 1

Абстрактный синтез

1.1. Дискретный автомат

Как правило, термин «*автомат*» используется в двух аспектах: устройство, выполняющее некоторые функции без непрерывного участия человека, или *математическая модель реального технического автомата*. Нас будет интересовать именно второе определение. Рассмотрим, как осуществляется построение такой модели.

Поведение любого технического устройства можно описать в терминах некоторых *физических переменных*:

- *входные* переменные – их значения задаются извне, они не определяются самим устройством, но влияют на его поведение;

- *выходные* переменные – для получения их значений построено само устройство, эти значения определяются как некоторые функции, зависящие от входных переменных;

- *внутренние* переменные – не являются ни входными, ни выходными, но необходимы для описания поведения устройства.

Будем рассматривать такие технические устройства, в которых значения переменных *проквантованы*. Это означает,

что из области значений каждой переменной выделены непересекающиеся интервалы, а значения самой переменной учитываются с точностью до интервала. Подобные устройства называются **дискретными устройствами**.

Проквантованность переменных позволяет изучать дискретные устройства с единой точки зрения. Их непосредственное рассмотрение заменяется анализом абстрактной модели, называемой **дискретным автоматом** (см. рис. 1.1).



Каждая физическая переменная, в общем случае с бесконечным числом значений, заменяется на **дискретную переменную**

Рис. 1.1. Переход к дискретному автомату

Число значений **дискретной переменной** конечно и равно числу выделенных интервалов в области значений физической переменной.

Пусть число интервалов физической переменной равно 2. Тогда ей соответствует дискретная переменная, принимающая значения «0» или «1» – это **двоичная** (**логическая** или **булева**) переменная.

Двоичные переменные дискретного автомата, соответствующие входным и выходным физическим переменным дискретного устройства, называются **входными и выходными полюсами** дискретного автомата. **Состояние полюса** – это значение соответствующей двоичной переменной.

Вход автомата – это совокупность входных полюсов. **Выход** автомата – это совокупность выходных полюсов. **Состояние входа (выхода)** – это вектор состояний входных (выходных) полюсов.

Комбинационный автомат – это дискретный автомат, удовлетворяющий следующему условию: каждому состоянию входа должно соответствовать *вполне определенное* состояние

выхода. Отсюда следует, что двоичная переменная y_i , описывающая состояние i -го выходного полюса, может быть представлена как логическая функция φ_i двоичных переменных x_1, x_2, \dots, x_n , которые описывают состояния входных полюсов. Следовательно, комбинационный автомат реализует некоторую систему логических функций:

$$\begin{cases} y_1 = \varphi_1(x_1, \dots, x_n), \\ \dots \\ y_m = \varphi_m(x_1, \dots, x_n); \end{cases} \quad (1.1)$$

или в векторном виде:

$$Y = \Phi(X). \quad (1.2)$$

Данная система логических функций называется **функциональной моделью** комбинационного автомата (см. рис. 1.2).

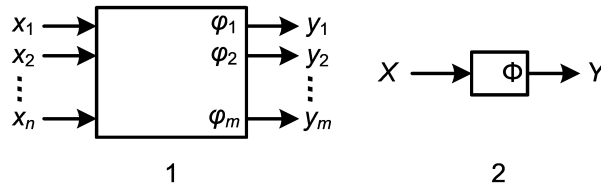


Рис. 1.2. Комбинационный автомат: 1 – соответствует модели (1.1), 2 – соответствует модели (1.2)

Заметим, что нами допускается некоторая идеализация дискретных устройств. На практике реальные устройства обладают *инерционностью*: изменение состояния входа автомата не приводит к мгновенному образованию нового состояния выхода – для этого требуется некоторое время. В ряде случаев этим явлением можно пренебречь.

Пусть n и m – это число входных и выходных полюсов, тогда, очевидно, существует 2^n различных состояний входа и 2^m различных состояний выхода. Тем не менее, для заданного

комбинационного автомата некоторые состояния выхода могут оказаться *нереализуемыми* ни при каком из состояний входа.

Пример 1.1. В качестве примера простейшего комбинационного автомата можно рассмотреть *элементарный преобразователь*. Это дискретное устройство, которому соответствует комбинационный автомат, имеющий один входной полюс, один выходной полюс, и состояния этих полюсов всегда совпадают. На практике это различные датчики. Функциональная модель элементарного преобразователя имеет вид: $y = \varphi(x) \equiv x$.

Рассмотрим работу дискретного автомата в течение некоторого отрезка времени. Изменяя во времени состояния входных полюсов, мы получаем последовательность состояний входа. В этом случае на выходе автомата образуется последовательность состояний выхода:

$$X^1, \dots, X^{t-k}, \dots, X^t \longrightarrow Y^1, \dots, Y^{t-k}, \dots, Y^t.$$

Автоматом без памяти называется автомат, поведение которого не зависит от прошлого, а полностью определяется текущим состоянием входа. Функциональная модель автомата без памяти имеет вид:

$$Y^t = \Phi(X^t). \quad (1.3)$$

Пример 1.2. Если значению (1011) вектора X соответствует значение (101) вектора Y , то это соответствие будет реализовываться автоматом всегда, вне зависимости от предшествующих значений вектора X .

Исходя из определений, получаем, что комбинационный автомат и автомат без памяти – эквивалентные понятия.

Автомат с памятью (или *последовательный автомат*) реализует функциональную зависимость не между отдельными состояниями входа и выхода, а между их последовательностями:

$$Y^t = \Phi(X^t, X^{t-1}, \dots, X^{t-k}). \quad (1.4)$$

Для этого последовательный автомат наделяется некоторым множеством **внутренних состояний**. В каждом из этих состояний последовательный автомат ведет себя подобно некоторому комбинационному автомату, то есть реализует однозначную функциональную зависимость между состояниями входа и выхода. Но *реакция* последовательного автомата на очередное состояние входа может выразиться также в смене внутреннего состояния.

До сих пор мы ограничивались рассмотрением логических переменных, то есть состояния входа и выхода автомата описывались двоичными векторами X и Y . В общем случае природа состояний входа и выхода, а также внутренних состояний автомата может быть произвольной. Будем считать, что эти состояния описываются некоторыми символами, которые выбираются из соответствующих множеств:

- 1) $Z = \{z_1, z_2, \dots, z_F\}$ – **абстрактный входной алфавит**;
- 2) $W = \{w_1, w_2, \dots, w_G\}$ – **абстрактный выходной алфавит**;
- 3) $A = \{a_1, a_2, \dots, a_M\}$ – **множество внутренних состояний**;

1.2. Определение абстрактного автомата

Абстрактный автомат – это математическая модель дискретного устройства, которая задается множеством из 6 элементов:

$$S = \{A, Z, W, \delta, \lambda, a_1\}, \quad (1.5)$$

где $A = \{a_1, \dots, a_m, \dots, a_M\}$ – множество **внутренних состояний** (алфавит состояний); $Z = \{z_1, \dots, z_f, \dots, z_F\}$ – множество **входных сигналов** или входных состояний (входной алфавит); $W = \{w_1, \dots, w_g, \dots, w_G\}$ – множество **выходных сигналов** или выходных состояний (выходной алфавит); $\delta : D_\delta \subseteq A \times Z \rightarrow A$ – **функция переходов**: $a_s = \delta(a_m, z_f)$;

$\lambda : D_\lambda \subseteq A \times Z \rightarrow W$ – **функция выходов**: $w_g = \lambda(a_m, z_f)$; $a_1 \in A$ – начальное состояние автомата¹.

Автомат называется **конечным**, если конечны множества A , Z и F . В дальнейшем будем рассматривать только конечные автоматы.

Заметим, что абстрактный автомат – это строгое математическое определение автомата с памятью или последовательного автомата.

Вообще говоря, абстрактный автомат задается функцией Φ такой, что $\Phi(z(t), z(t-1), \dots, z(t-k)) = w(t)$. Чтобы устранить время, как явную переменную, вводится понятие «состояние автомата». Теперь выходной сигнал w есть функция λ , зависящая от входного сигнала z и состояния a .

1.3. Автоматы Мура и Мили

Как отмечалось ранее, автомат имеет один вход и один выход или входной и выходной **каналы**. В каждый момент дискретного времени $t = 1, 2, \dots$ автомат находится в определенном состоянии $a(t)$ множества A (см. рис. 1.3).

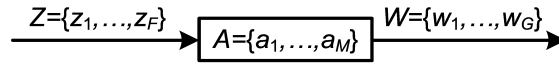


Рис. 1.3. Абстрактный автомат

В начальный момент, когда $t = 1$, автомат всегда находится в состоянии $a(1) = a_1$. В момент времени t автомат, находясь в состоянии $a(t)$, способен:

- 1) воспринимать на входном канале сигнал $z(t) \in Z$;
- 2) выдавать на выходном канале сигнал $w(t) \in W : w(t) = \lambda(a(t), z(t))$;
- 3) перейти в состояние $a(t+1) \in A : a(t+1) = \delta(a(t), z(t))$.

¹Далее, если не оговорено противное, под словом «автомат» будем понимать абстрактный автомат, а понятие «внутреннее состояние» сократим до термина «состояние».

Структура автомата представлена на рисунке 1.4. Переход от момента времени t к моменту $(t + 1)$ называется **тактом**. Память автомата (или запоминающее устройство) задерживает на один такт вычисляемое в текущий момент t значение функции δ . В момент времени $(t + 1)$ оно появляется на выходе запоминающего устройства.

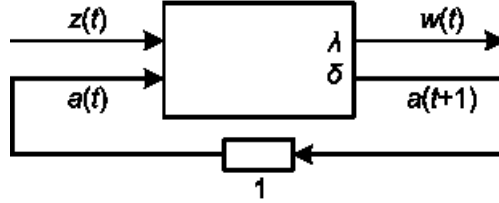


Рис. 1.4. Структура автомата с памятью; 1 – память автомата

Смысл понятия абстрактного автомата состоит в том, что он реализует отображение множества *слов* входного алфавита Z во множество слов выходного алфавита W . Другими словами, автомат индуцирует отображение Φ , которое ставит в соответствие **входному слову** $(z(1), z(2), z(3), \dots)$ **выходное слово** $(w(1), w(2), w(3), \dots)$ (см. рис. 1.5).

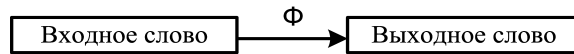


Рис. 1.5. Отображение, индуцированное автоматом

Исходя из вышесказанного, автомат можно описать одной из следующих функциональных моделей:

1. **Модель Мили (автомат Мили):**

$$\begin{cases} a(t+1) = \delta(a(t), z(t)); \\ w(t) = \lambda(a(t), z(t)). \end{cases} \quad (1.6)$$

2. **Модель Мура (автомат Мура):**

$$\begin{cases} a(t+1) = \delta(a(t), z(t)); \\ w(t) = \lambda(a(t)). \end{cases} \quad (1.7)$$

В обоих случаях $t = 1, 2, 3, \dots$. Пара $(a(t), z(t))$ называется полным состоянием автомата.

В зависимости от способа задания функции переходов δ и функции выходов λ выделяют два *класса* языков описания автоматов:

1. Начальные языки.
2. Автоматные (стандартные) языки.

1.4. Начальные языки

В начальных языках автомат описывается на поведенческом уровне. В этом случае функции δ и λ явно не заданы.

1.4.1. Язык регулярных выражений алгебры событий

Пусть задано конечное множество входных *букв* $Z = \{z_1, \dots, z_F\}$. Каждая буква этого алфавита соответствует некоторому *элементарному событию*. Над событиями разрешены следующие *операции*:

- 1) $z_i \vee z_j$ – сложение (дизъюнкция);
- 2) $z_i \cdot z_j$ – умножение (конъюнкция);
- 3) $\{z_i\}$ (или z_i^*) – итерация; (первые две операции – бинарные, последняя – унарная).

Регулярное выражение в алфавите Z – это выражение, построенное из букв алфавита Z , символов операций сложения, умножения, итерации, с использованием, если это необходимо, круглых скобок. Всякое регулярное выражение R определяет **регулярное событие** S , полученное из элементарных событий применением конечного числа раз операций 1) – 3).

Пример 1.3. Пусть $Z = \{x, y, z\}$, $R = x\{x \vee y \vee z\}(y \vee z)$. Регулярному выражению R соответствует регулярное событие S , состоящее из всех слов, которые начинаются буквой x и заканчиваются буквой y или z .

Регулярные события представляют особый интерес, так как для них справедливо нижеследующее.

Утверждение 1.1. *Регулярные события и только они представимы в конечных автоматах [6].*

Пример 1.4. Пусть требуется описать автомат, который выдает сигнал w_1 каждый раз, когда на входе происходит смена сигналов с z_1 на z_2 .

Из условия следует, что сигнал w_1 должен выдаваться на любую входную последовательность, которая заканчивается на « $z_1 z_2$ ». Событие «любая входная последовательность» записывается как $\{z_1 \vee z_2\}$. Таким образом, событие S_1 , в ответ на которое должен выдаваться сигнал w_1 , описывается следующим регулярным выражением: $S_1|w_1 = \{z_1 \vee z_2\}z_1 z_2$.

1.4.2. Язык логических схем алгоритмов

Язык логических схем алгоритмов был предложен А.А. Ляпуновым в 1953 г. При данном подходе функциональная схема автомата описывается конечной строкой, которая состоит из:

- 1) символов операторов;
- 2) логических условий;
- 3) верхних и нижних стрелок, которым приписаны натуральные числа $\binom{i}{\uparrow \downarrow}$.

Пусть $Y = \{Y_1, Y_2, \dots, Y_T\}$ – множество операторов, $X = \{x_1, x_2, \dots, x_L\}$ – множество логических условий (подробнее см. § 4.2).

Логическая схема алгоритма (ЛСА)² удовлетворяет следующим условиям:

- 1) содержит один начальный Y_n и один конечный Y_k операторы;
- 2) перед оператором Y_n и после оператора Y_k стрелки отсутствуют;

²В главе 2 под логической схемой мы будем понимать другую структуру, а именно, структурную схему автомата (см. § 2.2).

- 3) после каждого логического условия всегда стоит верхняя стрелка \uparrow ;
- 4) не существует двух одинаковых (с одинаковыми цифрами) нижних стрелок \downarrow ;
- 5) для каждой нижней стрелки должна быть по крайней мере одна соответствующая ей (с одинаковой цифрой) верхняя стрелка;
- 6) для каждой верхней стрелки должна быть ровно одна, соответствующая ей (с одинаковой цифрой), нижняя стрелка.

Пример 1.5. Опишем поведение автомата с помощью ЛСА:

$$Y_n x_1 \overset{1}{\uparrow} Y_1 \overset{1}{\downarrow} x_2 \overset{2}{\uparrow} Y_2 Y_3 x_3 \overset{2}{\uparrow} Y_4 \overset{2}{\downarrow} Y_5 Y_k \quad (1.8)$$

Данная ЛСА включает в себя операторы начала и конца, пять операторов Y_1, \dots, Y_5 , три логических условия x_1, x_2, x_3 . Начальному оператору соответствует начальное состояние автомата.

Если в начальном состоянии логическое условие $x_1 = 1$, то выполняем оператор Y_1 – первый справа после логического условия x_1 , а затем проверяем логическое условие x_2 . Если $x_1 = 0$, то выходим по верхней стрелке $\overset{1}{\uparrow}$ и входим по нижней стрелке с той же цифрой. Здесь проверяем логическое условие x_2 (минуя выполнение оператора Y_1).

Если $x_2 = 0$, то переходим к выполнению оператора Y_5 . Если $x_2 = 1$, то выполняем операторы Y_2, Y_3 и если $x_3 = 1$, то выполняем оператор Y_4 , а затем Y_5 ; если $x_3 = 0$, то сразу переходим к оператору Y_5 .

Далее выполняем оператор Y_k – работа автомата завершена.

1.4.3. Язык граф-схем алгоритмов

Граф-схема алгоритма (ГСА) – это ориентированный связный граф, содержащий вершины четырех типов (рис. 1.6):

- 1) начальную (входов нет, один выход);
- 2) конечную (один вход, выходов нет);

- 3) операционную или операторную (один вход, один выход);
- 4) условную (один вход, два выхода, помеченные «0» и «1»).

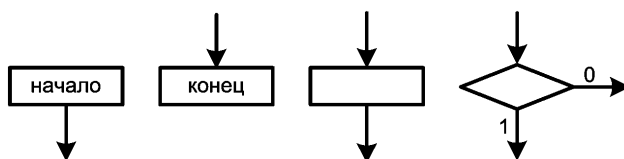


Рис. 1.6. Вершины ГСА

ГСА удовлетворяет следующим условиям:

- 1) содержит конечное множество вершин, каждая из которых принадлежит перечисленным выше типам;
- 2) имеет одну начальную и одну конечную вершины;
- 3) входы и выходы вершин соединены друг с другом с помощью дуг, направленных от выхода к входу (это общая направленность, но локально могут быть и петли);
- 4) каждый выход соединен только с одним входом;
- 5) любой вход соединен по крайней мере с одним выходом;
- 6) для любой вершины графа существует по крайней мере один путь из этой вершины к конечной вершине;
- 7) один из выходов условной вершины может соединяться с ее входом, что недопустимо для операторной вершины;
- 8) в каждой условной вершине записывается один из элементов множества $X = \{x_1, \dots, x_L\}$, так называемого множества логических условий; разрешается в различных условных вершинах запись одинаковых элементов множества X ;
- 9) в каждой операторной вершине записывается оператор, то есть элемент множества $Y = \{Y_1, Y_2, \dots, Y_T\}$; разрешается в различных операторных вершинах запись одинаковых операторов.

Содержательная ГСА – ГСА, в которой логические условия и операторы описываются в содержательных терминах.

Пример 1.6. В качестве логического условия x_i может выступать, к примеру, выражение $(A = 1 \vee B = 0)$. Для некоторого оператора Y_i в содержательной ГСА допустима запись $(A := A + 1)$.

Основные правила выполнения ГСА представлены в соответствующих ГОСТах. Существуют алгоритмы перехода от ГСА к ЛСА и от ЛСА к ГСА [1].

Пример 1.7. На рисунке 1.7 преведена ГСА, соответствующая логической схеме (1.8).

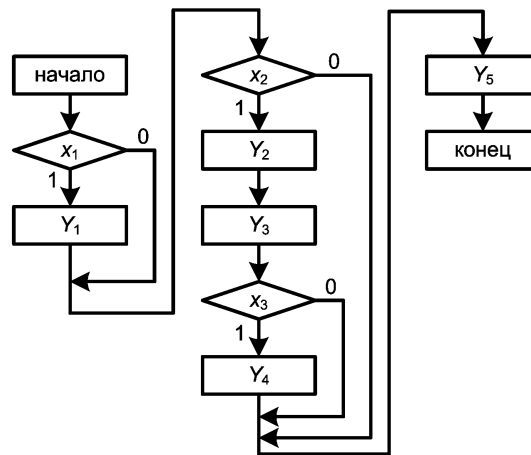


Рис. 1.7. Граф-схема алгоритма

Замечание 1.1. Язык ГСА используется не только для описания схем функционирования автомата на поведенческом уровне, но и для записи микропрограмм работы дискретных устройств (см. § 4.2).

1.5. Автоматные (стандартные) языки

Для строгого задания автомата S необходимо описать все элементы множества $S = \{A, Z, W, \delta, \lambda, a_1\}$.

1.5.1. Табличный метод

Для моделей Мили и Мура правила построения таблиц изложим отдельно.

Таблицы переходов и выходов автомата Мили. Обе таблицы строятся по следующей схеме: строки – входные сигналы, столбцы – состояния автомата. Крайнему левому столбцу соответствует начальное состояние автомата. Таблица переходов описывает внутренние состояния автомата, таблица выходов – выходные сигналы (см. табл. 1.1, 1.2).

Таблица 1.1. Общий вид таблицы переходов автомата Мили

	a_1	\dots	a_M
z_1	$\delta(a_1, z_1)$	\dots	$\delta(a_M, z_1)$
\dots	\dots	\dots	\dots
z_F	$\delta(a_1, z_F)$	\dots	$\delta(a_M, z_F)$

Таблица 1.2. Общий вид таблицы выходов автомата Мили

	a_1	\dots	a_M
z_1	$\lambda(a_1, z_1)$	\dots	$\lambda(a_M, z_1)$
\dots	\dots	\dots	\dots
z_F	$\lambda(a_1, z_F)$	\dots	$\lambda(a_M, z_F)$

Часто эти две таблицы объединяются в одну. При этом новое состояние (*состояние перехода*) и выходной сигнал, стоящие в одной клетке, разделяются косой чертой.

Пример 1.8. Автомат Мили S_1 задан таблицами переходов и выходов (см. табл. 1.3, 1.4).

Отмеченная таблица переходов автомата Мура. Таблица строится аналогично таблице переходов автомата Ми-

Таблица 1.3. Таблица переходов автомата Мили S_1

	a_1	a_2	a_3
z_1	a_3	a_1	a_1
z_2	a_1	a_3	a_2

Таблица 1.4. Таблица выходов автомата Мили S_1

	a_1	a_2	a_3
z_1	w_1	w_1	w_2
z_2	w_1	w_2	w_1

ли. В первой строке таблицы каждому состоянию приписывается выходной сигнал автомата (см. табл. 1.5).

Таблица 1.5. Общий вид таблицы переходов автомата Мура

	$\lambda(a_1)$	\dots	$\lambda(a_M)$
	a_1	\dots	a_M
z_1	$\delta(a_1, z_1)$	\dots	$\delta(a_M, z_1)$
\dots	\dots	\dots	\dots
z_F	$\delta(a_1, z_F)$	\dots	$\delta(a_M, z_F)$

Пример 1.9. Автомат Мура S_2 задан отмеченной таблицей переходов (см. табл. 1.6).

Таблица 1.6. Отмеченная таблица переходов автомата Мура S_2

	w_1	w_1	w_3	w_2	w_3
	a_1	a_2	a_3	a_4	a_5
z_1	a_2	a_5	a_5	a_3	a_3
z_2	a_4	a_2	a_2	a_1	a_1

1.5.2. Графовый метод

Граф автомата – это ориентированный связный граф, вершины которого соответствуют состояниям, а дуги – переходам между ними. Граф автомата строится по следующим правилам:

1. Две вершины графа автомата a_m (исходное состояние) и a_s (состояние перехода) соединяются дугой, направленной от a_m к a_s , если в автомате имеется переход от a_m к a_s , то есть, если $\exists z_f$ такое, что $a_s = \delta(a_m, z_f)$.

2. Дуге (a_m, a_s) графа автомата приписывается входной сигнал z_f и, для автомата Мили, выходной сигнал $w_g = \lambda(a_m, z_f)$. При описании автомата Мура в виде графа выходной сигнал $w_g = \lambda(a_m)$ записывается внутри вершины a_m или рядом с ней.

3. Если переход автомата из состояния a_m в состояние a_s происходит под действием нескольких входных сигналов, то дуге (a_m, a_s) приписываются все эти входные сигналы и, для автомата Мили, соответственные выходные сигналы.

Пример 1.10. На рисунке 1.8 представлены графы автоматов Мили и Мура.

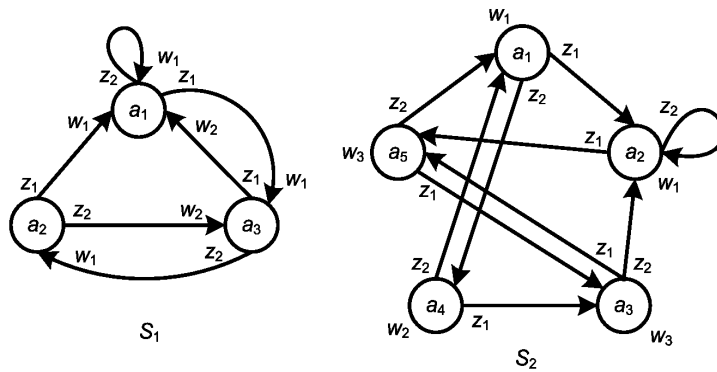


Рис. 1.8. Графы автоматов Мили (S_1) и Мура (S_2)

1.5.3. Матричный метод

Матрица соединений автомата – квадратная матрица $C = \|c_{ij}\|$, строки которой соответствуют исходным состояниям, а столбцы – состояниям перехода.

Для автомата Мили элемент $c_{ij} = z_f|w_g$, стоящий на пересечении i -й строки и j -го столбца, соответствует входному сигналу z_f , вызывающему переход из состояния a_i в состояние a_j , и выходному сигналу w_g , выдаваемому на этом переходе.

Если переход из a_i в a_j происходит под действием нескольких сигналов, то элемент c_{ij} представляет собой множество пар, соединенных знаком дизъюнкции.

Пример 1.11. Матрица соединений автомата Мили S_1 представлена в таблице 1.7.

Таблица 1.7. Матрица соединений автомата Мили S_1

	a_1	a_2	a_3
a_1	$z_2 w_1$	—	$z_1 w_1$
a_2	$z_1 w_1$	—	$z_2 w_2$
a_3	$z_1 w_2$	$z_2 w_1$	—

Для модели Мура элемент c_{ij} равен множеству всех входных сигналов на переходе (a_i, a_j) , а выход описывается вектором выходов W (см. табл. 1.8).

Таблица 1.8. Общий вид вектора выходов W

a_1	$\lambda(a_1)$
\vdots	\vdots
a_M	$\lambda(a_M)$

Пример 1.12. Матрица соединений и вектор выходов автомата Мура S_2 представлены в таблицах 1.9, 1.10.

Замечание 1.2. Далее будем рассматривать только **детерминированные** автоматы, у которых выполнено условие однозначности переходов: автомат, находящийся в некотором состоянии, под действием любого входного сигнала не может перейти более чем в одно состояние. Отсюда следует, что 1) в графе автомата из одной вершины не могут выходить две и более дуги, отмеченные одним и тем же входным сигналом;

Таблица 1.9. Матрица соединений автомата Мура S_2

	a_1	a_2	a_3	a_4	a_5
a_1	—	z_1	—	z_2	—
a_2	—	z_2	—	—	z_1
a_3	—	z_2	—	—	z_1
a_4	z_2	—	z_1	—	—
a_5	z_2	—	z_1	—	—

Таблица 1.10. Вектор выходов автомата Мура S_2

a_1	w_1
a_2	w_1
a_3	w_3
a_4	w_2
a_5	w_3

2) в матрице соединений в каждой строке любой входной сигнал не должен встречаться более одного раза.

Замечание 1.3. Если автомат обладает лишь одним внутренним состоянием, следовательно он должен в нем все время оставаться – такой автомат можно рассматривать как комбинационный.

Замечание 1.4. Если автомат обладает лишь одним входным сигналом, то он называется *автономным*.

Упражнение 1.1. Как выглядят таблицы переходов и выходов, графы и матрицы соединений комбинационного и автономного автоматов? Приведите примеры.

Упражнение 1.2. Приведите примеры (графы и матрицы соединений) недетерминированных автоматов.

1.6. Полностью и частично определенные автоматы

Автомат называется *полностью определенным* или *полным*, если $D_\delta = D_\lambda = A \times Z$. Иными словами, для любого начального состояния и любой входной последовательности (в пределах алфавита Z) однозначно определена выходная последовательность.

Автомат называется *частично определенным* или *ча-*

стичным, если функции δ и λ определены не для всех пар $(a_m, z_f) \in A \times Z$. В табличном задании таких автоматов на месте неопределенных состояний и выходных сигналов ставится прочерк. На графе автомата неопределенному состоянию можно поставить в соответствие дополнительную вершину.

Неопределенное состояние автомата называется **границным**. Дальнейшее поведение автомата, попавшего в граничное состояние, не определяется. Выходная последовательность, содержащая неопределенные сигналы, называется **частичной**.

Заметим, что модель частичного автомата является более абстрактной, чем модель полного автомата: она задает целое множество дискретных устройств.

Пример 1.13. Пусть частичный автомат Мили S_3 задан таблицами переходов и выходов (см. табл. 1.11, 1.12). На рисунке 1.9 представлен граф этого частичного автомата.

Таблица 1.11. Таблица переходов частичного автомата Мили S_3

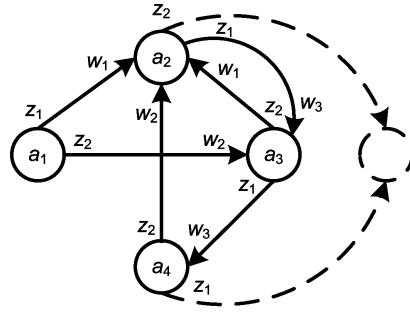
	a_1	a_2	a_3	a_4
z_1	a_2	a_3	a_4	—
z_2	a_3	—	a_2	a_2

Таблица 1.12. Таблица выходов частичного автомата Мили S_3

	a_1	a_2	a_3	a_4
z_1	w_1	w_3	w_3	—
z_2	w_2	—	w_1	w_2

1.7. Синхронные и асинхронные автоматы

Синхронный автомат — автомат, поведение которого рассматривается в дискретном времени, представляющем собой последовательность тактов t_1, t_2, t_3, \dots . При этом предполагается *постоянный интервал дискретности*, то есть изменение состояний автомата фиксируется через равные промежутки времени.

Рис. 1.9. Граф частичного автомата Мили S_3

Заметим, что до сих пор мы рассматривали синхронные автоматы.

Асинхронный автомат – автомат, для которого интервал дискретности является *переменным*. Внутреннее состояние асинхронного автомата может меняться лишь при изменении входного сигнала. Таким образом теряет смысл рассмотрение входных последовательностей, которые содержат одинаковые соседние символы.

Дадим строгое определение асинхронного автомата, используя понятие устойчивого состояния. Состояние a_s автомата S называется **устойчивым**, если $\forall z_f \in Z$ такого, что $\delta(a_m, z_f) = a_s$ имеет место $\delta(a_s, z_f) = a_s$. Автомат S называется **асинхронным**, если каждое его состояние $a_s \in A$ устойчиво. Автомат S называется **синхронным**, если он не является асинхронным.

Очевидно, что если в *графе* асинхронного автомата в состоянии a_s есть переход из другого состояния под действием сигнала z_f , то в вершине a_s должна быть петля, отмеченная символом z_f . Если в *таблице переходов* асинхронного автомата состояние a_s стоит на пересечении строки z_f и столбца a_m ($m \neq s$), то это состояние a_s обязательно должно встретиться в этой же строке в столбце a_s .

Построенные на практике автоматы как правило являются

асинхронными. На уровне абстрактной теории удобнее оперировать синхронными автоматами.

Пример 1.14. Рассмотрим автомат Мура S_4 (рис. 1.10, табл. 1.13). Все его состояния устойчивы, следовательно, этот автомат является асинхронным.

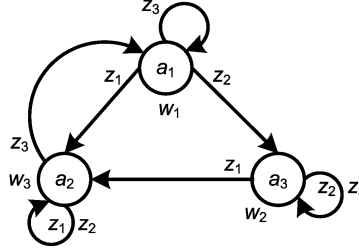


Рис. 1.10. Граф асинхронного автомата Мура S_4

Таблица 1.13. Таблица переходов асинхронного автомата Мура S_4

	w_1	w_3	w_2
	a_1	a_2	a_3
z_1	a_2	a_2	a_2
z_2	a_3	a_2	a_3
z_3	a_1	a_1	a_3

1.8. Совмещенная модель автомата (С-автомат)

Абстрактный С-автомат – это математическая модель дискретного устройства, определяемая множеством из 8 элементов:

$$S = \{A, Z, W, U, \delta, \lambda_1, \lambda_2, a_1\}, \quad (1.9)$$

где $W = \{w_1, \dots, w_g, \dots, w_G\}$ – множество выходных сигналов 1-го типа; $U = \{u_1, \dots, u_h, \dots, u_H\}$ – множество выходных сигналов 2-го типа; $\lambda_1 : D_{\lambda_1} \subseteq A \times Z \rightarrow W$ – функция выходов 1-го типа; $\lambda_2 : D_{\lambda_2} \subseteq A \rightarrow U$ – функция выходов 2-го типа. Так же как и ранее, A – множество состояний, Z – множество входных сигналов, a_1 – начальное состояние, δ – функция переходов.

Абстрактный С-автомат (не путать со структурным, здесь «С» означает совмещенный) имеет один вход, на который поступают сигналы из входного алфавита Z , и два выхода (см. рис. 1.11).

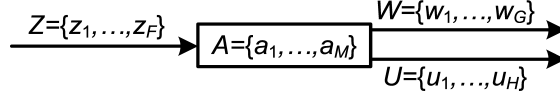


Рис. 1.11. С-автомат

Отличие С-автомата от моделей Мили и Мура состоит в том, что он одновременно реализует две функции выходов: λ_1 и λ_2 , каждая из которых характерна для этих моделей в отдельности.

Выходной сигнал $u_h = \lambda_2(a_m)$ выдается все время, пока автомат находится в некотором состоянии a_m . Выходной сигнал $w_g = \lambda_1(a_m, z_f)$ выдается во время действия входного сигнала z_f при нахождении автомата в состоянии a_m . (Это и есть отличие в функционировании автоматов Мура и Мили.)

Функциональная модель С-автомата имеет следующий вид:

$$\begin{cases} a(t+1) = \delta(a(t), z(t)); \\ w(t) = \lambda_1(a(t), z(t)); \\ u(t) = \lambda_2(a(t)), \end{cases} \quad (1.10)$$

где $t = 1, 2, \dots$ – такты времени.

При задании С-автомата в виде графа совмещаются графы автоматов Мили и Мура – вблизи «кружка»-состояния записывается выход u_h .

Пример 1.15. Граф С-автомата S_5 представлен на рисунке 1.12.

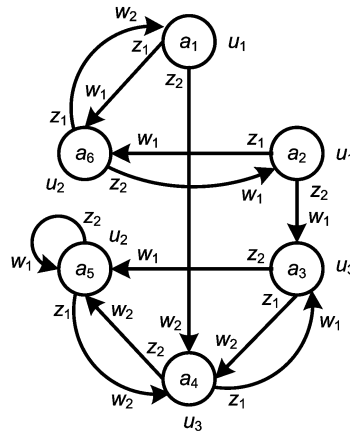


Рис. 1.12. Граф С-автомата S_5

1.9. Связь между автоматами Мили и Мура

1.9.1. Эквивалентные автоматы

Пример 1.16. Вернемся к автомату Мили S_1 из примера 1.8 (см. табл. 1.14). Пусть на вход этого автомата поступает входное слово $\xi = (z_1 z_1 z_2 z_1 z_2 z_2)$. Тогда выходное слово $\omega = \lambda(a_1, \xi) = (w_1 w_2 w_1 w_1 w_1 w_2)$ (см. табл. 1.15).

Таблица 1.14. Таблица переходов и выходов автомата Мили S_1

	a_1	a_2	a_3
z_1	$a_3 w_1$	$a_1 w_1$	$a_1 w_2$
z_2	$a_1 w_1$	$a_3 w_2$	$a_2 w_1$

Таблица 1.15. Вычисление реакции автомата S_1 на входное слово ξ

z_1	z_1	z_2	z_1	z_2	z_2	
a_1	a_3	a_1	a_1	a_3	a_2	a_3
w_1	w_2	w_1	w_1	w_1	w_2	

Пример 1.17. Рассмотрим автомат Мура S_6 (табл. 1.16). Пусть по-прежнему на вход автомата подается слово $\xi = (z_1 z_1 z_2 z_1 z_2 z_2)$ (см. табл. 1.17). Заметим, что выходной сигнал $w_1 = \lambda(a_1)$ (получаемый в момент времени $t = 1$) не зависит от входного сигнала z_1 . Следовательно, он не связан со словом, поступающим на вход автомата, начиная с момента времени $t = 1$. Получаем, что реакции автоматов S_1 и S_6 , находящихся в начальных состояниях, на входное слово ξ совпадают с точностью до сдвига на 1 такт.

Таблица 1.16. Отмеченная таблица переходов автомата Мура S_6

	w_1	w_2	w_1	w_1	w_2
	a_1	a_2	a_3	a_4	a_5
z_1	a_4	a_4	a_1	a_2	a_2
z_2	a_1	a_1	a_5	a_3	a_3

Таблица 1.17. Вычисление реакции автомата S_6 на входное слово ξ

z_1	z_1	z_2	z_1	z_2	z_2	
a_1	a_4	a_2	a_1	a_4	a_3	a_5
w_1	w_1	w_2	w_1	w_1	w_1	w_2

Таким образом, в ответ на входное слово длины k автомат Мили выдает выходное слово длины k . Автомат Мура в той же ситуации выдает выходное слово длины k со сдвигом на один такт.

Два автомата S_A и S_B с одинаковыми входными и выходными алфавитами называются *эквивалентными*, если после установления их в начальные состояния их реакции на любое входное слово совпадают.

Теорема 1.1. *Для любого автомата Мили существует эквивалентный ему автомат Мура и, обратно, для любого автомата Мура существует эквивалентный ему автомат Мили.*

Опишем алгоритмы взаимной трансформации моделей Мили и Мура. При этом в автоматах Мура будем пренебрегать выходным сигналом $\lambda(a_1)$, связанным с начальным состоянием.

1.9.2. Преобразование автомата Мура в эквивалентный автомат Мили

Пусть задан автомат Мура

$$S_A = \{A_A, Z_A, W_A, \delta_A, \lambda_A, a_{1_A}\}.$$

Требуется получить автомат Мили

$$S_B = \{A_B, Z_B, W_B, \delta_B, \lambda_B, a_{1_B}\},$$

эквивалентный исходному.

Пусть $A_B = A_A$, $Z_B = Z_A$, $W_B = W_A$, $\delta_B = \delta_A$, $a_{1_B} = a_{1_A} = a_1$.

Функцию λ_B определим следующим образом: если в автомате Мура $\delta_A(a_m, z_f) = a_s$, $\lambda_A(a_s) = w_g$, то в автомате Мили

$$\lambda_B(a_m, z_f) = w_g = \lambda_A(\delta_A(a_m, z_f)). \quad (1.11)$$

На практике для построения автомата Мили будем руководствоваться следующими правилами:

1. При *графическом способе задания* соответствие между соседними вершинами графов автоматов представлено на рисунке 1.13.

2. При *табличном способе задания* таблица переходов автомата Мили S_B совпадает с таблицей переходов автомата Мура S_A . Таблица выходов автомата S_B получается из таблицы переходов заменой символа a_s , стоящего на пересечении строки

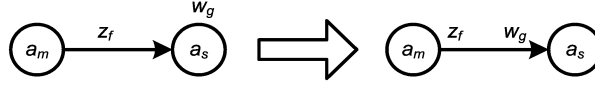


Рис. 1.13. Переход от автомата Мура к автомату Мили

z_f и столбца a_m , символом выходного сигнала w_g , отмечающего столбец a_s в таблице переходов автомата S_A .

Доказательство эквивалентности. Пусть символ z_f поступает на вход автомата Мура S_A , который находится в состоянии a_m . Следовательно, S_A перейдет в состояние $a_s = \delta_A(a_m, z_f)$ и выдаст сигнал $w_g = \lambda_A(a_s)$. Соответствующий автомат Мили S_B из состояния a_m также перейдет в состояние $a_s = \delta_B(a_m, z_f) = \delta_A(a_m, z_f)$ и выдаст тот же сигнал $w_g = \lambda_B(a_m, z_f) = \lambda_A(\delta_A(a_m, z_f)) = \lambda_A(a_s) = w_g$. Таким образом, для выходной последовательности длины 1 поведение автоматов S_1 и S_2 полностью совпадает. Далее по индукции получаем эквивалентность автоматов.

Упражнение 1.3. Постройте автомат Мили, эквивалентный автомату Мура S_2 из примера 1.9.

1.9.3. Преобразование автомата Мили в эквивалентный автомат Мура

Определим *преходящее состояние* как состояние, в которое, при представлении автомата в виде графа, не входит ни одна дуга, но которое имеет, по крайней мере, одну выходящую дугу. Далее, пока не оговорено противное, будем считать, что в автомате Мили *нет проходящих состояний*.

Пусть задан автомат Мили

$$S_A = \{A_A, Z_A, W_A, \delta_A, \lambda_A, a_{1_A}\},$$

при этом $|A_A| = M$. Требуется получить автомат Мура

$$S_B = \{A_B, Z_B, W_B, \delta_B, \lambda_B, a_{1_B}\},$$

эквивалентный исходному.

Пусть $Z_B = Z_A$, $W_B = W_A$.

Для определения A_B каждому состоянию $a_s \in A_A$ поставим в соответствие множество A_s всевозможных пар вида (a_s, w_g) , где w_g – выходной сигнал, приписанный входящей в a_s дуге. Таким образом,

$$A_s = \{(a_s, w_g) | \delta_A(a_m, z_f) = a_s \ \& \ \lambda_A(a_m, z_f) = w_g\}, \quad (1.12)$$

при этом мощность множества A_s равна числу различных выходных сигналов на дугах автомата S_A , входящих в состояние a_s . Например, состоянию, представленному на рисунке 1.14, соответствует множество $A_s = \{(a_s, w_1), (a_s, w_2), (a_s, w_3)\}$.

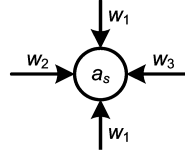


Рис. 1.14. Вершина графа автомата Мили

Определим A_B следующим образом:

$$A_B = \bigcup_{s=1}^M A_s. \quad (1.13)$$

Каждому состоянию $a_s^g = (a_s, w_g)$ автомата Мура S_B поставим в соответствие выходной сигнал w_g , то есть $\lambda_B(a_s^g) = w_g$.

Далее, пусть в автомате Мили S_A был переход $\delta_A(a_m, z_f) = a_s$ и при этом выдавался сигнал $\lambda_A(a_m, z_f) = w_g$, тогда в автомате Мура S_B будет переход из множества состояний A_m , порождаемых состоянием a_m , в состояние $(a_s, w_g) = a_s^g$ под действием входного сигнала z_f . Таким образом, $\delta_B(A_m, z_f) = a_s^g$ (см. рис. 1.15).

Так как при сравнении реакций автоматов S_A и S_B на входные слова не должен учитываться выходной сигнал в момент

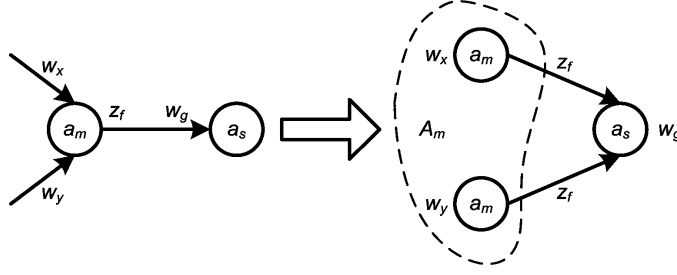


Рис. 1.15. Переход от автомата Мили к автомату Мура

времени $t = 1$, связанный с состоянием a_{1_B} автомата Мура S_B , то в качестве начального состояния a_{1_B} можно взять любое состояние множества A_1 , которое порождается начальным состоянием a_{1_A} автомата S_A .

Доказательство эквивалентности автоматов S_A и S_B аналогично предыдущему случаю.

Пример 1.18. Рассмотрим автомат Мили $S_1 = S_A$ из примера 1.8: $Z_A = \{z_1, z_2\}$, $W_A = \{w_1, w_2\}$, $A_A = \{a_1, a_2, a_3\}$, $a_{1_A} = a_1$. Построим эквивалентный ему автомат Мура.

$$A_1 = \{(a_1, w_1), (a_1, w_2)\} = \{b_1, b_2\},$$

$$A_2 = \{(a_2, w_1)\} = \{b_3\},$$

$$A_3 = \{(a_3, w_1), (a_3, w_2)\} = \{b_4, b_5\},$$

$$A_B = A_1 \cup A_2 \cup A_3 = \{b_1, b_2, b_3, b_4, b_5\},$$

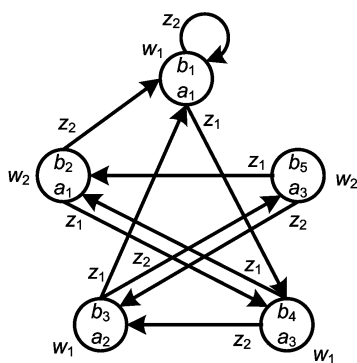
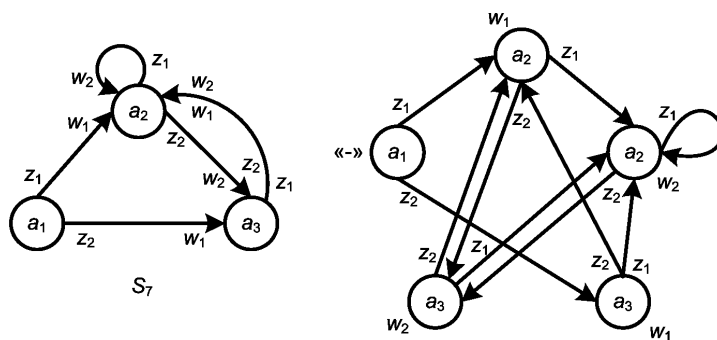
$$\lambda_B(b_1) = \lambda_B(b_3) = \lambda_B(b_4) = w_1,$$

$$\lambda_B(b_2) = \lambda_B(b_5) = w_2.$$

Граф полученного автомата Мура представлен на рисунке 1.16.

Пусть теперь в автомате Мили *допускается наличие переходящих состояний*. Переход от автомата Мили к автомату Мура рассмотрим на следующем примере.

Пример 1.19. Автомат Мили S_7 , содержащий переходящее состояние a_1 , и эквивалентный ему автомат Мура представлены на рисунке 1.17. При этом $A_B =$

Рис. 1.16. Автомат Мура, эквивалентный автомату Мили S_1 Рис. 1.17. Автомат Мили S_7 и эквивалентный ему автомат Мура

$$\{(a_2, w_1), (a_2, w_2), (a_3, w_1), (a_3, w_2), (a_1, -)\} = \{b_1, b_2, b_3, b_4, b_5\}.$$

Замечание 1.5. Возможен переход от С-автомата к эквивалентным автоматам Мили или Мура (с учетом возможных сдвигов сигналов во времени на 1 такт).

Замечание 1.6. При переходе от автомата Мура к эквивалентному автомату Мили число внутренних состояний не меняется. Обратный переход от автомата Мили к эквивалентному автомату Мура может привести к увеличению числа внутренних состояний. В итоге получаем два эквивалентных автомата Мура, число состояний которых в общем случае не совпадает. Аналогичный вывод справедлив и для модели Мили.

Таким образом, эквивалентные между собой автоматы могут иметь различное число состояний. В связи с этим возникает **задача:** *найти минимальный по числу состояний автомат в классе эквивалентных между собой автоматов.*

Теорема 1.2 (Мур). *Для любого абстрактного автомата S существует эквивалентный ему абстрактный автомат S_{min} с минимальным числом внутренних состояний.*

1.10. Минимизация полностью определенных автоматов

Автомат при своем первоначальном построении может быть излишне сложен. Если автомат является прообразом технического устройства, то целесообразно минимизировать количество состояний или, что то же самое, число технических компонент, из которых будет состоять проектируемое устройство.

1.10.1. Метод последовательных разбиений

Рассмотрим минимизацию полных автоматов Мили.

Идея метода такова: разобьем все состояния автомата на попарно непересекающиеся классы эквивалентных состояний и заменим каждый класс эквивалентности одним состоянием.

Два состояния a_m и a_s называются **эквивалентными** ($a_m \sim a_s$), если $\lambda(a_m, \xi) = \lambda(a_s, \xi)$ для любого входного слова ξ . Два состояния a_m и a_s **различимы**, если они не эквивалентны, то есть $a_m \not\sim a_s$.

Два состояния a_m и a_s **k -эквивалентны** ($a_m \overset{k}{\sim} a_s$), если $\lambda(a_m, \xi_k) = \lambda(a_s, \xi_k)$ для любых входных слов ξ_k длины k . Два состояния k -**различимы**, если они не k -эквивалентны, то есть $a_m \overset{k}{\not\sim} a_s$.

То, что определения \sim и $\overset{k}{\sim}$ задают отношения эквивалентности, то есть выполняется рефлексивность, транзитивность и симметричность, проверяется непосредственно. Поэтому множество состояний автомата A можно разбить на попарно непересекающиеся классы эквивалентности. Разбиения на классы эквивалентных и k -эквивалентных состояний будем обозначать π , π_k , соответственно.

Разбиение π позволяет определить *избыточные* состояния в A . Если два состояния a_m и a_s эквивалентны, то неважно находится автомат в состоянии a_m или a_s – на выход подается один и тот же сигнал. Более того, реакция автомата на входное слово не меняется. Таким образом, с точки зрения результата работы автомата данные состояния неразличимы, и одно из них избыточно, то есть может быть удалено из автомата.

Заметим, что если каждый класс эквивалентности содержит только один элемент, то множество A – **несократимо**.

Перейдем теперь непосредственно к алгоритму минимизации, в основе которого лежит последовательность разбиений. **Алгоритм Ауфенкампа – Хона** (или **метод последовательных разбиений**):

1. Найти последовательность разбиений $\pi_1, \pi_2, \dots, \pi_k, \pi_{k+1}$ множества A на классы 1-, 2-, ..., k -, $(k+1)$ -эквивалентных состояний. Разбиения строятся до тех пор, пока на каком-то $(k+1)$ -м шаге не окажется, что $\pi_{k+1} = \pi_k$. В этом случае можно показать, что $\pi_k = \pi$, то есть, k -эквивалентные состояния будут эквивалентными. Отметим, что число шагов k не превышает $(M-1)$, где $M = |A|$ – общее число состояний.

2. В каждом классе эквивалентности разбиения π выбрать

по одному элементу. Из этих элементов образовать новое множество состояний A' минимального автомата

$$S' = \{A', Z, W, \delta', \lambda', a'_1\} \sim S. \quad (1.14)$$

В качестве a'_1 выбрать одно из состояний, эквивалентных a_1 (как правило, берется $a'_1 = a_1$). Функцию переходов δ' и функцию выходов λ' определить на множестве $A' \times Z$ по следующему правилу: в таблицах переходов и выходов вычеркнуть столбцы, соответствующие состояниям, не вошедшим в A' .

Пример 1.20. Рассмотрим неминимальный автомат Мили S_8 (табл. 1.18). Построим эквивалентный ему минимальный автомат Мили, используя метод последовательных разбиений.

Таблица 1.18. Объединенная таблица переходов и выходов автомата

Мили S_8						
	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_{10}	a_{12}	a_5	a_7	a_3	a_7
z_2	a_5	a_7	a_6	a_{11}	a_9	a_{11}
z_1	w_1	w_1	w_2	w_2	w_1	w_2
z_2	w_2	w_2	w_1	w_1	w_2	w_1
a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	
a_3	a_{10}	a_7	a_1	a_5	a_2	z_1
a_6	a_4	a_6	a_8	a_9	a_8	z_2
w_1	w_1	w_2	w_2	w_2	w_2	z_1
w_2	w_2	w_1	w_1	w_1	w_1	z_2

Так как два состояния автомата 1-эквивалентны, если их реакции на всевозможные входные слова длины 1 совпадают, то соответствующие этим состояниям *столбцы в таблице выходов должны быть одинаковыми*. Таким образом, по таблице выходов получаем, что в автомате есть два класса 1-эквивалентных состояний B_1, B_2 . Тогда разбиение $\pi_1 = \{B_1, B_2\}$, где $B_1 = \{a_1, a_2, a_5, a_7, a_8\}$,

$B_2 = \{a_3, a_4, a_6, a_9, a_{10}, a_{11}, a_{12}\}$. Таблица разбиения π_1 строится по таблице переходов, в которой состояния заменяются соответствующими классами эквивалентности (см. табл. 1.19).

Таблица 1.19. Таблица разбиения π_1

	B_1				
	a_1	a_2	a_5	a_7	a_8
z_1	B_2	B_2	B_2	B_2	B_2
z_2	B_1	B_1	B_2	B_2	B_2

B_2							
a_3	a_4	a_6	a_9	a_{10}	a_{11}	a_{12}	
B_1	B_1	B_1	B_1	B_1	B_1	B_1	z_1
B_2	B_2	B_2	B_2	B_1	B_2	B_1	z_2

Заметим, что если $a_m \stackrel{1}{\sim} a_s$, то $a_m \stackrel{2}{\sim} a_s$ в том случае, когда любым входным сигналом эти состояния переводятся в 1-эквивалентные.

Проверим, например, что состояния a_1 и a_2 2-эквивалентны. Автомат в состоянии a_1 под действием входного сигнала z_1 выдает выходной сигнал w_1 и переходит в состояние a_{10} , из этого состояния под действием сигнала, например z_1 , выдаться сигнал w_2 . Далее, под действием сигнала z_1 автомат в состоянии a_2 выдает w_1 и переходит в a_{12} , а на следующем такте, опять же под действием сигнала z_1 , мы получим выходной сигнал w_2 . Аналогично можно получить, что для всех остальных двухбуквенных слов, составленных из z_1, z_2 , будут получаться одинаковые двухбуквенные слова, составленные из w_1, w_2 .

Итак, по таблице разбиения π_1 получаем разбиение $\pi_2 = \{C_1, C_2, C_3, C_4\}$, где $C_1 = \{a_1, a_2\}$, $C_2 = \{a_5, a_7, a_8\}$, $C_3 = \{a_3, a_4, a_6, a_9, a_{11}\}$, $C_4 = \{a_{10}, a_{12}\}$. Таблица разбиения π_2 строится также по таблице переходов (см. табл. 1.20).

Далее, по аналогии, строим разбиение $\pi_3 = \{D_1, D_2, D_3, D_4, D_5\}$, где $D_1 = \{a_1, a_2\}$, $D_2 = \{a_5, a_7\}$,

Таблица 1.20. Таблица разбиения π_2

	C_1		C_2		
	a_1	a_2	a_5	a_7	a_8
z_1	C_4	C_4	C_3	C_3	C_4
z_2	C_2	C_2	C_3	C_3	C_3

C_3					C_4		
a_3	a_4	a_6	a_9	a_{11}	a_{10}	a_{12}	
C_2	C_2	C_2	C_2	C_2	C_1	C_1	z_1
C_3	C_3	C_3	C_3	C_3	C_2	C_2	z_2

$D_3 = \{a_8\}$, $D_4 = \{a_3, a_4, a_6, a_9, a_{11}\}$, $D_5 = \{a_{10}, a_{12}\}$, и его таблицу (см. табл. 1.21).

Таблица 1.21. Таблица разбиения π_3

	D_1		D_2		D_3
	a_1	a_2	a_5	a_7	a_8
z_1	D_5	D_5	D_4	D_4	D_5
z_2	D_2	D_2	D_4	D_4	D_4

D_4					D_5		
a_3	a_4	a_6	a_9	a_{11}	a_{10}	a_{12}	
D_2	D_2	D_2	D_2	D_2	D_1	D_1	z_1
D_4	D_4	D_4	D_4	D_4	D_3	D_3	z_2

Оказывается, что π_4 будет совпадать с π_3 , что в общем-то уже видно по таблице π_3 . Отсюда следует, что $\pi_3 = \pi$.

Выберем произвольно из каждого класса разбиения π по одному состоянию: $A' = \{a_1, a_5, a_8, a_3, a_{10}\}$. Объединенная таблица переходов и выходов минимального автомата Мили примет вид, представленный в таблице 1.22.

Минимизация полных автоматов Мура проводится аналогично. Отличие заключается лишь в том, что на первом шаге

Таблица 1.22. Объединенная таблица переходов и выходов минимального автомата Мили

	a_1	a_5	a_8	a_3	a_{10}
z_1	a_{10}	a_3	a_{10}	a_5	a_1
z_2	a_5	a_3	a_3	a_3	a_8
z_1	w_1	w_1	w_1	w_2	w_2
z_2	w_2	w_2	w_2	w_1	w_1

строится разбиение π_0 на 0-эквивалентные состояния. Состояния a_m и a_s **0-эквивалентны** ($a_m \overset{0}{\sim} a_s$), если эти состояния одинаково отмечены: $\lambda(a_m) = \lambda(a_s)$.

Пример 1.21. Рассмотрим неминимальный автомат Мура S_9 (табл. 1.23). Очевидно, что, к примеру, состояния a_1, a_2, a_8

Таблица 1.23. Отмеченная таблица переходов автомата Мура S_9

	w_1	w_1	w_3	w_3	w_3	w_2
	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_{10}	a_{12}	a_5	a_7	a_3	a_7
z_2	a_5	a_7	a_6	a_{11}	a_9	a_{11}
w_3	w_1	w_2	w_2	w_2	w_2	
a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	
a_3	a_{10}	a_7	a_1	a_5	a_2	z_1
a_6	a_4	a_6	a_8	a_9	a_8	z_2

0-эквивалентны. В соответствии с алгоритмом Ауфенкампа-Хона получаем последовательность разбиений:

1. Разбиение $\pi_0 = \{B_1, B_2, B_3\}$, где $B_1 = \{a_1, a_2, a_8\}$, $B_2 = \{a_6, a_9, a_{10}, a_{11}, a_{12}\}$, $B_3 = \{a_3, a_4, a_5, a_7\}$.
2. Разбиение $\pi_1 = \{C_1, C_2, C_3, C_4\}$, где $C_1 = \{a_1, a_2, a_8\}$, $C_2 = \{a_6, a_9, a_{11}\}$, $C_3 = \{a_{10}, a_{12}\}$, $C_4 = \{a_3, a_4, a_5, a_7\}$.
3. Далее, несложно показать, что $\pi_2 = \pi_1$.

Минимальный автомат Мура принимает вид, представленный таблицей 1.24.

Таблица 1.24. Отмеченная таблица переходов минимального автомата Мура

	w_1	w_2	w_2	w_3
	a_1	a_6	a_{10}	a_3
z_1	a_{10}	a_3	a_1	a_3
z_2	a_3	a_6	a_1	a_6

Описанный алгоритм последовательных разбиений можно применить и для *минимизации полного С-автомата*. В этом случае под *одноэквивалентными* будем понимать состояния, которые одинаково отмечены и имеют одинаковые столбцы в таблице выходов.

Пример 1.22. Рассмотрим неминимальный С-автомат S_{10} (табл. 1.25). Последовательность разбиений при минимизации:

Таблица 1.25. Отмеченная таблица переходов и выходов С-автомата S_{10}

	u_1	u_1	u_3	u_3	u_3	u_2
	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_{10}	a_{12}	a_5	a_7	a_3	a_7
z_2	a_5	a_7	a_6	a_{11}	a_9	a_{11}
z_1	w_1	w_1	w_2	w_2	w_1	w_2
z_2	w_2	w_2	w_1	w_1	w_2	w_1

u_3	u_1	u_2	u_2	u_2	u_2	
a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	
a_3	a_{10}	a_7	a_1	a_5	a_2	z_1
a_6	a_4	a_6	a_8	a_9	a_8	z_2
w_1	w_1	w_2	w_2	w_2	w_2	z_1
w_2	w_2	w_1	w_1	w_1	w_1	z_2

1. Разбиение $\pi_1 = \{B_1, B_2, B_3, B_4\}$, где $B_1 = \{a_1, a_2, a_8\}$, $B_2 = \{a_3, a_4\}$, $B_3 = \{a_5, a_7\}$, $B_4 = \{a_6, a_9, a_{10}, a_{11}, a_{12}\}$.
2. Разбиение $\pi_2 = \{C_1, C_2, C_3, C_4, C_5, C_6\}$, где $C_1 = \{a_1, a_2\}$,

$C_2 = \{a_8\}$, $C_3 = \{a_3, a_4\}$, $C_4 = \{a_5, a_7\}$, $C_5 = \{a_6, a_9, a_{11}\}$,
 $C_6 = \{a_{10}, a_{12}\}$.

3. Далее, $\pi_3 = \pi_2$.

Минимальный С-автомат будет иметь вид, представленный таблицей 1.26.

Таблица 1.26. Отмеченная таблица переходов и выходов минимального С-автомата

	u_1	u_1	u_3	u_3	u_2	u_2
	a_1	a_8	a_3	a_5	a_6	a_{10}
z_1	a_{10}	a_{10}	a_5	a_3	a_5	a_1
z_2	a_5	a_3	a_6	a_6	a_6	a_8
z_1	w_1	w_1	w_2	w_1	w_2	w_2
z_2	w_2	w_2	w_1	w_2	w_1	w_1

1.10.2. Таблица пар состояний

Рассмотрим другой способ построения минимального автомата. Состояния a_i и a_j **явно различимы**, если различаются соответствующие им столбцы в таблице выходов.

Пример 1.23. В автомате Мили S_8 из примера 1.20 состояния $\{a_1, a_2, a_5, a_7, a_8\}$ явно различимы с состояниями $\{a_3, a_4, a_6, a_9, a_{10}, a_{11}, a_{12}\}$. Будем обозначать такие подмножества множества состояний A как A_{r_1} , A_{r_2} и т. д.

Идея алгоритма заключается в последовательном рассмотрении всевозможных пар состояний и исключении тех из них, в которых состояния не являются эквивалентными. Пары (a_i, a_i) не рассматриваются.

В качестве основы алгоритма строится **таблица пар состояний**. Принцип построения таблицы следующий:

- 1) так как явно различимые состояния не эквивалентны, то такие пары в таблицу не включаются;
- 2) для каждой пары отводится строка, для каждого входного сигнала – столбец. В клетках таблицы, в соответствии с

таблицей переходов, указывается пара состояний, в которую переходит автомат из данной пары состояний (по строке) при данном входном воздействии (по столбцу).

Далее производится последовательное исключение пар неэквивалентных состояний, при этом исключаемые пары отмечаются. Правило исключения пар основывается на следующем факте: *если $a_i \sim a_j$, то и $\delta(a_i, z_f) \sim \delta(a_j, z_f)$. Таким образом, если $\delta(a_i, z_f) \not\sim \delta(a_j, z_f)$, то и $a_i \not\sim a_j$.*

На первом шаге отмечаются пары, которые под действием некоторого входного сигнала переходят в пару явно различных состояний, то есть в пару, отсутствующую в первом столбце таблицы и состоящую из различных состояний.

Отмеченные пары состоят из неэквивалентных состояний, и на следующем шаге отмечаются все те пары, которые переходят в пары, отмеченные на предыдущем шаге и т.д.

Процесс заканчивается тогда, когда среди неотмеченных пар уже нет таких, которые можно отметить в соответствии с выбранным правилом.

Неотмеченные пары представляют собой попарно эквивалентные состояния.

Объединяя эквивалентные состояния в классы эквивалентности и выбирая из каждого класса по одному представителю, строится минимальный автомат (аналогично предыдущему способу минимизации). Необходимо отметить, что каждое состояние, не представленное ни в одной из неотмеченных пар, образует отдельный класс.

Пример 1.24. Найдем классы эквивалентных состояний автомата Мили S_8 из примера 1.20 при помощи таблицы пар состояний (см. табл. 1.27).

Отметим, что для заполнения первого столбца таблицы достаточно перебрать всевозможные пары состояний из каждого подмножества A_{r_i} .

Рассматривая неотмеченные пары состояний, получаем следующие классы эквивалентности: $\{a_1, a_2\}$, $\{a_5, a_7\}$, $\{a_3, a_4, a_6, a_9, a_{11}\}$, $\{a_{10}, a_{12}\}$, $\{a_8\}$.

Таблица 1.27. Таблица пар состояний автомата Мили S_8

	z_1	z_2
	(a_1, a_2)	(a_{10}, a_{12})
✓	(a_1, a_5)	(a_{10}, a_3)
✓	(a_1, a_7)	(a_{10}, a_3)
✓	(a_1, a_8)	(a_{10}, a_{10})
✓	(a_2, a_5)	(a_{12}, a_3)
✓	(a_2, a_7)	(a_{12}, a_3)
✓	(a_2, a_8)	(a_{12}, a_{10})
	(a_5, a_7)	(a_3, a_3)
✓✓	(a_5, a_8)	(a_3, a_{10})
✓✓	(a_7, a_8)	(a_3, a_{10})
	(a_3, a_4)	(a_5, a_7)
	(a_3, a_6)	(a_5, a_7)
	(a_3, a_9)	(a_5, a_7)
✓	(a_3, a_{10})	(a_5, a_1)
	(a_3, a_{11})	(a_5, a_5)
✓	(a_3, a_{12})	(a_5, a_2)
	(a_4, a_6)	(a_7, a_7)
	(a_4, a_9)	(a_7, a_7)
✓	(a_4, a_{10})	(a_7, a_1)
	(a_4, a_{11})	(a_7, a_5)
✓	(a_4, a_{12})	(a_7, a_2)
	(a_6, a_9)	(a_7, a_7)
✓	(a_6, a_{10})	(a_7, a_1)
	(a_6, a_{11})	(a_7, a_5)
✓	(a_6, a_{12})	(a_7, a_2)
✓	(a_9, a_{10})	(a_7, a_1)
	(a_9, a_{11})	(a_7, a_5)
✓	(a_9, a_{12})	(a_7, a_2)
✓	(a_{10}, a_{11})	(a_1, a_5)
	(a_{10}, a_{12})	(a_1, a_2)
✓	(a_{11}, a_{12})	(a_5, a_2)

Упражнение 1.4. Оцените трудоемкость алгоритма Ауфенкампа-Хона.

Упражнение 1.5. Оцените трудоемкость алгоритма минимизации полного автомата, использующего таблицу пар состояний.

1.11. Минимизация частично определенных автоматов

Задача минимизации частичного автомата значительно сложнее задачи минимизации полного автомата и ставится несколько по-другому.

В этом параграфе будем рассматривать автоматы Мили. Все результаты несложно перенести и на модель Мура.

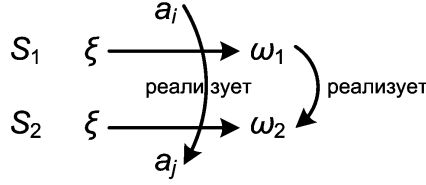
Будем говорить, что *последовательность* δ_i *реализует* последовательность δ_j , если δ_j можно доопределить так, что она совпадет с δ_i .

Пример 1.25. Последовательность $\delta_i = (54 - 4 - 62)$ реализует последовательность $\delta_j = (54 - - - 6-)$.

Входная последовательность называется *допустимой* для внутреннего состояния автомата a_i , если можно найти порождаемую ею выходную последовательность, то есть существует последовательность внутренних состояний. Отметим, что для полного автомата любая входная последовательность является допустимой.

Состояние a_i автомата S_1 *реализует* состояние a_j частичного автомата S_2 , если для любой допустимой входной последовательности, отвечающая ей выходная последовательность автомата S_1 будет реализовывать выходную последовательность автомата S_2 , отвечающую этому же входному слову (рис. 1.18).

Автомат S_1 *реализует* частичный автомат S_2 , если для любого состояния автомата S_2 найдется реализующее его состояние автомата S_1 . В свою очередь, будем говорить, что

Рис. 1.18. Состояние a_i реализует состояние a_j

частичный **автомат** S_2 **реализуется** автоматом S_1 .

Задача: для заданного частичного автомата S найти реализующий его автомат S_{min} , обладающий минимально возможным числом состояний.

Прежде чем перейти к описанию метода минимизации частичных автоматов, дадим еще несколько определений.

Две **последовательности совместимы**, если существует третья последовательность, реализующая каждую из них.

Пример 1.26. Последовательности $(122 - 3 - -5)$ и $(-2 - 435 - -)$ совместимы, так как, к примеру, последовательность $(122435 - 5)$ реализует каждую из них.

Состояния a_i и a_j некоторого частичного автомата **совместимы**, если для любой допустимой входной последовательности соответствующие этим состояниям выходные последовательности будут совместимы.

Замечание 1.5. Совместимые состояния – это некоторый аналог эквивалентных состояний в полном автомате. Отношение совместимости рефлексивно, симметрично, но не обязательно транзитивно, то есть это отношение **толерантности**. Отсюда следует, что все совместимые между собой состояния объединяются в классы толерантности или **классы совместимости**: A_1, A_2, \dots, A_m . Классы совместимости образуют покрытие множества состояний: $\bigcup_{i=1}^m A_i = A$.

Замечание 1.6. Классы совместимости – это некоторый

аналог классов эквивалентных состояний в полном автомате, но $A_i \cap A_j$ не обязательно равно пустому множеству.

Замечание 1.7. Автомат S^* , реализующий частичный автомат S , называется *квазиэквивалентным* автомату S . Отношение квазиэквивалентности рефлексивно, транзитивно, но не симметрично. Допустима запись вида: $S \subset S^*$.

Рассмотрим алгоритм минимизации частичных автоматов, основанный на построении *таблицы пар состояний и матрицы совместимости*.

Таблица пар состояний строится по аналогии с таблицей для полных автоматов (см. § 1.10). Первый столбец содержит пары таких состояний, для которых при любом входном сигнале отсутствуют различные выходы, то есть соответствующие этой паре состояний столбцы в таблице выходов совместимы (1-совместимость).

В остальных столбцах записываются пары состояний, в которые перейдет автомат под воздействием заданного входного сигнала. При этом клетки, соответствующие запрещенным входам для данной пары состояний (не определено новое состояние хотя бы для одного состояния из пары), заполняются прочерками и при исключении пар не учитываются. Исключение пар производится по тому же принципу, что и для полных автоматов.

Неотмеченные пары представляют собой попарно совместимые состояния.

Пример 1.27. Пусть задан частичный автомат Мили S_{11} . Для простоты, внутренние состояния и выходные сигналы автомата будем обозначать арабскими цифрами (см. табл. 1.28, 1.29).

Таблица 1.28. Таблица переходов частичного автомата Мили S_{11}

	0	1	2	3	4	5
a	1	—	3	5	5	—
b	—	4	1	5	5	—

Таблица 1.29. Таблица выходов частичного автомата Мили S_{11}

	0	1	2	3	4	5
a	0	—	0	—	0	—
b	—	1	0	0	—	—

Таблица 1.30. Таблица пар состояний частичного автомата S_{11}

	a	b
(0, 1)	—	—
✓ (0, 2)	(1, 3)	—
(0, 3)	(1, 5)	—
(0, 4)	(1, 5)	—
(0, 5)	—	—
(1, 4)	—	(4, 5)
(1, 5)	—	—
(2, 3)	(3, 5)	(1, 5)
(2, 4)	(3, 5)	(1, 5)
(2, 5)	—	—
(3, 4)	(5, 5)	(5, 5)
(3, 5)	—	—
(4, 5)	—	—

Таблица пар состояний для частичного автомата S_{11} представлена в таблице 1.30.

Далее определим **множество совместимости** (**C-множество**) как некоторое множество состояний частичного автомата S , на которое налагаются следующие условия:

- 1) каждая пара состояний из C-множества совместима;
- 2) это множество не является подмножеством другого C-множества, содержащего большее число состояний.

Найдем все C-множества частичного автомата S . В качестве исходных данных рассматриваются пары совместимых состояний, полученные по таблице пар состояний. На каждом шаге строится некоторое семейство множеств состояний.

1. Положим $k = 1$. Первое семейство множеств состояний

состоит из всех пар совместимых состояний автомата S и из отдельных состояний, не принадлежащих ни одной из этих пар; $k := 2$.

2. Для каждого множества состояний $A_i = \{a_{i_1}, \dots, a_{i_h}\}$, содержащегося в $(k-1)$ -м семействе, выполняем следующие операции:

2.1. Определяем l состояний a_{j_1}, \dots, a_{j_l} таких, что эти l состояний не принадлежат данному множеству A_i и образуют совместимую пару с каждым состоянием из этого множества.

2.2. Заменяем множество $\{a_{i_1}, \dots, a_{i_h}\}$ l множествами $\{a_{i_1}, \dots, a_{i_h}, a_{j_d}\}_{d=1}^l$ (если $l = 0$, то замена не происходит).

3. Исключаем из нового семейства множеств все одинаковые множества (кроме одного представителя) и все множества, являющиеся подмножествами других множеств. Получаем k -е семейство множеств.

4. Если k -е семейство отличается от $(k-1)$ -го, то $k := k+1$ и переходим на шаг 2, иначе – алгоритм завершен.

Чтобы выполнить этот алгоритм, построим *матрицу совместимости* C_S :

$$[C_S]_{ij} = \begin{cases} 1, & \text{если } (a_i, a_j) \text{ или } (a_j, a_i) - \text{пара} \\ & \text{совместимых состояний из таблицы пар;} \\ 0, & \text{иначе.} \end{cases}$$

По построению, если в каждой из строк, отвечающих состояниям из множества $\{a_{i_1}, \dots, a_{i_h}\}$, содержится «1» в каком-нибудь столбце a_{j_d} , то a_{j_d} образует пары совместимых состояний с каждым элементом из этого множества. Следовательно, данное множество заменяется множеством $\{a_{i_1}, \dots, a_{i_h}, a_{j_d}\}$.

Итак, мы получили все C -множества частичного автомата S : A_1, \dots, A_m . Заметим, что множества A_i ($i = 1, \dots, m$) – это не что иное, как рассмотренные ранее классы совместимости.

Пример 1.28. Найдём все C -множества частичного автомата S_{11} . Используя таблицу пар состояний, полученную в примере 1.27, заполним матрицу совместимости (см. табл. 1.31) и выпишем семейство множеств состояний на первом шаге.

Таблица 1.31. Матрица совместимости частичного автомата S_{11}

	0	1	2	3	4	5
0	0	1	0	1	1	1
1	1	0	0	0	1	1
2	0	0	0	1	1	1
3	1	0	1	0	1	1
4	1	1	1	1	0	1
5	1	1	1	1	1	0

Для $k = 1$ получаем следующее семейство множеств: $\{\{0, 1\}, \{0, 3\}, \{0, 4\}, \{0, 5\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$.

Пусть $k = 2$. Строки 0 и 1 содержат «1» в 4-м и 5-м столбцах, следовательно, множеству $\{0, 1\}$ соответствуют множества $\{0, 1, 4\}, \{0, 1, 5\}$, при этом само множество $\{0, 1\}$ и множества $\{0, 4\}, \{0, 5\}, \{1, 4\}, \{1, 5\}$ будут исключены. Повторяя подобные рассуждения для всех множеств первого шага (с учетом исключаемых), получаем новое семейство множеств: $\{\{0, 1, 4\}, \{0, 1, 5\}, \{0, 3, 4\}, \{0, 3, 5\}, \{0, 4, 5\}, \{1, 4, 5\}, \{2, 3, 4\}, \{2, 4, 5\}, \{3, 4, 5\}\}$.

Для $k = 3$ семейство множеств имеет вид: $\{\{0, 1, 4, 5\}, \{0, 3, 4, 5\}, \{2, 3, 4, 5\}\}$.

На следующем шаге ($k = 4$) семейство множеств не изменится, следовательно, получены все С-множества автомата. Обозначим их: $A_1 = \{0, 1, 4, 5\}$, $A_2 = \{0, 3, 4, 5\}$, $A_3 = \{2, 3, 4, 5\}$.

Завершают минимизацию следующие рассуждения. Пусть A_1, A_2, \dots, A_n – множества состояний автомата S такие, что каждое состояние автомата включено, по крайней мере, в одно множество A_l ($l = 1, \dots, n$), то есть $\bigcup_{l=1}^n A_l = A$. Множество $\{A_1, \dots, A_n\}$ назовем *группировкой* (покрытием), n – мощность группировки.

Пара состояний (a_i, a_j) называется *сопряженной*, если оба состояния принадлежат, по крайней мере, одному из мно-

жеств A_l . Если пара (a_i, a_j) не является сопряженной, то она называется **разобщенной**.

Правильная группировка – это группировка, удовлетворяющая двум условиям:

1) для любой сопряженной пары (a_i, a_j) реакции автомата S , который находится в одном из этих состояний, на любой допустимый входной сигнал z_f одинаковы: $\lambda(a_i, z_f) = \lambda(a_j, z_f)$ (если один или оба выходных сигнала не определены, то это условие несущественно для данной пары);

2) состояния, в которые переходит автомат из состояний a_i, a_j под действием сигнала z_f , либо одинаковы, либо составляют сопряженную пару; иными словами, если $a_k = \delta(a_i, z_f)$, $a_s = \delta(a_j, z_f)$, то либо $a_k = a_s$, либо пара (a_k, a_s) – сопряженная (если одно или оба состояния перехода не определены, то это условие несущественно для данной пары).

Можно показать, что справедливы следующие утверждения.

Утверждение 1.2. Все C -множества частичного автомата S образуют правильную группировку.

Утверждение 1.3. Из правильной группировки $\{A_1, \dots, A_n\}$ получается автомат, реализующий исходный, если множествам A_1, \dots, A_n поставить в соответствие внутренние состояния a_1, \dots, a_n нового автомата.

Утверждение 1.4. Автомат S_{min} соответствует наименьшей (по мощности) правильной группировке.

Таким образом, автомат S_{min} соответствует наименьшей правильной группировке, построенной из полученных C -множеств. Иными словами, необходимо отобрать минимальное число C -множеств, которые образуют покрытие множества A и являются правильной группировкой.

Вообще говоря, автомат, реализующий исходный, можно получить, если всем C -множествам A_1, \dots, A_m поставить в соответствие состояния a_1, \dots, a_m . Но такой автомат не всегда будет минимальным.

Пример 1.29. Проведем минимизацию частичного автомата S_{11} . Так как мощность множества состояний A равна 6, а

мощности С-множеств, полученных в примере 1.28, совпадают и равны 4, то мощность минимальной правильной группировки должна быть не менее 2.

Возьмем С-множества $A_1 = \{0, 1, 4, 5\}$ и $A_3 = \{2, 3, 4, 5\}$. Так как $A_1 \cup A_3 = \{0, 1, 2, 3, 4, 5\} = A$, то $\{A_1, A_3\}$ – покрытие множества A . Проверим, что группировка $\{A_1, A_3\}$ правильная. Рассмотрим, в какие множества состояний (множества «переходов») перейдут множества A_1 и A_3 под действием различных входных сигналов и каковы будут соответствующие множества выходных сигналов (см. табл. 1.32, 1.33).

Таблица 1.32. Множества «переходов»

	$A_1 = \{0, 1, 4, 5\}$	$A_3 = \{2, 3, 4, 5\}$
a	$\{1, 5\}$	$\{3, 5\}$
b	$\{4, 5\}$	$\{1, 5\}$

Таблица 1.33. Множества выходных сигналов

	$A_1 = \{0, 1, 4, 5\}$	$A_3 = \{2, 3, 4, 5\}$
a	$\{0\}$	$\{0\}$
b	$\{1\}$	$\{0\}$

Так как состояния из множеств A_1 и A_3 переходят в состояния, которые образуют подмножества этих же множеств: $\{1, 5\} \subset A_1$, $\{3, 5\} \subset A_3$, $\{4, 5\} \subset A_1 \cap A_3$, $\{1, 5\} \subset A_1$, то выполнено второе условие правильной группировки (всевозможные пары состояний во множествах «переходов» будут сопряженными, более того, они будут совместимыми, так как A_1 и A_3 являются С-множествами). Анализ множеств выходных сигналов показывает, что выполнено и первое условие правильной группировки (все множества выходных сигналов содержат по одному элементу).

Таким образом, $\{A_1, A_3\}$ – минимальная правильная группировка. Поставим в соответствие множеству A_1 состояние «0» минимального автомата, множеству A_3 – состояние «1». Тогда можно выписать таблицы переходов и выходов искомого автомата S_{min} (см. табл. 1.34, 1.35).

Таблица 1.34. Таблица переходов автомата S_{min}

	0	1
a	0	1
b	0	0

Таблица 1.35. Таблица выходов автомата S_{min}

	0	1
a	0	0
b	1	0

Заметим, что, если автомат находится в состоянии «0» и на вход подается сигнал « b », то в качестве нового состояния можно выбрать не «0», а «1» (так как $\{4, 5\} \subset A_1 \cap A_3$).

Упражнение 1.6. Покажите, что отношение совместимости состояний для частичных автоматов не является транзитивным (приведите примеры).

Уражнение 1.7. Оцените трудоемкость описанного алгоритма минимизации частичного автомата.

1.12. Операции над автоматами

1.12.1. Разбиение автомата (декомпозиция)

Все состояния автомата можно разделить на три группы, в соответствии с их *типом* (см. рис. 1.19):

Преходящее состояние. Как было определено ранее, это состояние, в которое, при представлении автомата в виде графа, не входит ни одна дуга, но которое имеет, по крайней мере, одну выходящую дугу.

Тупиковое состояние. В это состояние можно перейти, по крайней мере, из одного другого состояния, но выйти из него нельзя.

Изолированное состояние. В это состояние нельзя перейти ни из какого другого состояния, и из этого состояния нельзя перейти ни в какое другое.

Подавтомат — это автомат, определенный некоторым подмножеством множества состояний исходного автомата.

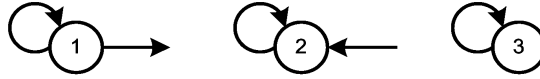


Рис. 1.19. Типы состояний автомата: 1 – переходящее, 2 – тупиковое, 3 – изолированное

та. *Тупиковый, изолированный, переходящий подавтоматы* определяются аналогично одноименным состояниям (см. рис. 1.20).

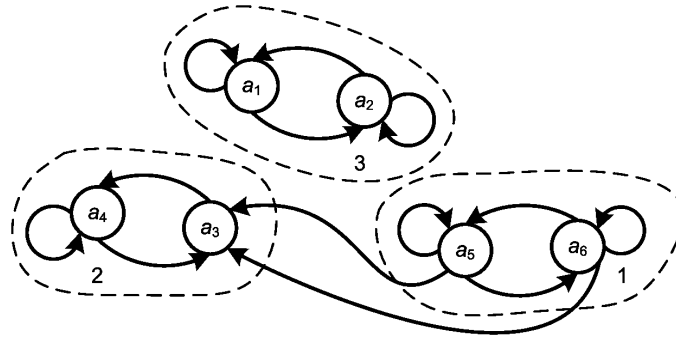
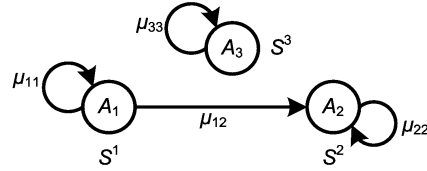


Рис. 1.20. Подавтоматы: 1 – переходящий, 2 – тупиковый, 3 – изолированный

Очевидно, что если начальное состояние автомата S принадлежит множеству состояний A_i , которое задает тупиковый или изолированный подавтомат, то S можно упростить исключением всех состояний, которые не принадлежат множеству A_i , и всех дуг, начинающихся в этих состояниях.

Пусть S^1 , S^2 , S^3 – переходящий, тупиковый и изолированный подавтоматы автомата S (см. рис. 1.21). Эти подавтоматы характеризуются тремя множествами состояний A_1 , A_2 , A_3 и соответствующими матрицами соединений (см. § 1.5) μ_{11} , μ_{22} , μ_{33} . Матрица соединений μ_{12} характеризует переходы из состояний переходящего подавтомата S^1 к состояниям тупикового подавтомата S^2 .

Рис. 1.21. Подавтоматы автомата S

Несложно показать, что выделение преходящих, тупиковых, и изолированных подавтоматов соответствует разбиению множества состояний A на непересекающиеся подмножества. В нашем случае $A = A_1 \cup A_2 \cup A_3$ и $A_i \cap A_j = \emptyset$, при $i \neq j$. Тогда матрица соединений автомата S может быть представлена в виде таблицы 1.36. Отсюда следует, что *разбиение автомата S на подавтоматы S^i можно осуществить преобразованием его матрицы соединений путем перестановки соответствующих строк и столбцов.*

Таблица 1.36. Матрица соединений автомата S , разбитого на преходящий, тупиковый и изолированный подавтоматы S^1 , S^2 , S^3

	A_1	A_2	A_3
A_1	μ_{11}	μ_{12}	0
A_2	0	μ_{22}	0
A_3	0	0	μ_{33}

1.12.2. Соединение автоматов (композиция)

1. Параллельное соединение

Пусть даны два автомата:

$$S_1 = \{A_1, Z, W_1, \delta_1, \lambda_1, a_1\},$$

$$S_2 = \{A_2, Z, W_2, \delta_2, \lambda_2, a_2\}.$$

Входной алфавит Z у этих автоматов общий и имеется некоторое устройство ϕ (автомат без памяти), объединяющее выходы автоматов:

$$\phi : W_1 \times W_2 \rightarrow W.$$

В результате параллельного соединения получаем новый автомат (см. рис. 1.22)

$$S = \{A, Z, W, \delta, \lambda, a\},$$

для которого заданы следующие параметры:

- 1) $A = A_1 \times A_2$, то есть это всевозможные пары состояний автоматов S_1 и S_2 : $A = \{a_m = (a_{m_1}, a_{m_2}) | a_{m_1} \in A_1, a_{m_2} \in A_2\}$;
- 2) входной алфавит Z ;
- 3) выходной алфавит $W = \phi(W_1 \times W_2)$;
- 4) функция переходов $\delta(a_m, z_j) = (\delta_1(a_{m_1}, z_j), \delta_2(a_{m_2}, z_j))$;
- 5) функция выходов $\lambda(a_m, z_j) = \phi(\lambda_1(a_{m_1}, z_j), \lambda_2(a_{m_2}, z_j))$.

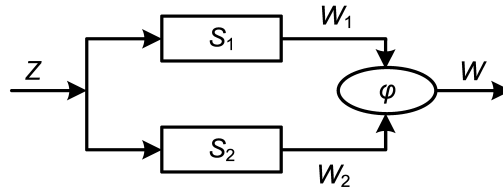


Рис. 1.22. Параллельное соединение автоматов

2. Последовательное соединение

При последовательном соединении выходы первого автомата являются входами второго автомата. Даны два автомата:

$$S_1 = \{A_1, Z, W_1, \delta_1, \lambda_1, a_1\},$$

$$S_2 = \{A_2, W_1, W, \delta_2, \lambda_2, a_2\}.$$

В результате последовательного соединения получаем новый автомат (см. рис. 1.23)

$$S = \{A, Z, W, \delta, \lambda, a\},$$

для которого основные параметры заданы следующим образом:

- 1) $A = A_1 \times A_2 = \{a_m = (a_{m_1}, a_{m_2}) | a_{m_1} \in A_1, a_{m_2} \in A_2\}$;
- 2) входной алфавит Z ;
- 3) выходной алфавит W ;
- 4) $\delta(a_m, z_j) = (\delta_1(a_{m_1}, z_j), \delta_2(a_{m_2}, \lambda_1(a_{m_1}, z_j)))$ – функция переходов;
- 5) $\lambda(a_m, z_j) = \lambda_2(a_{m_2}, \lambda_1(a_{m_1}, z_j))$ – функция выходов.

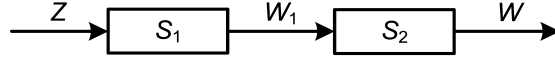


Рис. 1.23. Последовательное соединение автоматов

3. Соединение с обратной связью

Пусть даны два автомата:

$$S_1 = \{A_1, Z_1, W_1, \delta_1, \lambda_1, a_1\},$$

$$S_2 = \{A_2, W_1, W_2, \delta_2, \lambda_2, a_2\}.$$

Имеется некоторый функциональный преобразователь γ , представляющий собой автомат без памяти. Этот автомат реализует отображение:

$$\gamma : Z \times W_2 \rightarrow Z_1.$$

Заметим, что в случае соединения с обратной связью, по крайней мере, один из автоматов S_1 или S_2 должен быть автоматом Мура. Пусть S_2 – автомат Мура: $W_2 = \lambda_2(A_2)$. В результате соединения с обратной связью получаем новый автомат (см. рис. 1.24)

$$S = \{A, Z, W, \delta, \lambda, a\},$$

для которого заданы параметры:

- 1) $A = A_1 \times A_2 = \{a_m = (a_{m_1}, a_{m_2}) | a_{m_1} \in A_1, a_{m_2} \in A_2\}$;
- 2) входной алфавит Z ;

- 3) выходной алфавит $W = W_1$;
- 4) функция переходов $\delta(a_m, z_j) = (\delta_1(a_{m_1}, \gamma(z_j, \lambda_2(a_{m_2}))), \delta_2(a_{m_2}, \lambda_1(a_{m_1}, \gamma(z_j, \lambda_2(a_{m_2}))))))$;
- 5) $\lambda(a_m, z_j) = \lambda_1(a_{m_1}, \gamma(z_j, \lambda_2(a_{m_2})))$ – функция выходов.

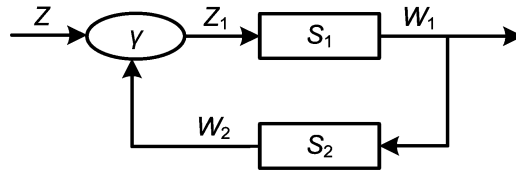


Рис. 1.24. Соединение автоматов с обратной связью

1.13. Классификация

В заключении главы перечислим основные параметры, по которым осуществляется классификация автоматов:

1. По закону функционирования:
 - автомат Мура;
 - автомат Мили.
2. По конечности множеств A, Z, W :
 - конечные автоматы;
 - бесконечные автоматы.
3. По объему памяти:
 - автомат с памятью (последовательный);
 - автомат без памяти (комбинационный).
4. По степени раскрытия структуры:
 - абстрактные автоматы (структура не раскрыта);
 - структурные автоматы (раскрыты детали структуры).
5. По полноте переходов:
 - полностью определенные автоматы;
 - частично определенные автоматы.

6. По стабильности периода следования входных сигналов:

- синхронные автоматы (период следования входных сигналов постоянен);
- асинхронные автоматы (период следования входных сигналов – переменная величина).

7. По однозначности переходов:

- детерминированные автоматы;
- недетерминированные автоматы.

8. По отношению между автоматами:

- подавтоматы;
- надавтоматы.

И т. д.

Глава 2

Структурный синтез

2.1. Элементы и базис

Первый этап проектирования технического устройства – это абстрактный синтез. На этом этапе автомат задается таблицей, графом либо матрицей. Заканчивается абстрактный синтез минимизацией числа состояний автомата.

Вторым этапом проектирования является структурный синтез. *Цель* этого этапа – построение схемы, реализующей автомат, на основе композиции автоматных элементов, принадлежащих заданному конечному множеству. В структурном автомате учитывается структура входных и выходных сигналов автомата, а также его внутреннее устройство на уровне логических схем.

В действительности технические устройства состоят из конструктивных единиц – так называемых **реальных элементов**. В основе их функционирования лежит непосредственное использование некоторых физических явлений.

Автоматный элемент (или просто *элемент*) – это абстрактная модель реального элемента.

Элементный базис (или просто *базис*) – это конечное множество элементов (множество типов элементов), из кото-

рых можно синтезировать автомат по заданной функциональной модели. Очевидно, что в автомат может входить несколько элементов одного типа.

Основные *требования*, налагаемые на базис – это полнота и эффективность.

Базис называется **структурно полным** (или просто **полным**) относительно некоторого класса автоматов, если в нем может быть синтезирован любой автомат этого класса. Напомним, что мы рассматриваем класс дискретных автоматов.

Замечание 2.1. В отличие от элементного базиса, функциональным базисом называется система логических функций, через которые может быть выражена любая функция алгебры логики. Такая система логических функций называется **функционально полной**. Напомним, что примерами функционально полных систем являются $\{\langle \text{И} \rangle, \langle \text{ИЛИ} \rangle, \langle \text{НЕ} \rangle\}$, $\{\langle \text{И} \rangle, \langle \text{НЕ} \rangle\}$, $\{\langle \text{ИЛИ} \rangle, \langle \text{НЕ} \rangle\}$, $\{\langle \text{И-НЕ} \rangle\}$, $\{\langle \text{ИЛИ-НЕ} \rangle\}$.

Более **эффективным** будет тот базис, синтезируемые в котором автоматы являются лучшими в каком-либо смысле: проще, дешевле, надежнее и т. д. Эффективность базиса зависит и от применяемых методов синтеза.

Рассмотрим теперь вопрос о том, какие элементы могут входить в базис. Выделяют две группы:

- 1) логические элементы;
- 2) элементы памяти.

Логический элемент – это элементарный комбинационный автомат, функциональные свойства которого представляются достаточно простой логической функцией. Логический элемент имеет количество входов, равное числу аргументов функции, и только один выход. Отметим, что здесь и далее под **входом** или **выходом** будем понимать входной или выходной полюс элемента.

Основными логическими элементами являются (см. рис. 2.1):

- 1) элемент $\langle \text{НЕ} \rangle$ (инвертор);
- 2) элемент $\langle \text{И} \rangle$ (конъюнктор);

- 3) элемент «ИЛИ» (дизъюнктор);
- 4) элемент «И-НЕ»;
- 5) элемент «ИЛИ-НЕ».

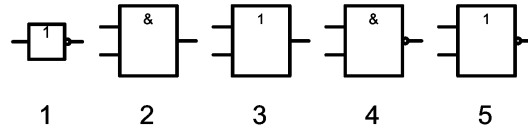


Рис. 2.1. Логические элементы: (1) – «НЕ», (2) – «И», (3) – «ИЛИ», (4) – «И-НЕ», (5) – «ИЛИ-НЕ»

На рисунке 2.1 все логические элементы, за исключением инвертора, имеют два входа. Еще раз отметим, что количество входов этих элементов может быть увеличено.

Элемент памяти (запоминающий элемент) – это элементарный последовательный автомат.

Основными элементами памяти являются (см. рис. 2.2):

1) **элемент задержки** – имеет один вход, один выход и осуществляет задержку поступившего на вход сигнала на один такт. Ниже приведена отмеченная таблица переходов элемента задержки (см. табл. 2.1).

2) **триггер**. Различают следующие типы триггеров:

2.1) **триггер со счетным входом** – имеет один вход и один выход и задается таблицей 2.2. Исходя из этой таблицы, получаем, что если на вход триггера со счетным входом подать «0», то состояние триггера, а следовательно, и выходной сигнал, не изменятся; если же на вход подать «1», то состояние триггера сменится на противоположное, и на следующем такте изменится выходной сигнал.

2.2) **триггер с отдельными выходами** – имеет два входа и один выход и задается таблицей 2.3. Таким образом, если на оба входа триггера с отдельными входами подать нули, то состояние триггера, а следовательно, и выходной сигнал, не изменятся. Если на первый вход подать «0», а на второй – «1», то триггер из любого состояния перейдет в состояние «1», и на следующем такте выдаст выходной сигнал, равный

Таблица 2.1. Отмеченная таблица переходов элемента задержки

	0	1
	0	1
0	0	0
1	1	1

Таблица 2.2. Отмеченная таблица переходов триггера со счетным входом

	0	1
	0	1
0	0	1
1	1	0

«1». Если же ситуация обратная: на первый вход подана «1», а на второй – «0», то триггер из любого состояния перейдет в состояние «0», и на следующем такте выдаст «0». Наконец, ситуация, когда на оба входа подаются единицы, является запрещенной – в этом случае поведение триггера не определено. Реакцию триггера с раздельными входами на различные входные сигналы можно представить в виде таблицы 2.4.

Таблица 2.3. Отмеченная таблица переходов триггера с раздельными выходами

	0	1
	0	1
0 0	0	1
0 1	1	1
1 0	0	0
1 1	—	—

Таблица 2.4. Реакция триггера с раздельными входами на различные входные сигналы

Вход	Выход
0 0	$0 \vee 1$
0 1	1
1 0	0
1 1	?

Замечание 2.2. Очевидно, что для правильной работы автомата нельзя разрешить, чтобы сигналы на входе запоминающих элементов непосредственно участвовали в образовании выходных сигналов, которые по цепям обратной связи подавались бы в тот же самый момент времени на эти входы.

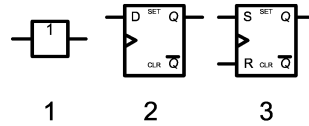


Рис. 2.2. Элементы памяти: (1) – элемент задержки, (2) – триггер со счетным входом, (3) – триггер с отдельными входами

Следовательно, запоминающие элементы должны быть автоматами Мура.

Более того, можно показать, что для синтеза автоматов с минимальным числом элементов памяти, необходимо в качестве таких элементов выбирать автоматы Мура, являющиеся **полными**, то есть имеющие:

1) **полную систему переходов**: для любой пары состояний (a_m, a_s) существует входной сигнал z_f такой, что $\delta(a_m, z_f) = a_s$, иными словами, в каждом столбце таблицы переходов встречаются все состояния автомата;

2) **полную систему выходов**: каждому состоянию поставлен в соответствие свой выходной сигнал, отличный от выходных сигналов других состояний \implies число выходных сигналов равно числу состояний \implies состояния можно отождествить с выходными сигналами \implies в элементах памяти используются одни и те же обозначения для выходных сигналов и состояний.

Теорема 2.1 (о структурной полноте). Если элементный базис содержит элемент памяти (автомат Мура с нетривиальной памятью (более одного состояния), обладающий полной системой переходов и полной системой выходов) и какую-либо функционально полную систему логических элементов, то он является структурно полным [2].

2.2. Схемы

Рассмотрим некоторую совокупность автоматных элементов (элементы могут повторяться). Пусть P – множество всех входов (входных полюсов), Q – множество всех выходов (выходных полюсов). Далее, пусть $X \subseteq P$, $Y \subseteq Q$. Зададим отображение $\Omega : P \setminus X \rightarrow Q$, которое определяет связи между элементами множеств P и Q . Полученную таким образом структуру будем называть **сетью**, при этом X – входные полюсы сети, Y – выходные полюсы сети. **Логическая схема** (или просто **схема**) – это графическое представление сети. Тем самым, термины «сеть» и «схема» в нашем понимании тождественны.

Можно показать, что *структура любого дискретного автомата может быть представлена некоторой схемой* [4].

Обратное, в общем случае, неверно. Для обратного соответствия необходимо, чтобы по схеме можно было однозначно определить функциональные свойства автомата.

Пример 2.1. На рисунке 2.3 (1) изображена схема, которая не является автоматом. Действительно, пусть $x = 0$. Если $y = 0$, то $y = 1$ и если $y = 1$, то $y = 0$. Следовательно, невозможно определить значение выходной переменной y . Это противоречие можно разрешить, если добавить в схему элемент памяти, например, элемент задержки (см. рис. 2.3 (2)).

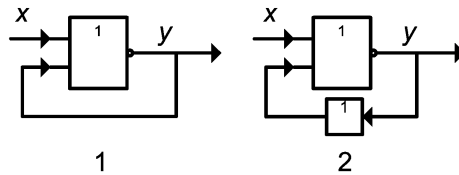


Рис. 2.3. Примеры схем: 1 – схема, но не автомат; 2 – схема, которой соответствует автомат

Правильной схемой назовем схему, которую можно интерпретировать, как структуру дискретного автомата. *Далее будем рассматривать только правильные схемы.*

Линейно-упорядоченная схема – это схема, которая не имеет контуров.

К-ярусная схема – это схема, элементы которой могут быть разбиты на классы (ярусы) с номерами $1, 2, \dots, k$ так, что любая межэлементная связь связывает выходной полюс элемента i -го яруса с входным полюсом элемента $(i + 1)$ -го яруса.

Комбинационная схема – это линейно-упорядоченная схема, составленная только из логических элементов. Структура комбинационного автомата представляется комбинационной схемой [4]. Комбинационная схема реализует столько логических функций, сколько имеет выходных полюсов.

В общем случае логическая схема допускает наличие контуров и элементов памяти. Простейший пример схемы, содержащей контуры, представлен на рисунке 2.4. Нетрудно убедиться, что эта схема оказывается триггером с отдельными входами. Отметим, что один и тот же тип триггера может быть представлен различными схемами, в зависимости от элементного базиса [5].

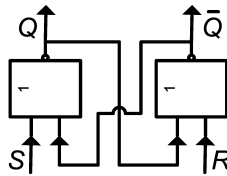


Рис. 2.4. Схема триггера с отдельными входами

2.3. Канонический метод структурного синтеза

Пусть дан некоторый элементный базис, который состоит из автоматных элементов: $\{S_1, \dots, S_n\} = \mathcal{B}$. Рассмотрим совокупность автоматных элементов S_1, \dots, S_k такую, что $\forall i \ S_i \in \mathcal{B}$.

Объединим автоматы S_1, \dots, S_k в систему совместно работающих автоматов, тем самым получим схему (сеть) такую, что в каждый момент автоматного времени:

- 1) на все входные полюсы схемы подается набор входных сигналов – структурный входной сигнал схемы;
- 2) со всех выходных полюсов схемы снимается набор выходных сигналов – структурный выходной сигнал схемы.

Положим, что в каждый момент автоматного времени структурный выходной сигнал схемы однозначно определяется

- 1) поступившей к этому времени конечной последовательностью структурных входных сигналов;
- 2) начальными состояниями входящих в схему автоматов.

Построенную схему можно рассматривать как некоторый **структурный** автомат S , при этом данную схему будем называть логической схемой автомата S . Иными словами, структурный автомат S представляет собой композицию автоматов S_1, \dots, S_k .

Для дальнейших построений опишем основные параметры абстрактного и структурного автоматов (см. табл. 2.5).

В *абстрактном C -автомате*:

- один входной канал, на который поступают сигналы из алфавита $Z = \{z_1, \dots, z_F\}$;
- два выходных канала, на которые поступают сигналы из алфавитов $W = \{w_1, \dots, w_G\}$ и $U = \{u_1, \dots, u_H\}$.

В *структурном автомате*:

- L входных каналов x_1, \dots, x_L ;
- $N + D$ выходных каналов y_1, \dots, y_N и r_1, \dots, r_D .

На эти каналы поступают сигналы из структурного алфавита автомата. Будем рассматривать двоичный структурный алфавит $\{0, 1\}$

Так как число различных булевых векторов длины L равно 2^L , то для кодирования всех входных сигналов абстрактного алфавита необходимо, чтобы выполнялось неравенство: $2^L \geq F$. Следовательно

$$L \geq \lceil \log_2 F \rceil, \quad (2.1)$$

Таблица 2.5. Входные и выходные сигналы в абстрактном и структурном автоматах

Сигнал в абстрактном алфавите	Бинарный вектор – код этого сигнала	Комментарий
z_f	$(e_{f_1}, \dots, e_{f_L})$	$e_{f_l} \in \{0, 1\}, f \in \{1, \dots, F\}$ e_{f_l} – сигнал на канале x_l
w_g	$(e_{g_1}, \dots, e_{g_N})$	$e_{g_n} \in \{0, 1\}, g \in \{1, \dots, G\}$ e_{g_n} – сигнал на канале y_n
u_h	$(e_{h_1}, \dots, e_{h_D})$	$e_{h_d} \in \{0, 1\}, h \in \{1, \dots, H\}$ e_{h_d} – сигнал на канале r_d

где скобки $\lceil \dots \rceil$ обозначают округление до большего целого. Аналогично:

$$N \geq \lceil \log_2 G \rceil, \quad (2.2)$$

$$D \geq \lceil \log_2 H \rceil. \quad (2.3)$$

Канонический метод структурного синтеза заключается в следующем:

1. В качестве базиса выбираются *элементы памяти* и *логические элементы* (теоретически, структурный автомат можно получить и как композицию других элементарных автоматов). Тем самым, обоснованием метода служит теорема о структурной полноте.

2. Логическая схема C -автомата представляется в виде **структурной схемы**, состоящей из трех частей (см. рис. 2.5):

- 1) память;
- 2) комбинационная схема КС1;
- 3) комбинационная схема КС2.

Память автомата состоит из предварительно выбранных элементов памяти – элементарных автоматов Мура. Пусть $\{\Pi_1, \dots, \Pi_I\}$ – выбранные элементы памяти. Состояние a_m абстрактного автомата ($a_m \in \{a_1, \dots, a_M\}$) кодируется вектором

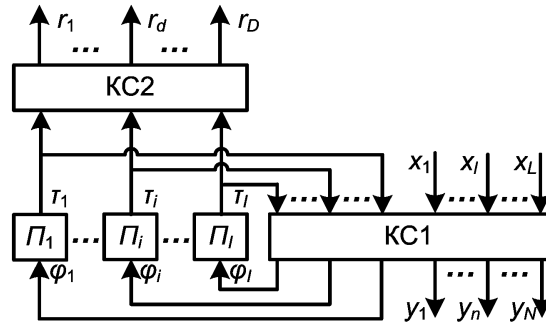


Рис. 2.5. Структурная схема С-автомата

состояний структурного автомата $(e_{m_1}, \dots, e_{m_I})$, где e_{m_i} – состояние элемента памяти Π_i (см. табл. 2.6). Очевидно, что

$$I \geq \lceil \log_{\Theta} M \rceil, \quad (2.4)$$

где Θ – число состояний элемента памяти. Будем считать, что все Π_i одинаковы. Заметим, что в большинстве случаев алфавит состояний элементов памяти двоичный.

Таблица 2.6. Состояния в абстрактном и структурном автоматах

Состояние в абстрактном алфавите	Вектор – код этого состояния	Комментарий
a_m	$(e_{m_1}, \dots, e_{m_I})$	$e_{m_i} \in \{0, 1, \dots, (\Theta - 1)\},$ $m \in \{1, \dots, M\}$

Таким образом, переходу С-автомата из состояния a_m в состояние a_s под действием входного сигнала z_f с выдачей выходного сигнала типа Мили w_g отвечает переход структурного автомата из состояния $(e_{m_1}, \dots, e_{m_I})$ в состояние $(e_{s_1}, \dots, e_{s_I})$ под действием входного сигнала $(e_{f_1}, \dots, e_{f_L})$ с выдачей выходного сигнала $(e_{g_1}, \dots, e_{g_N})$.

Изменение состояний элементов памяти на таком переходе происходит под действием сигналов $\varphi_1, \dots, \varphi_I$, снимаемых с выхода КС1. После перехода в состояние a_s автомат выдает сигнал типа Мура u_h , или в структурном автомате – $(e_{h_1}, \dots, e_{h_D})$, все время, пока находится в этом состоянии.

Комбинационная схема КС1 служит для формирования выходных сигналов типа Мили и входных сигналов элементов памяти, схема КС2 – для формирования выходных сигналов типа Мура.

Если синтезируется автомат Мили, то подсхемы КС2 в автомате нет. Если синтезируется автомат Мура, то нет выходных каналов y_1, \dots, y_N .

Соответствие между каналами абстрактного и структурного автоматов представлено в таблице 2.7.

Таблица 2.7. Входы и выходы абстрактного и структурного автоматов

Абстрактный автомат	Структурный автомат
Внутреннее состояние (a)	Выходы (они же состояния) элементов памяти – входы КС1 и КС2 (τ_1, \dots, τ_I) Входы элементов памяти – выходы КС1 ($\varphi_1, \dots, \varphi_I$)
Вход (z)	L входных каналов (x_1, \dots, x_L)
Выход 1-го типа (w)	N выходных каналов 1-го типа (y_1, \dots, y_N)
Выход 2-го типа (u)	D выходных каналов 2-го типа (r_1, \dots, r_D)

Напомним, что абстрактный С-автомат описывается следующими функциями:

$$\begin{cases} a = \delta(a, z); \\ w = \lambda_1(a, z); \\ u = \lambda_2(a), \end{cases} \quad (2.5)$$

Возвращаясь к каноническому методу получаем, что *после*

выбора элементов памяти и кодирования состояний и сигналов, синтез структурного автомата сводится к синтезу логической схемы, которая реализует систему логических функций:

$$\left\{ \begin{array}{l} \varphi_1 = \varphi_1(\tau_1, \dots, \tau_I, x_1, \dots, x_L); \\ \dots \\ \varphi_I = \varphi_I(\tau_1, \dots, \tau_I, x_1, \dots, x_L); \end{array} \right\} \delta$$

$$\left\{ \begin{array}{l} y_1 = y_1(\tau_1, \dots, \tau_I, x_1, \dots, x_L); \\ \dots \\ y_N = y_N(\tau_1, \dots, \tau_I, x_1, \dots, x_L); \end{array} \right\} \lambda_1 \quad (2.6)$$

$$\left\{ \begin{array}{l} r_1 = r_1(\tau_1, \dots, \tau_I); \\ \dots \\ r_D = r_D(\tau_1, \dots, \tau_I). \end{array} \right\} \lambda_2$$

Эта система функций (система уравнений) называется **канонической** [1, 2]. В структурном автомате функция $\varphi = (\varphi_1, \dots, \varphi_I)$ называется **функцией возбуждения** памяти автомата, а функция $\tau = (\tau_1, \dots, \tau_I)$ – **функцией обратной связи** от памяти автомата к КС.

Подводя итог, еще раз перечислим **основные этапы канонического метода структурного синтеза**:

1. Выбор базиса.
2. Кодирование состояний и сигналов автомата.
3. Построение канонической системы логических функций.
4. Построение логической схемы автомата.

2.4. Функция входов элемента памяти

Более подробно разберем способы описания элементов памяти.

Элемент памяти можно рассматривать на абстрактном и на структурном уровнях. Пусть дан некоторый элемент памяти Π , абстрактная модель которого представлена на рисунке 2.6(1). Напомним, что элемент памяти – это полный автомат Мура, в котором выходы отождествлены с состояниями.

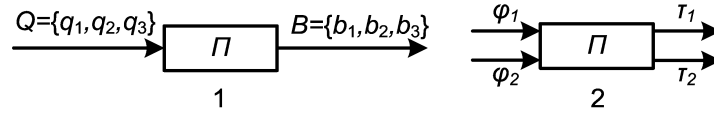


Рис. 2.6. Элемент памяти: 1 – абстрактный, 2 – структурный

Пусть, к примеру, абстрактный элемент памяти Π задается таблицей переходов 2.8.

Определим для абстрактного элемента памяти **функцию входов**. Эта функция представляет собой зависимость входного сигнала в момент времени t от выходных сигналов в моменты времени t и $(t+1)$: $q(t) = \mu(b(t), b(t+1))$. Данную функцию зададим таблично ($q_f = \mu(b_m, b_s)$), используя таблицу переходов (см. табл. 2.9).

Таблица 2.8. Таблица переходов абстрактного элемента памяти Π

	b_1	b_2	b_3
q_1	b_1	b_2	b_3
q_2	b_2	b_3	b_1
q_3	b_3	b_1	b_2

Таблица 2.9. Таблица функции входов абстрактного элемента памяти Π

b_m	q_f	b_s
b_1	q_1	b_1
b_1	q_2	b_2
b_1	q_3	b_3
b_2	q_3	b_1
b_2	q_1	b_2
b_2	q_2	b_3
b_3	q_2	b_1
b_3	q_3	b_2
b_3	q_1	b_3

Итак, абстрактный автомат Π имеет три входных и три выходных сигнала. Эти сигналы должны быть закодированы. При двоичном структурном алфавите структурный автомат

П будет иметь два входных и два выходных канала, так как три различных сигнала можно закодировать двумя битами (см. рис. 2.6(2)). Закодируем входные и выходные сигналы абстрактного элемента памяти (см. табл. 2.10). Тогда функция входов структурного элемента памяти будет определяться таблицей 2.11.

Таблица 2.10. Кодирование входных и выходных сигналов абстрактного элемента памяти П

	φ_1	φ_2		τ_1	τ_2
q_1	0	0	b_1	0	0
q_2	0	1	b_2	0	1
q_3	1	0	b_3	1	0

Таблица 2.11. Таблица функции входов структурного элемента памяти П

τ_1	τ_2	φ_1	φ_2	τ_1	τ_2
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	1	1	0	0	0
0	1	0	0	0	1
0	1	0	1	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	0	0	1	0

В дальнейшем подобные функции входов элементов памяти понадобятся для построения функций возбуждения памяти, входящих в каноническую систему уравнений.

Отметим, что в общем случае, если абстрактный элемент памяти имеет p входных сигналов $\{q_1, \dots, q_p\}$ и h выходных сигналов (или состояний) $\{b_1, \dots, b_h\}$, то число его входных и выходных каналов должно быть $K \geq \lceil \log_\eta p \rceil$, $T \geq \lceil \log_\nu h \rceil$, где основания логарифмов совпадают с числом различных букв в

структурных алфавитах входа и выхода. Чаще всего используется двоичный структурный алфавит и в качестве элементов памяти берутся автоматы с двумя состояниями.

2.5. Пример канонического метода структурного синтеза

¹Пусть необходимо синтезировать частичный С-автомат S , заданный таблицами 2.12, 2.13.

Таблица 2.12. Таблица переходов С-автомата

	a_1	a_2	a_3
z_1	a_2	—	a_1
z_2	a_3	a_1	—
z_3	a_2	a_3	a_3

Таблица 2.13. Отмеченная таблица выходов С-автомата

	u_1	u_2	u_1
	a_1	a_2	a_3
z_1	w_3	—	w_2
z_2	w_4	w_3	—
z_3	w_2	w_1	w_3

1. Выбор базиса. В качестве базиса возьмем функционально полную систему логических элементов «И», «ИЛИ» «НЕ» и элемент памяти П, заданный таблицей переходов 2.14.

Таблица 2.14. Таблица переходов абстрактного элемента памяти П

	b_1	b_2
q_1	b_1	b_2
q_2	b_2	b_1

2. Кодирование. Используем структурный алфавит $\{0, 1\}$. Результаты кодирования сигналов элемента памяти и С-автомата представлены в таблицах 2.15 – 2.22.

¹Этот и ряд других примеров взяты из [1].

С-автомат насчитывает 3 состояния. Элемент памяти П имеет два состояния. Следовательно, учитывая неравенство $I \geq \lceil \log_2 3 \rceil$, получаем $I = 2$. Таким образом, структурный автомат S будет иметь два элемента памяти Π_1 и Π_2 .

Для кодирования 3-х входных сигналов С-автомата потребуется $L = 2$ ($L \geq \lceil \log_2 3 \rceil$) входных каналов x_1 и x_2 .

Для кодирования 4-х выходных сигналов 1-го типа С-автомата необходимо $N = 2$ ($N \geq \lceil \log_2 4 \rceil$) выходных каналов 1-го типа y_1 и y_2 . Для кодирования 2-х выходных сигналов 2-го типа С-автомата необходимо $D = 1$ ($D \geq \lceil \log_2 2 \rceil$) выходных каналов 2-го типа r .

Таблица переходов структурного автомата S (см. табл. 2.23, 2.24) получается из таблицы переходов абстрактного С-автомата путем замены состояний и сигналов их кодами. Отмеченная таблица выходов структурного автомата S (см. табл. 2.25, 2.26) получается аналогично из соответствующей таблицы абстрактного С-автомата.

Таблица 2.15. Кодирование входных сигналов элемента

памяти П	
	φ
q_1	0
q_2	1

Таблица 2.16. Кодирование выходных сигналов элемента

памяти П	
	τ
b_1	0
b_2	1

Таблица 2.17. Таблица переходов элемента памяти П

	0	1
0	0	1
1	1	0

Таблица 2.18. Таблица функции входов элемента памяти П

$\tau_{\text{исходное}}$	φ	$\tau_{\text{перехода}}$
0	0	0
0	1	1
1	1	0
1	0	1

Таблица 2.19. Кодирование состояний С-автомата

	τ_1	τ_2
a_1	0	0
a_2	0	1
a_3	1	1

Таблица 2.20. Кодирование входных сигналов С-автомата

	x_1	x_2
z_1	0	0
z_2	0	1
z_3	1	0

Таблица 2.21. Кодирование выходных сигналов 1-го типа С-автомата

	y_1	y_2
w_1	1	0
w_2	0	0
w_3	1	1
w_4	0	1

Таблица 2.22. Кодирование выходных сигналов 2-го типа С-автомата

	r
u_1	1
u_2	0

Таблица 2.23. Таблица переходов структурного автомата S

	0 0	0 1	1 1
0 0	0 1	— —	0 0
0 1	1 1	0 0	— —
1 0	0 1	1 1	1 1

Таблица 2.24. «Эталонная» таблица переходов

	τ_1	τ_2
x_1 x_2	τ_1	τ_2

Таблица 2.25. Отмеченная таблица выходов структурного автомата S

	1	0	1
	0 0	0 1	1 1
0 0	1 1	— —	0 0
0 1	0 1	1 1	— —
1 0	0 0	1 0	1 1

Таблица 2.26. «Эталонная» таблица выходов

	r
	τ_1 τ_2
x_1 x_2	y_1 y_2

Структурная схема автомата S изображена на рисунке 2.7.

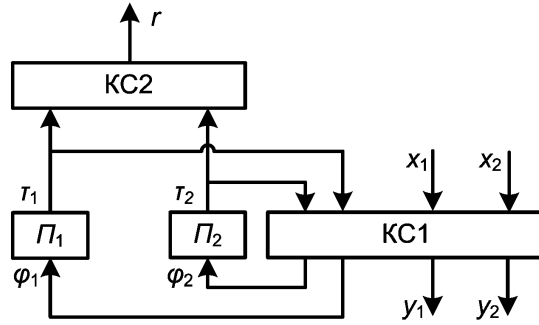


Рис. 2.7. Структурная схема автомата S

3. Каноническая система. Итак, после выбора элементов памяти и кодирования состояний и сигналов мы свели синтез C -автомата к синтезу двух комбинационных схем КС1 и КС2, которые реализуют функции:

$$\left\{ \begin{array}{l} \varphi_1 = \varphi_1(\tau_1, \tau_2, x_1, x_2); \\ \varphi_2 = \varphi_2(\tau_1, \tau_2, x_1, x_2); \end{array} \right\} \delta \quad \left\{ \begin{array}{l} y_1 = y_1(\tau_1, \tau_2, x_1, x_2); \\ y_2 = y_2(\tau_1, \tau_2, x_1, x_2); \end{array} \right\} \lambda_1 \quad \left\{ \begin{array}{l} r = r(\tau_1, \tau_2); \end{array} \right\} \lambda_2 \quad (2.7)$$

Функции y_i получаются непосредственно по отмеченной таблице выходов структурного автомата S как дизъюнкции конъюнкций, соответствующих наборам переменных $\tau_1, \dots, \tau_I, x_1, \dots, x_L$, на которых эти функции принимают значение единицы:

$$\begin{aligned} y_1 &= \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 \bar{x}_2 \vee \bar{\tau}_1 \tau_2 \bar{x}_1 x_2 \vee \bar{\tau}_1 \tau_2 x_1 \bar{x}_2 \vee \tau_1 \tau_2 x_1 \bar{x}_2; \\ y_2 &= \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 \bar{x}_2 \vee \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 x_2 \vee \bar{\tau}_1 \tau_2 \bar{x}_1 x_2 \vee \tau_1 \tau_2 x_1 \bar{x}_2. \end{aligned} \quad (2.8)$$

Функции r_i получаются аналогично – используется отмеченная таблица выходов структурного автомата S . Берутся дизъюнкции конъюнкций, соответствующие наборам переменных τ_1, \dots, τ_I , на которых эти функции принимают значение единицы:

$$r = \bar{\tau}_1 \bar{\tau}_2 \vee \tau_1 \tau_2. \quad (2.9)$$

Функции φ_i (функции возбуждения памяти) строятся по таблице переходов структурного автомата S с использованием таблицы функции входов структурного автомата Π (элемента памяти).

Разберем процесс построения функций φ_i на примере. Пусть автомат S находится в состоянии «01» и на вход поступил сигнал «10». В соответствии с таблицей переходов автомат перейдет в состояние «11». Этот переход складывается из двух переходов элементов памяти:

$$\Pi_1 : 0 \longrightarrow 1, \quad \Pi_2 : 1 \longrightarrow 1.$$

Переходы элементов памяти Π_1 и Π_2 происходят под действием сигналов функций возбуждения φ_1 и φ_2 , которые поступают на их входы. Воспользовавшись таблицей функции входов элемента памяти Π , получаем:

$$\Pi_1 : 0 \xrightarrow{\varphi_1=1} 1, \quad \Pi_2 : 1 \xrightarrow{\varphi_2=0} 1.$$

Таким образом, при построении *таблицы функций возбуждения памяти* в таблице переходов структурного автомата S вместо кода состояния перехода ставятся коды сигналов, поступивших на входы элементов памяти и приведших к этому состоянию (см. табл. 2.27, 2.28).

Используя полученную таблицу 2.27, функции φ_i строятся как дизъюнкции конъюнкций, соответствующих наборам переменных $\tau_1, \dots, \tau_I, x_1, \dots, x_L$, на которых эти функции принимают значение единицы:

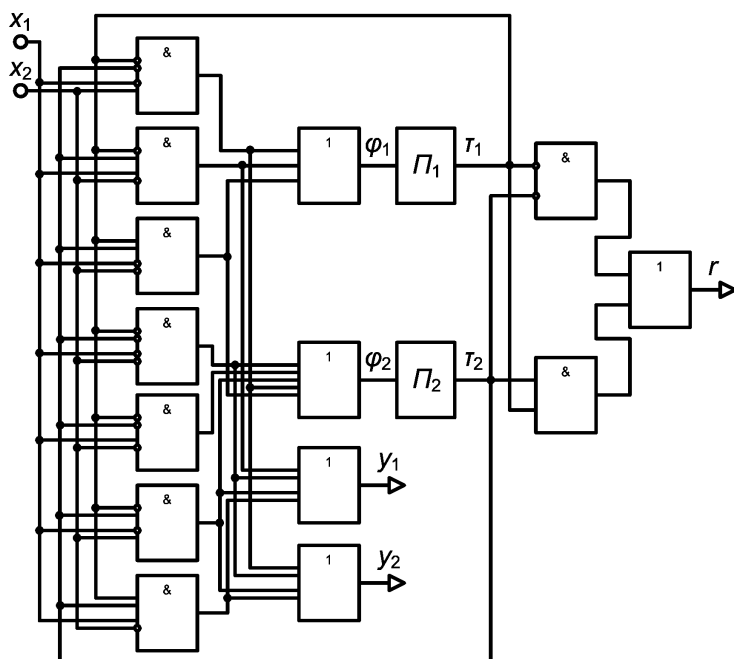
$$\begin{aligned} \varphi_1 &= \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 x_2 \vee \bar{\tau}_1 \tau_2 x_1 \bar{x}_2 \vee \tau_1 \tau_2 \bar{x}_1 \bar{x}_2; \\ \varphi_2 &= \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 \bar{x}_2 \vee \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 x_2 \vee \bar{\tau}_1 \bar{\tau}_2 x_1 \bar{x}_2 \vee \\ &\quad \bar{\tau}_1 \tau_2 \bar{x}_1 x_2 \vee \tau_1 \tau_2 \bar{x}_1 \bar{x}_2. \end{aligned} \quad (2.10)$$

Таблица 2.27. Таблица функций возбуждения памяти

	0	0	0	1	1	1
0	0	1	—	—	1	1
0	1	1	1	1	—	—
1	0	0	1	0	0	0

Таблица 2.28. «Эталонная» таблица функций возбуждения памяти

	τ_1	τ_2
x_1 x_2	φ_1	φ_2

Рис. 2.8. Логическая схема автомата S

4. Построение логической схемы. На основе полученных формул для функций $y_1, y_2, r, \varphi_1, \varphi_2$ строится логическая схема автомата S (см. рис. 2.8).

Заметим, что функции из канонической системы предварительно можно минимизировать, воспользовавшись любым методом минимизации в классе нормальных форм [3].

2.6. Выбор элементов памяти

Рассмотрим синтез автомата не на абстрактном элементе памяти Π , а на реальных устройствах: элементах задержки, триггерах со счетным входом и триггерах с отдельными входами. В качестве примера будем по-прежнему рассматривать S -автомат S из § 2.5.

Очевидно, что функции выходов y_1, \dots, y_N и r_1, \dots, r_D не зависят от конкретного вида элемента памяти и поэтому будут совпадать с теми, что получились в предыдущем параграфе (см. формулы 2.8, 2.9).

Изменяются функции возбуждения памяти $\varphi_1, \dots, \varphi_I$.

2.6.1. Синтез автомата на задержках

Напомним, что элемент задержки имеет один вход и один выход и осуществляет задержку на один такт (см. табл. 2.29, 2.30).

Функция входов элемента задержки проста: то, что поступает на вход элемента, то и становится значением памяти ($\varphi = \tau_{\text{перехода}}$). Таким образом, состояние, в которое перейдет элемент задержки, полностью совпадает с поступившим на его вход сигналом. Отсюда следует, что *таблица функций возбуждения памяти структурного автомата S полностью совпадает с таблицей переходов этого автомата*. Тогда

$$\begin{aligned}\varphi_1 &= \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 x_2 \vee \bar{\tau}_1 \tau_2 x_1 \bar{x}_2 \vee \tau_1 \tau_2 x_1 \bar{x}_2, \\ \varphi_2 &= \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 \bar{x}_2 \vee \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 x_2 \vee \bar{\tau}_1 \bar{\tau}_2 x_1 \bar{x}_2 \vee \\ &\quad \bar{\tau}_1 \tau_2 x_1 \bar{x}_2 \vee \tau_1 \tau_2 x_1 \bar{x}_2.\end{aligned}\tag{2.11}$$

Таблица 2.29. Таблица переходов элемента за-

держки		
	0	1
0	0	0
1	1	1

Таблица 2.30. Таблица функции входов элемента задержки

$\tau_{\text{исходное}}$	φ	$\tau_{\text{перехода}}$
0	0	0
0	1	1
1	0	0
1	1	1

Таблица 2.31. Таблица переходов триггера со счетным входом

	0	1
0	0	1
1	1	0

Таблица 2.32. Таблица функции входов триггера со счетным входом

$\tau_{\text{исходное}}$	φ	$\tau_{\text{перехода}}$
0	0	0
0	1	1
1	1	0
1	0	1

2.6.2. Синтез автомата на триггерах со счетным входом

Триггер со счетным входом имеет один вход и один выход и характеризуется таблицами 2.31, 2.32.

Анализируя функцию входов, получаем, что триггер со счетным входом работает как сумматор по модулю 2:

$$\varphi = \tau_{\text{исходное}} \oplus \tau_{\text{перехода}}.$$

Отсюда следует справедливость утверждения: *таблица функций возбуждения памяти структурного автомата S получается из его таблицы переходов по следующему правилу: элемент таблицы функций возбуждения равен покомпонентной сумме по модулю 2 кодов исходного состояния и состояния перехода (см. табл. 2.33, 2.34).*

Заметим, что в предыдущем параграфе в качестве элемен-

Таблица 2.33. Фрагмент таблицы переходов структурного автомата

		памяти	
		τ_1^j	τ_2^j
x_1^i	x_2^i	τ_1	τ_2

Таблица 2.34. Фрагмент таблицы функций возбуждения

		памяти	
		τ_1^j	τ_2^j
x_1^i	x_2^i	$\tau_1^j \oplus \tau_1$	$\tau_2^j \oplus \tau_2$

та памяти был выбран именно триггер со счетным входом. Исходя из этого, получаем, что функции возбуждения памяти определяются уравнениями 2.10.

2.6.3. Синтез автомата на триггерах с раздельными входами

Отличие от предыдущих элементов памяти заключается в том, что у данного триггера имеется два входа и, соответственно, четыре различные комбинации входных сигналов (см. табл. 2.35, 2.36). (Но выход по-прежнему один!) Напомним, что одна из комбинаций входных сигналов считается запрещенной и поэтому в таблице переходов всего три строки (можно было бы указать и четыре, но на сигнале «11» все переходы были бы не определены). Прочерк в таблице функции входов означает, что соответствующая переменная может принимать значения как «0», так и «1».

Таблица 2.35. Таблица переходов триггера с раздельными входами

	0	1
0 0	0	1
0 1	1	1
1 0	0	0

Таблица 2.36. Таблица функции входов триггера с раздельными входами

$\tau_{\text{исходное}}$	ψ	φ	$\tau_{\text{перехода}}$
0	—	0	0
0	0	1	1
1	1	0	0
1	0	—	1

Из таблицы функции входов следует, что при поступлении «1» на нулевой вход ψ триггер переходит в состояние «0» вне зависимости от того, в каком состоянии он находился. Аналогично, при поступлении «1» на единичный вход φ триггер в любом случае переходит в состояние «1».

Рассмотрим таблицу переходов структурного автомата S и подробно разберем случай, когда под действием входного сигнала «00» состояние «00» переходит в состояние «01». Этот переход складывается из переходов двух триггеров. Первый триггер из состояния «0» переходит в состояние «0». Тогда на вход триггера должен поступить сигнал «-0», что легко устанавливается из таблицы функции входов. Второй триггер переходит из состояния «0» в состояние «1». В этом случае на вход триггера должен поступить сигнал «01». Сигнал возбуждения памяти будет следующий: $\psi_1\varphi_1\psi_2\varphi_2 = (-001)$.

Если же переход происходит из состояния «00» в состояние «11», то $\psi_1\varphi_1\psi_2\varphi_2 = (0101)$.

Подставляя такие четверки в таблицу переходов структурного автомата S , мы получим таблицу функций возбуждения памяти (см. табл. 2.37).

Таблица 2.37. Таблица функций возбуждения памяти при синтезе автомата на триггерах с раздельными входами

	00	01	11
00	-001	—	1010
01	0101	-010	—
10	-001	010-	0-0-

Исходя из этой таблицы, функции возбуждения памяти принимают вид:

$$\begin{aligned}
 \psi_1 &= \tau_1\tau_2\bar{x}_1\bar{x}_2, \\
 \varphi_1 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2, \\
 \psi_2 &= \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee \tau_1\tau_2\bar{x}_1\bar{x}_2, \\
 \varphi_2 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2.
 \end{aligned}
 \tag{2.12}$$

Замечание 2.3. Во всех трех случаях логическая схема автомата строится аналогично схеме на рисунке 2.8.

Замечание 2.4. Рассмотренный метод синтеза может быть использован и для построения автомата с различными элементами памяти, то есть в случае, когда линейка Π не однотипна.

Пусть, например, структурный автомат имеет 2 входных сигнала и 6 элементов памяти: Π_1, Π_2 – элементы задержки, Π_3, Π_4 – триггеры со счетным входом, Π_5, Π_6 – триггеры с отдельными входами. Тогда построение таблицы функций возбуждения памяти иллюстрируют таблицы 2.38, 2.39.

Таблица 2.38. Столбец таблицы переходов структурного автомата

0	1	1	0	1	0
1	0	1	0	0	0
0	1	0	1	1	1

Таблица 2.39. Столбец таблицы функций возбуждения памяти

0	1	1	0	1	0
1	0	0	0	10	-0
0	1	1	1	0-	01

Замечание 2.5. В заключении параграфа отметим, что выражение функций мы получили в виде дизъюнктивных нормальных форм. Возможно получить и КНФ, проведя эквивалентные преобразования.

2.7. Графическая интерпретация канонического метода

Канонический метод структурного синтеза можно провести и графически. В этом случае структурный автомат представляется в виде графа. Предварительно строятся таблицы переходов и выходов структурного автомата.

По-прежнему будем рассматривать С-автомат S из § 2.5. Для удобства еще раз приведем его структурные таблицы (см. табл. 2.40, 2.41).

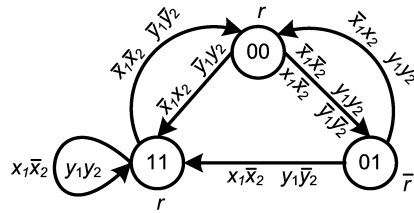
Таблица 2.40. Таблица переходов структурного автомата S

		0	0	0	1	1	1
0	0	0	1	—	—	0	0
0	1	1	1	0	0	—	—
1	0	0	1	1	1	1	1

Таблица 2.41. Отмеченная таблица выходов структурного автомата S

		та S					
		1		0		1	
0	0	1	1	—	—	0	0
0	1	0	1	1	1	—	—
1	0	0	0	1	0	1	1

Граф автомата строится в соответствии с общими правилами. Отличие заключается в том, что все сигналы на графе заменяются соответствующими переменными или их отрицаниями (см. рис. 2.9).

Рис. 2.9. Граф структурного автомата S

2.7.1. Синтез на задержках

Из таблицы функции входов элемента задержки получаем, что если $(\tau_{\text{перехода}} = 1)$, то и $(\varphi = 1)$. Отсюда следует, что если дуга графа входит в состояние, в котором элемент памяти Π_i принимает значение «1», то эта дуга отмечается символом φ_i .

Иными словами, каждой дуге приписано столько φ_i , сколько единиц в коде состояния, в которое входит дуга (см. рис. 2.10).

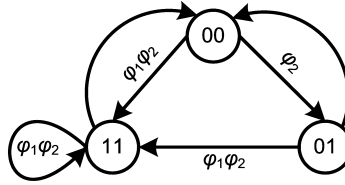


Рис. 2.10. Функции возбуждения памяти при синтезе на задержках

2.7.2. Синтез на триггерах со счетным входом

Из таблицы функции входов триггера со счетным входом следует, что $(\varphi = 1) \iff (\tau_{\text{исходное}} \neq \tau_{\text{перехода}})$. Таким образом, если на некотором переходе из одного состояния в другое триггер Π_i меняет состояние (с «0» на «1» или с «1» на «0»), то соответствующая дуга графа отмечается символом φ_i (см. рис. 2.11).

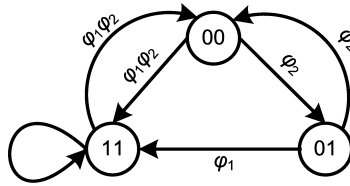


Рис. 2.11. Функции возбуждения памяти при синтезе на триггерах со счетным входом

2.7.3. Синтез на триггерах с отдельными входами

По таблице функции входов триггера с отдельными входами получаем:

$$0 \xrightarrow{\varphi=1} 1 \qquad 1 \xrightarrow{\psi=1} 0$$

Таким образом, каждой дуге графа приписывается набор символов из множества $\{\varphi_i, \psi_i\}_{i=1}^I$ по следующему правилу: если Π_i меняет состояние с «0» на «1», то дуга отмечается символом φ_i ; если Π_i меняет состояние с «1» на «0», то дуга отмечается символом ψ_i (см. рис. 2.12).

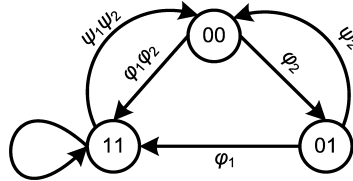


Рис. 2.12. Функции возбуждения памяти при синтезе на триггерах с отдельными входами

После построения соответствующего графа выписываем каноническую систему уравнений:

1. Для функций возбуждения памяти φ_i (ψ_i) – это дизъюнкция конъюнкций; каждая конъюнкция соответствует коду состояния, из которого выходит дуга, и входному сигналу, приписанному этой дуге. При этом рассматриваются только те дуги, которые отмечены символом φ_i (ψ_i). Например, при синтезе на триггерах с отдельными входами $\varphi_1 = \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2$.

2. Для функций y_i – аналогично. Например, $y_1 = \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 \vee \tau_1\tau_2x_1\bar{x}_2$.

3. Функции r_i – это дизъюнкции конъюнкций; каждая конъюнкция соответствует коду состояния, отмеченного символом r_i . В нашем примере $r = \bar{\tau}_1\bar{\tau}_2 \vee \tau_1\tau_2$.

Глава 3

Кодирование состояний автомата

3.1. Гонки в автомате

Пусть задан некоторый абстрактный автомат и $\{a_1, \dots, a_M\}$ – состояния автомата. Напомним, что каждое состояние автомата будет кодироваться вектором длины I , где $I \geq \lceil \log_{\Theta} M \rceil$, Θ – число состояний элемента памяти. Кроме того, число элементов памяти также равно I .

Если при переходе автомата из одного состояния в другое должны изменить свои состояния несколько элементов памяти, то между ними начинаются *состязания*. Выигрывает состязание тот элемент памяти, который изменил свое состояние первым. Через цепь обратной связи он может изменить входные сигналы других элементов памяти до того, как они изменят свои состояния. В итоге автомат может перейти в состояние, не предусмотренное законом его функционирования.

Причины состязаний следующие:

- элементы памяти имеют различные, хотя и достаточно близкие, времена срабатывания;
- различны задержки сигналов возбуждения, поступающие

на входные каналы элементов памяти по логическим цепям, имеющим не одинаковую длину.

Пример 3.1. Пусть автомат под действием сигнала z_f должен перейти из состояния a_m с кодом «0101» в состояние a_s с кодом «1001». Это означает, что элемент памяти

Π_1 переходит из состояния «0» в состояние «1»,

Π_2 — из «1» в «0»,

Π_3 не меняет состояние,

Π_4 не меняет состояние.

В следствие состязаний автомат может оказаться в некотором промежуточном состоянии a_k . Если $\delta(a_k, z_f) = a_s$, то такие состязания называются **допустимыми** (см. рис. 3.1). Если $\delta(a_k, z_f) \neq a_s$, то такие состязания называются **гонками** (см. рис. 3.2).

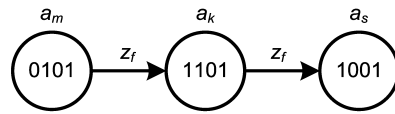


Рис. 3.1. Допустимые состязания между элементами памяти

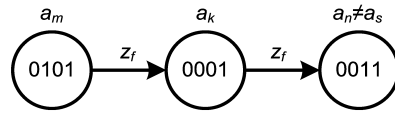


Рис. 3.2. Гонки между элементами памяти

Существуют следующие **способы ликвидации гонок**:

1. Аппаратные методы.

1.1. Тактирование входных сигналов автомата импульсами. Пусть x_1, \dots, x_L — входные каналы, p — входной канал от генератора синхроимпульса:

$$p = \begin{cases} 1, & \text{в момент прихода импульса;} \\ 0, & \text{иначе.} \end{cases}$$

В этом случае входной сигнал на переходе из состояния a_m в состояние a_s будет не z_f , а $(p \cdot z_f)$. Длительность импульса t_p должна быть меньше времени прохождения самого короткого пути сигнала обратной связи. Тогда к моменту перехода в промежуточное состояние a_k сигнал p будет равен «0», следовательно, $p \cdot z_f = 0$ и гонки будут исключены.

1.2. Двойная память – каждый элемент памяти дублируется. Перезапись из нижней памяти в верхнюю происходит в момент отсутствия тактирующего сигнала.

2. Специальные методы противогоночного кодирования состояний. При таком кодировании состояний гонки должны быть устранены.

2.1. Алгоритм развязывания пар переходов.

2.2. Соседнее кодирование.

3.2. Противогоночное кодирование состояний

3.2.1. Развязывание пар переходов

Пусть (α, β) и (γ, δ) – две пары двоичных кодов длины I , то есть $|\alpha| = |\beta| = |\gamma| = |\delta| = I$.

Две пары (α, β) и (γ, δ) называются **развязанными**, если $\exists i (1 \leq i \leq I)$ такое, что $\alpha_i = \beta_i$, $\gamma_i = \delta_i$, $\alpha_i \neq \gamma_i$. В противном случае пары называются **связанными**.

Теорема 3.1. В автомате, состояния которого закодированы двоичными кодами конечной длины, гонки отсутствуют тогда и только тогда, когда для любых двух переходов (a_m, a_s) и (a_k, a_l) (таких, что $a_s \neq a_l$), происходящих под действием одного и того же входного сигнала, соответствующие пары кодов состояний развязаны [1].

Замечание 3.1. Эта теорема доказана применительно к асинхронным автоматам. Если автомат синхронный, то условие $a_s \neq a_l$ может быть заменено более сильным: развязывать необходимо пары переходов, для которых $\{a_m, a_s\} \cap \{a_k, a_l\} = \emptyset$.

Пример 3.2. Пусть в автомате есть два перехода (a_m, a_s) и (a_k, a_l) , происходящие под действием одного и того же входного сигнала:

$$a_m = \boxed{1}001 \xrightarrow{z_f} \boxed{1}110 = a_s \quad (3.1)$$

$$a_k = \boxed{0}001 \xrightarrow{z_f} \boxed{0}110 = a_l \quad (3.2)$$

Эти два перехода (или две пары состояний) *развязаны*, так как $\exists i = 1$ такое, что $a_{m_1} = a_{s_1} = 1$, $a_{k_1} = a_{l_1} = 0$.

Теорема 3.1 лежит в основе алгоритма противогоночного кодирования, *идея* которого заключается в следующем: последовательно просматривать все пары переходов, для которых имеется хотя бы один общий входной сигнал, и выбирать коды так, чтобы каждая такая пара переходов была развязана.

На промежуточных этапах алгоритма состояниям автомата будут соответствовать коды, значения некоторых разрядов которых могут быть неопределены – *неполные* коды.

Пусть (a_m, a_s) , (a_k, a_l) – пара переходов, (α, β) и (γ, δ) – их коды, i – текущая длина кодов.

Алгоритм развязывания пары переходов:

1. Положить $i = 0$, перейти на шаг 2.
2. Если $i = 0$, то перейти на шаг 8, иначе перейти на шаг 3.
3. Если $\exists r$ ($1 \leq r \leq i$) такое, что значения r -тых разрядов четверки $\alpha, \beta, \gamma, \delta$ образуют набор (0011) или набор (1100), иными словами:

$$(\alpha_r, \beta_r, \gamma_r, \delta_r) = (0011) \vee (1100), \quad (3.3)$$

то перейти на шаг 9, иначе перейти на шаг 4.

4. Если $\exists r$ ($1 \leq r \leq i$) такое, что $(\alpha_r, \beta_r, \gamma_r, \delta_r)$ образует один из наборов:

$$\begin{array}{ccccc} (-011) & (0-11) & (00-1) & (001-) & (-01-) \\ (- -11) & (0--1) & (00--) & (-0-1) & (0-1-) \\ (- - -1) & (0---) & (-0--) & (--1-) & (----) \end{array}$$

то перейти на шаг 5, иначе перейти на шаг 6.

5. Доопределить неопределенные значения r -тых разрядов четверки $\alpha, \beta, \gamma, \delta$ так, чтобы $(\alpha_r, \beta_r, \gamma_r, \delta_r) = (0011)$, перейти на шаг 9.

6. Если $\exists r$ ($1 \leq r \leq i$) такое, что $(\alpha_r, \beta_r, \gamma_r, \delta_r)$ образует один из наборов:

$(- 1 0 0)$	$(1 - 0 0)$	$(1 1 - 0)$	$(1 1 0 -)$	$(- 1 0 -)$
$(- - 0 0)$	$(1 - - 0)$	$(1 1 - -)$	$(- 1 - 0)$	$(1 - 0 -)$
$(- - - 0)$	$(1 - - -)$	$(- 1 - -)$	$(- - 0 -)$	

то перейти на шаг 7, иначе перейти на шаг 8.

7. Доопределить неопределенные значения r -тых разрядов четверки $\alpha, \beta, \gamma, \delta$ так, чтобы $(\alpha_r, \beta_r, \gamma_r, \delta_r) = (1100)$, перейти на шаг 9.

8. Доопределить коды $\alpha, \beta, \gamma, \delta$ одним неопределенным разрядом, увеличить i на единицу, перейти на шаг 4.

9. Пара переходов $(a_m, a_s), (a_k, a_l)$ развязана.

Замечание 3.2. Длина кода в большинстве случаев оказывается неминимальной, так как с каждым новым разрядом могут развязываться пары переходов, которые уже были развязаны ранее. Следовательно, необходима *минимизация длины кода*. Для этого:

1. Исключаем один из разрядов кодов. При этом некоторые пары переходов могут оказаться связанными. Следовательно, применяем алгоритм развязывания пар переходов.

2. Исключаем еще один разряд и вновь применяем алгоритм. И так до тех пор, пока длина кода не перестанет уменьшаться или ее уже нельзя уменьшить ($I \geq \lceil \log_{\Theta} M \rceil$).

3. Если значения некоторых разрядов окажутся неопределены, то их можно доопределить произвольно.

Пример 3.3. Пусть дана таблица переходов некоторого частичного асинхронного автомата (см. табл. 3.1). Закодируем состояния автомата, используя алгоритм развязывания пар переходов.

Очевидно, что должны быть развязаны всевозможные пары переходов в каждом из трех множеств Z_1, Z_2, Z_3 , в которых переходы группируются в соответствии с входными сигналами (см. табл. 3.2).

Таблица 3.1. Таблица переходов частичного асинхронного автомата

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_2	a_2	a_4	a_4	a_6	a_6	—
z_2	a_1	a_3	a_3	a_1	a_3	—	—
z_3	—	a_5	a_7	—	a_5	—	a_7

Таблица 3.2. Множества переходов, происходящих под действием общего входного сигнала

Z_1	Z_2	Z_3
(a_1, a_2)	(a_1, a_1)	(a_2, a_5)
(a_2, a_2)	(a_2, a_3)	(a_3, a_7)
(a_3, a_4)	(a_3, a_3)	(a_5, a_5)
(a_4, a_4)	(a_4, a_1)	(a_7, a_7)
(a_5, a_6)	(a_5, a_3)	
(a_6, a_6)		

Развязывание пар переходов в первом столбце начнем с перехода (a_1, a_2) . Согласно теореме, пару (a_1, a_2) и (a_2, a_2) развязывать не надо, так как состояния перехода совпадают. Первая пара переходов, которую необходимо развязывать – это пара (a_1, a_2) и (a_3, a_4) . Для этих состояний (a_1, a_2, a_3, a_4) по первой переменной τ_1 образуем четверку (0011) (см. столбец «12|34» в табл. 3.3).

Паре переходов (a_1, a_2) и (a_4, a_4) соответствует четверка (0011) по переменной τ_1 (см. столбец «12|34» в табл. 3.3), то есть эта пара тоже развязана.

Пара переходов (a_1, a_2) и (a_5, a_6) образует четверку (00 - -) по переменной τ_1 . Для развязывания этой пары доопределим эту четверку до (0011) (см. столбец «12|56» в табл. 3.3).

Из этого же столбца видно, что пара (a_1, a_2) и (a_6, a_6) развязана (четверка (0011)). Точно также развязаны пары, образованные переходом (a_2, a_2) и всеми последующими переходами в множестве Z_1 .

Рассмотрим пару (a_3, a_4) , (a_5, a_6) . Из столбца «12|56» (см.

табл. 3.3) получаем четверку (1111), то есть пара не развязана. Заводим новую переменную τ_2 и определяем ее значение для состояний a_3 , a_4 равным нулю, для a_5 , a_6 – единице (см. столбец «34|56» в табл. 3.3).

После этого остальные пары переходов в множестве Z_1 также будут развязаны.

Действуя аналогично, развязываем пары переходов в множествах Z_2 и Z_3 и заполняем таблицу 3.3.

Таблица 3.3. Кодирование состояний алгоритмом развязывания пар переходов

	12 34	12 56	34 56	11 23	23 41	41 53	25 37
a_1	0	0	0 –	0 1	0 1 1	0 1 1	0 1 1 –
a_2	0	0	0 –	0 0	0 0 0	0 0 0	0 0 0 0
a_3	1	1	1 0	1 0	1 0 0	1 0 0	1 0 0 1
a_4	1	1	1 0	1 0	1 0 1	1 0 1	1 0 1 –
a_5	–	1	1 1	1 1	1 1 –	1 1 0	1 1 0 0
a_6	–	1	1 1	1 1	1 1 –	1 1 –	1 1 – –
a_7	–	–	– –	– –	– – –	– – –	– – – 1
	τ_1	τ_1	$\tau_1 \tau_2$	$\tau_1 \tau_2$	$\tau_1 \tau_2 \tau_3$	$\tau_1 \tau_2 \tau_3$	$\tau_1 \tau_2 \tau_3 \tau_4$

Теперь переходим к минимизации. Исключаем переменную τ_1 и повторяем процесс развязывания пар переходов (см. табл. 3.4, 3.5). Переменная τ_5 добавлена, чтобы развязать пару переходов (a_1, a_2) , (a_5, a_6) . Все остальные пары остались развязаны.

Далее исключаем переменную τ_2 и получаем таблицу кодов с тремя переменными, в которой, после проверки, оказываются развязанными все пары переходов (см. табл. 3.6). Дальнейшая минимизация невозможна, так как для кодирования семи состояний необходимо не менее трех переменных. Доопределяя прочерки, получаем таблицу противогоночного кодирования состояний автомата (см. табл. 3.7).

Замечание 3.3. Рассмотренный способ кодирования не исключает состязания, он ликвидирует только гонки.

Таблица 3.4. Исключение пе-
ременной τ_1

	$\tau_2\tau_3\tau_4$
a_1	1 1 –
a_2	0 0 0
a_3	0 0 1
a_4	0 1 –
a_5	1 0 0
a_6	1 – –
a_7	– – 1

Таблица 3.5. Развязывание
пар

	$\tau_2\tau_3\tau_4\tau_5$
a_1	1 1 – 0
a_2	0 0 0 0
a_3	0 0 1 –
a_4	0 1 1 –
a_5	1 0 0 1
a_6	1 – – 1
a_7	0 – 1 –

Таблица 3.6. Исключение пе-
ременной τ_2

	$\tau_3\tau_4\tau_5$
a_1	1 – 0
a_2	0 0 0
a_3	0 1 –
a_4	1 1 –
a_5	0 0 1
a_6	– – 1
a_7	– 1 –

Таблица 3.7. Противогоночное
кодирование

	$\tau_3\tau_4\tau_5$
a_1	1 0 0
a_2	0 0 0
a_3	0 1 0
a_4	1 1 0
a_5	0 0 1
a_6	1 0 1
a_7	0 1 1

3.2.2. Соседнее кодирование

При соседнем кодировании состояний автомата условие отсутствия гонок (и состязаний вообще) всегда выполнено.

Идея алгоритма заключается в следующем: любые два состояния, связанные дугой на графе автомата, кодируются наборами, отличающимися состояниями лишь одного элемента памяти (см. рис. 3.3) [1].

К сожалению, соседнее кодирование не всегда возможно (см. рис. 3.4).

Требования к графу автомата, допускающего соседнее кодирование:

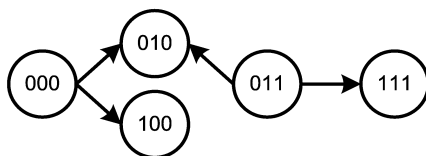


Рис. 3.3. Соседнее кодирование

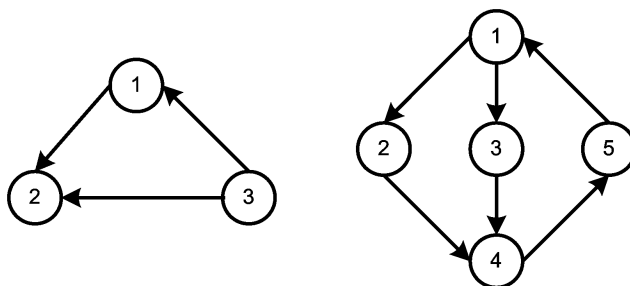


Рис. 3.4. Графы автоматов, которые не допускают соседнее кодирование

1. В графе автомата не существует циклов с нечетным числом вершин.
2. Два соседних состояния второго порядка (путь между ними состоит из двух ребер, независимо от направления) не должны иметь более двух состояний, лежащих между ними.

3.3. Кодирование состояний и сложность логической схемы

В этом параграфе попытаемся ответить на вопрос о том, как минимизировать число устройств, необходимых для реализации функций автомата.

3.3.1. Оптимальное кодирование при синтезе на задержках

Один из способов минимизации – это *оптимальное кодирование*. Рассматривая канонический метод структурного синтеза, получаем, что различные варианты кодирования состояний автомата приводят к различным выражениям функций выходов и функций возбуждения памяти. Отсюда непосредственно следует, что от способа кодирования зависит сложность логической схемы автомата.

Пример 3.4. Вернемся к примеру, разобранным в § 2.5. Напомним, что мы выбрали следующую кодировку состояний:

$$a_1 = 00 \quad a_2 = 01 \quad a_3 = 11$$

В результате, при синтезе на задержках, мы получили довольно сложную форму для функций возбуждения памяти (см. формулы (2.11)). Закодируем теперь состояния по иному:

$$a_1 = 01 \quad a_2 = 10 \quad a_3 = 00$$

В этом случае получим:

$$\begin{aligned} \varphi_1 &= \bar{\tau}_1 \tau_2 \bar{x}_1 \bar{x}_2 \vee \bar{\tau}_1 \tau_2 x_1 \bar{x}_2, \\ \varphi_2 &= \tau_1 \bar{\tau}_2 \bar{x}_1 x_2 \vee \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 \bar{x}_2. \end{aligned} \quad (3.4)$$

Функции возбуждения памяти существенно упростились. Почему так получилось? Дело в том, что мы выбрали в качестве «00» состояние, наиболее часто встречающееся в таблице переходов – как-бы «основное» состояние.

В общем случае используется следующий **алгоритм оптимального кодирования при синтезе автомата на задержках**:

1. Каждому состоянию автомата a_m ($1 \leq m \leq M$) сопоставим целое число N_m – число переходов в состояние a_m , или, что то же самое, число появлений a_m в таблице переходов, или число входящих дуг на графе автомата.

2. Упорядочим числа N_1, \dots, N_M по невозрастанию.

3. Состояние a_t с наибольшим N_t закодируем одними нулями: $(00 \dots 0)$.

4. Следующие I состояний (I равно числу элементов памяти или длине кода) в упорядоченном по пункту 2 списке закодируем наборами, содержащими одну «1»: $(00 \dots 01), (00 \dots 10), \dots, (10 \dots 00)$.

5. Затем используем коды, содержащие две единицы и т.д.

В результате получаем такое кодирование, при котором чем больше существует переходов в некоторое состояние, тем меньше единиц содержится в его коде. А так как при синтезе на задержках функции возбуждения памяти строятся непосредственно по таблице переходов, мы получаем их оптимальные, с точки зрения количества логических элементов, выражения.

Аналогичные соображения могут быть использованы и при кодировании выходных сигналов для минимизации функций выходов.

Пример 3.5. Вновь вернемся к автомату из § 2.5. В отмеченной таблице выходов (см. табл. (2.13)) выходной сигнал w_3 встречается 3 раза, w_2 — 2 раза, w_1 — 1 раз, w_4 — 1 раз, u_1 — 2 раза, u_2 — 1 раз. Закодируем выходные сигналы в соответствии с упорядочиванием (см. табл. 3.8, 3.9).

Таблица 3.8. Кодирование выходных сигналов 1-го типа

	y_1	y_2
w_3	0	0
w_2	0	1
w_1	1	0
w_4	1	1

Таблица 3.9. Кодирование выходных сигналов 2-го типа

	r
u_1	0
u_2	1

После такой кодировки мы получим отмеченную таблицу выходов, в которой будет *мало* единиц (см. табл. 3.10).

Таблица 3.10. Отеченная таблица выходов

		0		1		0	
		0	0	0	1	1	1
0	0	0	0	—	—	0	1
0	1	1	1	0	0	—	—
1	0	0	1	1	0	0	0

В итоге выражения для функций выходов примут следующий вид:

$$\begin{aligned}
 y_1 &= \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 x_2 \vee \bar{\tau}_1 \tau_2 x_1 \bar{x}_2; \\
 y_2 &= \bar{\tau}_1 \bar{\tau}_2 \bar{x}_1 x_2 \vee \bar{\tau}_1 \bar{\tau}_2 x_1 \bar{x}_2 \vee \tau_1 \tau_2 \bar{x}_1 \bar{x}_2; \\
 r &= \bar{\tau}_1 \tau_2.
 \end{aligned} \tag{3.5}$$

Несомненно, есть выигрыш, по сравнению с формулами (2.8), (2.9).

Замечание 3.4. Рассмотренный алгоритм оптимального кодирования состояний автомата для минимизации функций возбуждения памяти применим только при синтезе на задержках, так как в общем случае таблица функций возбуждения не совпадает с таблицей переходов автомата.

3.3.2. Эвристический алгоритм кодирования, уменьшающий сложность логической схемы

Если в качестве элементов памяти выбраны произвольные элементы, используется *эвристический алгоритм кодирования* состояний.

Идея алгоритма заключается в поиске такого кодирования, при котором минимизируется суммарное количество изменений состояний элементов памяти на всех переходах автомата.

Введем **весовую функцию** $W = \sum W_{ms}$, где $W_{ms} = |K_m - K_s|$ — расстояние между кодами состояний a_m и a_s , равное числу элементов памяти, изменяющих свое состояние на

переходе (a_m, a_s) . Суммирование производится по всем переходам автомата. Введенная функция может служить одной из оценок сложности логической схемы.

Далее разберем работу алгоритма по шагам. Пусть задан автомат S .

1. Построить матрицу M , которая состоит из всех различных пар (α_r, β_r) таких, что в автомате S есть переход $(a_{\alpha_r}, a_{\beta_r})$. При этом петли не рассматриваются.

$$\begin{vmatrix} \alpha_1 & \beta_1 \\ & \dots \\ \alpha_r & \beta_r \\ & \dots \\ \alpha_R & \beta_R \end{vmatrix}$$

2. Переставить в матрице строки так, чтобы выполнялось условие:

$$\{\alpha_r, \beta_r\} \cap \{\alpha_1, \beta_1, \dots, \alpha_{r-1}, \beta_{r-1}\} \neq \emptyset \quad |_{r=2}^R \quad (3.6)$$

Это условие означает, что хотя бы один элемент из строки r содержится в какой-нибудь из предыдущих строк. Предполагается, что автомат не имеет изолированных подавтоматов – тогда такая перестановка всегда существует.

3. Закодировать состояния из первой строки матрицы M :

$$K_{\alpha_1} = (00 \dots 00) \quad K_{\beta_1} = (00 \dots 01)$$

4. Вычеркнуть из матрицы M первую строку и все последующие, в которых оба элемента закодированы. Полученную матрицу обозначить M' .

5. В силу условия (3.6), в первой строке матрицы M' закодирован один элемент. Пусть второй незакодированный элемент – это γ .

6. Построить матрицу M_γ путем выбора из матрицы M' строк, содержащих элемент γ . Пусть $B_\gamma = \{\gamma_1, \dots, \gamma_F\}$ – множество элементов из матрицы M_γ , которые уже закодированы (их коды $K_{\gamma_1}, \dots, K_{\gamma_F}$, соответственно).

7. Для каждого K_{γ_f} ($1 \leq f \leq F$) найти $C_{\gamma_f}^1$ – множество кодов, соседних с K_{γ_f} и еще не занятых для кодирования состояний автомата. Построить множество $D_\gamma^1 = \cup_{f=1}^F C_{\gamma_f}^1$. Если

$D_\gamma^1 = \emptyset$, то построить $D_\gamma^2 = \cup_{f=1}^F C_{\gamma_f}^2$, где $C_{\gamma_f}^2$ – множество кодов, у которых кодовое расстояние с K_{γ_f} равно 2. И так до тех пор, пока не найдется $D_\gamma^n \neq \emptyset$. Пусть $D_\gamma^n = \{K_{\delta_1}, \dots, K_{\delta_G}\}$.

8. Для каждого K_{δ_g} ($1 \leq g \leq G$) найти $W_{\delta_g \gamma_f} = |K_{\delta_g} - K_{\gamma_f}|$ – кодовые расстояния между K_{δ_g} и всеми использованными кодами K_{γ_f} ($1 \leq f \leq F$) из B_γ .

9. Найти $W_{\delta_g} = \sum_{f=1}^F W_{\delta_g \gamma_f}$ ($1 \leq g \leq G$). При этом, если в автомате имеются переходы (a_γ, a_{γ_f}) и (a_{γ_f}, a_γ) , то в сумме слагаемое $W_{\delta_g \gamma_f}$ повторяется дважды.

10. Из D_γ^n выбрать код K_γ , у которого $W_\gamma = \min_g W_{\delta_g}$. Элементу γ (состоянию a_γ) поставить в соответствие код K_γ .

11. Из матрицы M' вычеркнуть строки, в которых оба элемента закодированы – получить новую матрицу M' . Если она пуста, то перейти на шаг 12, иначе – на шаг 5.

12. Вычислить W и оценить **качество кодирования** $k = W/p$, где p – число переходов в автомате (без учета петель). Несложно показать справедливость следующих утверждений.

Утверждение 3.1. При соседнем кодировании $k = 1$.

Утверждение 3.2. Если I – длина кода, то $1 \leq k \leq I$ (так как $1 \leq W_{ms} \leq I$ и в сумме для расчета W всего p слагаемых).

Пример 3.6. Пусть дан автомат A , изображенный на рисунке 3.5.

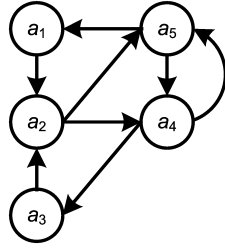


Рис. 3.5. Граф автомата A

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

Очевидно, что условие (3.6) выполнено. Длина кода $I = 3$ ($I \geq \lceil \log_2 5 \rceil$). Тогда $K_1 = (000)$, $K_2 = (001)$. Кодирование будем иллюстрировать картой Карно (по горизонтали — τ_1 , по вертикали — $\tau_2\tau_3$, в клетках записаны номера состояний) (см. табл. 3.11).

Таблица 3.11. Карта Карно (1)

	00	01	11	10
0	1	2		
1				

$$M' = \begin{vmatrix} 2 & 4 \\ 2 & 5 \\ 3 & 2 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix}$$

Состояние a_2 закодировано, таким образом $\gamma = 4$.

$$M_4 = \begin{vmatrix} 2 & 4 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \end{vmatrix}$$

$B_4 = \{2\}$ — номера состояний из M_4 , которые уже закодированы

$$C_2^1 = \{(101), (011)\}; \quad D_4^1 = C_2^1 = \{(101), (011)\} \neq \emptyset;$$

$$W_{101} = |(101) - (001)| = 1; \quad W_{011} = |(011) - (001)| = 1.$$

Таблица 3.12. Карта Карно (2)

	00	01	11	10
0	1	2		
1		4		

Выбираем $K_4 = (101)$ (см. табл. 3.12).

Вновь строим матрицу M' :

$$M' = \begin{vmatrix} 2 & 5 \\ 3 & 2 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix} \quad \gamma = 5.$$

$$M_5 = \begin{vmatrix} 2 & 5 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix} \quad B_5 = \{2, 4, 1\}$$

$$C_2^1 = \{(011)\}; \quad C_4^1 = \{(100), (111)\}; \quad C_1^1 = \{(100), (010)\};$$

$$D_5^1 = C_2^1 \cup C_4^1 \cup C_1^1 = \{(011), (100), (111), (010)\} \neq \emptyset;$$

$$W_{011} = |(011) - (001)| + |(011) - (101)| + |(011) - (101)| + \\ + |(011) - (000)| = 1 + 2 + 2 + 2 = 7;$$

$$W_{100} = |(100) - (001)| + |(100) - (101)| + |(100) - (101)| + \\ + |(100) - (000)| = 2 + 1 + 1 + 1 = 5;$$

$$W_{111} = |(111) - (001)| + |(111) - (101)| + |(111) - (101)| + \\ + |(111) - (000)| = 2 + 1 + 1 + 3 = 7;$$

$$W_{010} = |(010) - (001)| + |(010) - (101)| + |(010) - (101)| + \\ + |(010) - (000)| = 2 + 3 + 3 + 1 = 9.$$

Выбираем $K_5 = (100)$ (см. табл. 3.13).

Таблица 3.13. Карта Карно (3)

	00	01	11	10
0	1	2		
1	5	4		

Строим новую матрицу M' :

$$M' = \begin{vmatrix} 3 & 2 \\ 4 & 3 \end{vmatrix} \quad \gamma = 3.$$

$$M_3 = \begin{vmatrix} 3 & 2 \\ 4 & 3 \end{vmatrix} \quad B_3 = \{2, 4\}$$

$$C_2^1 = \{(011)\}; \quad C_4^1 = \{(111)\};$$

$$D_3^1 = C_2^1 \cup C_4^1 = \{(011), (111)\} \neq \emptyset;$$

$$W_{011} = |(011) - (001)| + |(011) - (101)| = 1 + 2 = 3;$$

$$W_{111} = |(111) - (001)| + |(111) - (101)| = 2 + 1 = 3.$$

Выбираем $K_3 = (011)$ (см. табл. 3.14).

Таблица 3.14. Карта Карно (4)

	00	01	11	10
0	1	2	3	
1	5	4		

В заключении оценим качество кодирования:

$$k = W/p = (1 + 1 + 2 + 1 + 2 + 1 + 1 + 1)/8 = 10/8 = 1,25.$$

Глава 4

Синтез микропрограммных автоматов

4.1. Структура дискретного устройства

Пусть дискретное устройство выполняет *команды (операции)* из набора $K = \{k_1, k_2, \dots, k_S\}$ над операндами из множества D с целью получения результатов, относящихся к множеству R . При этом в каждый момент времени устройство выполняет только одну команду, которая, как правило, занимает несколько тактов машинного времени.

Следуя В.М. Глушкову, дискретное устройство будем представлять в виде двух частей или подсистем:

- 1) *операционное устройство (ОУ)* или операционный автомат;
- 2) *устройство управления (УУ)* или управляющий автомат.

Заметим, что на практике часто используется многоступен-

чатое представление дискретной системы. Так, при проектировании процессора, арифметическое устройство (АУ) рассматривается как часть операционного устройства. Но внутри АУ есть свой автомат управления, а к операционной части АУ относят его регистры, сумматоры и т. д.

Рассмотрим структуру дискретного устройства более подробно (см. рис. 4.1).

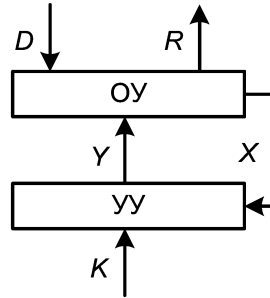


Рис. 4.1. Структура дискретного устройства

ОУ принимает из внешней среды, хранит и преобразует внешние данные из множества D (входные слова); выдает во внешнюю среду результаты преобразований, принадлежащие множеству R (выходные слова); передает в УУ *осведомительные сигналы* $x_i \in X = \{x_1, \dots, x_L\}$. Осведомительные сигналы представляют собой значения *логических условий*. Сами логические условия также будем обозначать символами множества X .

Определим *микрооперацию* как элементарный процесс переработки информации в ОУ, происходящий за время одного такта. Для выполнения конкретной микрооперации УУ должно выработать *управляющий сигнал (сигнал микрооперации)* $y_i \in \{y_1, \dots, y_N\} = Y$. Микрооперацию будем обозначать тем же символом множества Y , что и соответствующий сигнал. Если же в устройстве одновременно реализуется несколько микроопераций y_{t_1}, \dots, y_{t_U} , то этот

набор назовем **микрокомандой**, которую будем обозначать $Y_t = \{y_{t_1}, \dots, y_{t_U}\}$. При $U = 1$ микрокоманда состоит из одной микрооперации. Возможен и случай нуля – это означает, что за данный такт времени УУ не выдает никаких микрокоманд (или выдает микрокоманду «пауза»).

УУ задает порядок выполнения микроопераций (микрокоманд) в ОУ при помощи управляющих сигналов $y_i \in Y = \{y_1, \dots, y_N\}$. Последовательность управляющих сигналов (а значит и порядок следования микроопераций) определяется командой $k \in K$, поступающей в УУ извне, и осведомительными сигналами из множества X .

Таким образом, *ОУ служит для хранения информации, выполнения микроопераций и вычисления логических условий. Задача УУ – выработка распределенной во времени последовательности управляющих сигналов, под воздействием которых в ОУ осуществляются микрооперации.*

Описание команды $k \in K$ в терминах микроопераций и логических условий будем называть **микропрограммой**. Тогда можно сказать, что *УУ обеспечивает требуемый порядок следования микроопераций на основе заданных микропрограмм.*

Заметим, что с позиции теории автоматов и ОУ, и УУ можно рассматривать как некоторые последовательные автоматы. Основное их различие – в количестве внутренних состояний ($10^2 - 10^3$ в УУ и $10^{10} - 10^{100}$ в ОУ). Количественная разница влечет качественно различный подход к решению задачи синтеза данных автоматов.

Задачу синтеза операционных устройств оставим за рамками нашего рассмотрения. Заинтересованный читатель может обратиться, например, к [5]. Мы остановимся на синтезе устройств управления. Эта задача представляет наибольшую сложность при проектировании дискретной системы, в частности из-за того, что устройство управления не повторяется при переходе от одной системы к другой.

Итак, УУ будем отождествлять с последовательным автоматом, который реализует микропрограмму работы дис-

кредного устройства¹. Такой автомат называется **микропрограммным**. Для записи самих микропрограмм удобно использовать язык граф-схем алгоритмов (ГСА). Исходя из вышесказанного, **синтез УУ** разбивается на три этапа:

1. Получение ГСА микропрограммы.
2. Построение графа (или списка переходов) микропрограммного автомата.
3. Построение логической схемы (структурный синтез) микропрограммного автомата.

В заключении параграфа остановимся на требованиях, которые налагаются на набор микроопераций и логических условий $W = \{y_1, \dots, y_N, x_1, \dots, x_L\}$. Как и в случае элементного базиса, это полнота и эффективность.

1. Набор микроопераций и логических условий W называется **полным**, если из его элементов можно составить микропрограмму для любой операции k из системы команд дискретного устройства.

2. **Эффективность** понимается в том смысле, что набор W минимизирует время выполнения операций при соблюдении ограничений на оборудование и т. п.

Как эти задачи решены?

1. Существуют необходимые и достаточные условия полноты набора W [2].

2. Для оценки эффективности введем следующие обозначения. Пусть k_1, \dots, k_S – заданный набор команд; p_1, \dots, p_S – вероятности использования команд; $C(W)$ – цена оборудования, реализующего набор W . Очевидно, что для любой команды k_i можно построить микропрограмму M_i , оптимальную в смысле минимума среднего времени ее выполнения $t_i(W)$. Следовательно, среднее время выполнения команды:

$$T(W) = \sum_{i=1}^S p_i t_i(W), \quad (4.1)$$

¹Строго говоря, УУ реализует набор микропрограмм. Но очевидно, что достаточно рассмотреть алгоритм синтеза для случая одной микропрограммы.

быстродействие:

$$V(W) = 1/T(W), \quad (4.2)$$

цена единицы быстродействия:

$$q(W) = C(W)/V(W) = C(W)T(W). \quad (4.3)$$

Набор W *эффективен*, если он минимизирует $q(W)$. Поиск такого набора – задача весьма *трудоемкая*. Как правило, на практике набор микроопераций и логических условий выбирается *эвристически*.

4.2. ГСА микропрограммы

Предположим, что в операторных вершинах ГСА записаны микрокоманды или *операторы* Y_1, \dots, Y_T , различающиеся хотя бы номерами. Тогда вершины можно отождествить с записанными в них микрокомандами-операторами. Начальной вершине поставим в соответствие оператор Y_0 , а конечной – Y_{T+1} . В этом случае путь из вершины i в вершину j можно записать в виде:

$$Y_i p_{i_1} \dots p_{i_r} \dots p_{i_R} Y_j, \quad (4.4)$$

проходящий только через условные вершины $p_{i_1} \dots p_{i_R}$. Каждому такому пути соответствует конъюнкция:

$$\alpha_{ij} = x_{i_1}^{e_{i_1}} \dots x_{i_r}^{e_{i_r}} \dots x_{i_R}^{e_{i_R}}, \quad (4.5)$$

где $x_{i_r} \in X$ – логическое условие, записанное в условной вершине p_{i_r} ; $e_{i_r} \in \{0, 1\}$ – символ, приписанный выходу условной вершины p_{i_r} , через который проходит рассматриваемый путь; $x_{i_r}^1 = x_{i_r}$ и $x_{i_r}^0 = \bar{x}_{i_r}$. Если между вершинами Y_i, Y_j есть несколько путей, проходящих через разные условные вершины, то α_{ij} равно дизъюнкции конъюнкций, соответствующих всем путям от вершины i к вершине j :

$$\alpha_{ij} = \bigvee_{h=1}^H \alpha_{ij}^h, \quad (4.6)$$

где α_{ij}^h – конъюнкция, соответствующая h -тому пути из Y_i в Y_j .

Функция α_{ij} называется **функцией перехода** от вершины i к вершине j или **от оператора Y_i к оператору Y_j** . Очевидно, что $\alpha_{ij}\alpha_{ik} = 0$ (так как $x_{i_r}\bar{x}_{i_r} = 0$), это условие носит название **условие ортогональности**. Также несложно показать, что выполняется **условие полноты**: $\bigvee_{j=1}^{T+1} \alpha_{ij} = 1$ (так как $x_{i_r} \vee \bar{x}_{i_r} = 1$). Полное ортогональное множество функций называется **нормальным** множеством.

Всевозможные наборы значений двоичных переменных x_1, \dots, x_L обозначим через $\Delta_1, \dots, \Delta_\Theta$, где $\Theta = 2^L$. Определим **процесс выполнения ГСА**, начиная с некоторого оператора Y_0 , на произвольной, возможно бесконечной, последовательности наборов $\Delta_{m_1}, \dots, \Delta_{m_q}, \dots$:

Шаг 0: выписываем оператор Y_0 .

Шаг 1: переменным x_1, \dots, x_L задаем значения из набора Δ_{m_1} . В множестве функций перехода $\alpha_{01}, \dots, \alpha_{0T+1}$ выбираем функцию, принимающую значение «1» на этом наборе: $\alpha_{0i_1}(\Delta_{m_1}) = 1$. В строку рядом с Y_0 записываем Y_{i_1} :

$$Y_0 Y_{i_1}.$$

Шаг 2: На следующем шаге придаем переменным x_1, \dots, x_L значения из набора Δ_{m_2} . Аналогично предыдущему шагу, из множества функций перехода $\alpha_{i_1 1}, \dots, \alpha_{i_1 T+1}$ выбираем функцию $\alpha_{i_1 i_2}(\Delta_{m_2}) = 1$ и записываем:

$$Y_0 Y_{i_1} Y_{i_2}.$$

При переходе к Y_{T+1} алгоритм заканчивается.

Заметим, что во время выполнения оператора Y_t ($t = 1, \dots, T+1$) могут изменяться зависящие от него логические условия, которые образуют подмножество B_t . В этом случае говорят, что задано **распределение сдвигов**, являющееся дополнительной информацией о логике работы устройства.

Последовательность наборов значений логических условий $\Delta_{m_1}, \dots, \Delta_{m_q}, \dots$ называется **допустимой для ГСА** при заданном распределении сдвигов, если $\Delta_{m_{q+1}}$ совпадает с Δ_{m_q} .

или отличается от него значениями переменных из множества сдвигов B_{i_q} , то есть «внешних» изменений переменные x_1, \dots, x_L не испытывают.

Пример 4.1. Рассмотрим граф-схему алгоритма, представленную на рисунке 4.2.

Схема имеет 16 условных и 11 операторных вершин. $X = \{x_1, \dots, x_8\}$, $Y = \{y_1, \dots, y_9\}$, то есть соответствующий последовательный автомат должен иметь 8 входных каналов и 9 выходных. Это означает, что для каждого логического условия имеется свой входной канал, а для каждой микрооперации – свой выходной канал.

В общем случае, если число микроопераций равно N , то каждому оператору $Y_t = \{y_{t_1}, \dots, y_{t_U}\}$ отвечает вектор выходов (y_1, \dots, y_N) с ненулевыми компонентами, входящими в Y_t . В частности, входному оператору Y_0 отвечает нулевой вектор. Будем говорить, что при выполнении Y_0 выходные сигналы не выдаются, а при выполнении других операторов Y_t выдаются сигналы y_{t_1}, \dots, y_{t_U} .

Функция $\alpha_{01} = x_1$ принимает значение равное единице при $x_1 = 1$ – это конъюнкция содержимого всех условных вершин на пути от Y_0 до Y_1 . Далее, $\alpha_{02} = \bar{x}_1 x_6 x_2$, $\alpha_{04} = \bar{x}_1 x_6 \bar{x}_2 \vee \bar{x}_1 \bar{x}_6 x_4$, $\alpha_{09} = \bar{x}_1 \bar{x}_6 \bar{x}_4 x_7$, $\alpha_{08} = \bar{x}_1 \bar{x}_6 \bar{x}_4 \bar{x}_7$, а $\alpha_{03} = \alpha_{05} = \alpha_{06} = \alpha_{07} = 0$, то есть соответствующих путей нет, и т.д.

Обратим внимание на два случая:

1. За Y_5 сразу следует Y_{10} , тогда, очевидно, $\alpha_{5,10} = 1$, но это равенство получается и формально, как конъюнкция путского множества логических переменных.

2. Условная вершина, один из выходов которой соединен с ее входом (как в случае вершины x_3 , идущей после оператора Y_{10}), называется **возвратной**. В чем смысл возвратной вершины? Вершина как-бы «ждет» прихода $x_3 = 1$ и называется еще *ждущей* вершиной.

Путь от Y_{10} к Y_{11} – это объединение двух путей: $\alpha_{10,11} = \alpha_{10,11}^1 \vee \alpha_{10,11}^2 = x_3 x_4 \vee \bar{x}_3 x_3 x_4 = x_3 x_4$. Обратим внимание на то, что оператор Y_{11} сработает только тогда, когда $x_3 = 1$.

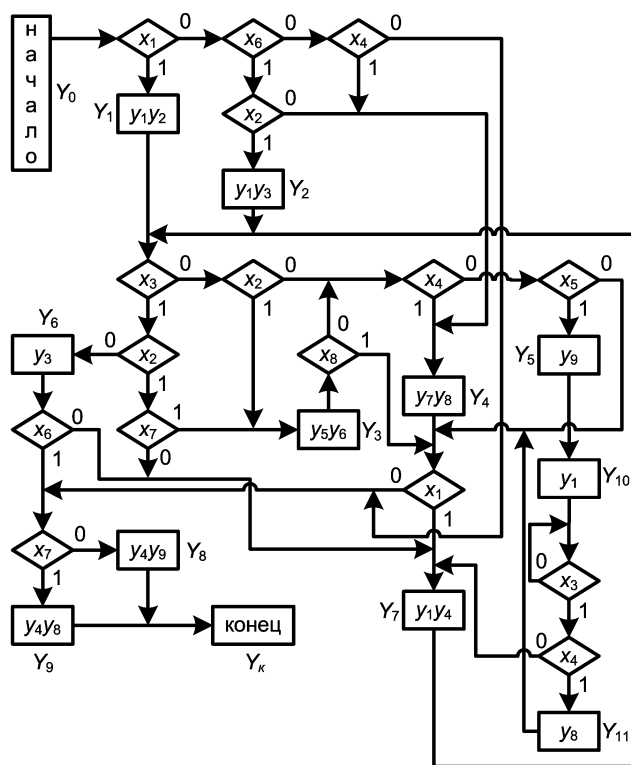


Рис. 4.2. Пример ГСА

4.3. Синтез микропрограммного автомата Мили по ГСА

Пусть дана ГСА микропрограммы. Как построить граф микропрограммного автомата Мили, отвечающий данной граф-схеме? Решение этой задачи разобьем на две части:

1. Получение отмеченной ГСА.
2. Построение графа микропрограммного автомата Мили.

Построение отмеченной ГСА при синтезе микропрограммного автомата Мили заключается в том, что входы вершин, следующих за операторными, отмечаются символами a_1, a_2, \dots по правилам:

- 1) символом a_1 отмечается вход первой вершины, следующей за начальной, а также вход конечной вершины;
- 2) входы вершин, следующих за операторными вершинами, отмечаются символами a_2, a_3, \dots ;
- 3) входы двух различных вершин, за исключением конечной, не могут быть отмечены одинаковыми символами;
- 4) вход вершины может отмечаться только одним символом.

Пример 4.2. ГСА, отмеченная по этим правилам, представлена на рисунке 4.3.

Очевидно, что мощность множества меток $\{a_1, a_2, \dots\} < \infty$. Пусть имеется M меток: $\{a_1, \dots, a_M\}$, L логических условий $\{x_1, \dots, x_L\}$ и N микроопераций $\{y_1, \dots, y_N\}$.

Построение графа микропрограммного автомата Мили проводится по отмеченной ГСА. Будем двигаться от одной метки a_m к другой метке a_s в направлении ориентации дуг ГСА. Если между двумя метками расположена одна операторная вершина $Y(a_m, a_s)$, то такие метки являются **соседними** (возможен и случай отсутствия операторной вершины, но тогда метка a_s должна совпадать с a_1). Путь между соседними метками на ГСА называется **путем перехода**. Каждому пути перехода можно поставить в соответствие конъюнкцию

$$X(a_m, a_s) = x_{m_1}^{e_{m_1}} \wedge \dots \wedge x_{m_R}^{e_{m_R}}, \quad (4.7)$$

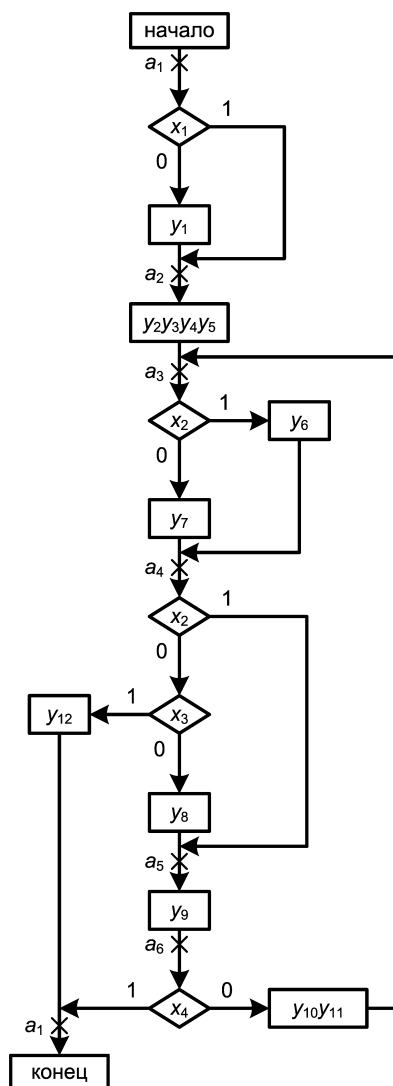


Рис. 4.3. Отмеченная ГСА при синтезе микропрограммного автомата Мили

которая берется по всем логическим условиям, входящим в рассматриваемый переход (см. рис. 4.4). Напомним, что $e_{m_r} \in \{0, 1\}$ – это символ, приписанный тому выходу условной вершины, который принадлежит рассматриваемому переходу, $x_{m_r}^1 = x_i$ и $x_{m_r}^0 = \bar{x}_i$.

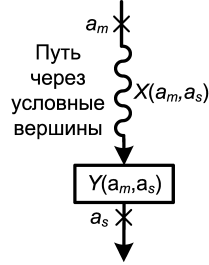


Рис. 4.4. Схематичное изображение пути перехода

Получаем, что каждый путь перехода можно представить в виде:

$$a_m X(a_m, a_s) Y(a_m, a_s) a_s, \quad (4.8)$$

где $a_m, a_s \in \{a_1, \dots, a_M\}$ или

$$a_m X(a_m, a_1) a_1, \quad (4.9)$$

где $a_m \in \{a_2, \dots, a_M\}$. Кроме того, в пути вида (4.8) допустим случай $R = 0$ (отсутствие логического условия):

$$a_m Y(a_m, a_s) a_s. \quad (4.10)$$

Если между a_m и a_s существует несколько путей перехода, проходящих через одну и ту же операторную вершину, будем считать, что $X(a_m, a_s) = \bigvee_{h=1}^H X_h(a_m, a_s)$.

Алгоритм построения графа микропрограммного автомата Мили S состоит из следующих шагов:

1. В качестве состояний автомата выбрать метки $\{a_1, \dots, a_M\}$, a_1 – начальное состояние.
2. Найти все пути перехода на отмеченной ГСА.

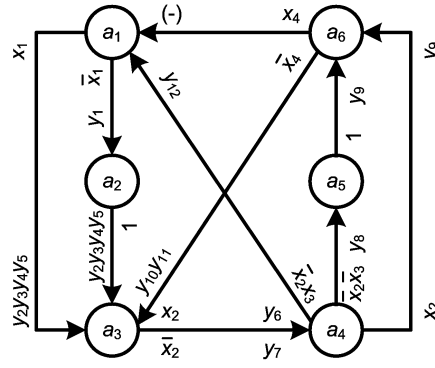


Рис. 4.5. Граф микропрограммного автомата Мили

2.1. Если некоторый путь содержит несколько символов x_r^{er} , то взять только один символ.

2.2. Если некоторый путь содержит как x_r , так и \bar{x}_r , то этот путь не рассматривать.

3. Каждому пути перехода $a_m X(a_m, a_s) Y(a_m, a_s) a_s$ поставить в соответствие переход автомата S из состояния a_m в состояние a_s под действием входного сигнала $X(a_m, a_s)$ с выдачей выходного сигнала $Y(a_m, a_s)$.

4. Каждому пути перехода $a_m Y(a_m, a_s) a_s$ поставить в соответствие переход автомата S из состояния a_m в состояние a_s под действием входного сигнала «1» (конъюнкция пустого множества переменных) с выдачей выходного сигнала $Y(a_m, a_s)$.

5. Каждому пути перехода $a_m X(a_m, a_1) a_1$ поставить в соответствие переход автомата S из состояния a_m в состояние a_1 под действием входного сигнала $X(a_m, a_1)$ с выдачей выходного сигнала Y_0 или «-» (пустой оператор).

Пример 4.3. Граф микропрограммного автомата Мили, построенный по отмеченной ГСА (рис. 4.3), представлен на рисунке 4.5.

Замечание 4.1. Состояния микропрограммного автомата заданы в абстрактном алфавите, тогда как входные и выходные сигналы определены в структурном алфавите $\{0, 1\}$. При этом входные каналы автомата – это множество логических условий $\{x_1, \dots, x_L\}$, выходные каналы – это множество микроопераций $\{y_1, \dots, y_N\}$.

Замечание 4.2. Как правило, работа микропрограммного автомата тактируется сигналами от генератора синхросигнала – в этом случае решена проблема гонок.

Замечание 4.3. Для запуска микропрограммного автомата используется еще один специальный сигнал B , который относится к группе входных сигналов и имеет длительность, равную такту. Этот *запускающий* сигнал необходим, чтобы исключить возможность появления выходных сигналов, в моменты, когда автомат находится в состоянии a_1 и не работает. При $B = 0$ автомат сохраняет начальное состояние a_1 и выходные сигналы отсутствуют. При $B = 1$ генерируется выходной сигнал и автомат переходит в новое состояние.

Введение сигнала B в закон функционирования автомата соответствует включению ждущей условной вершины B в ГСА микропрограммы (см. рис. 4.6). На графе автомата дугам, ис-

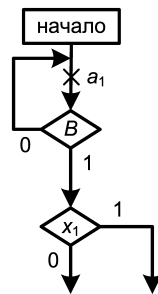
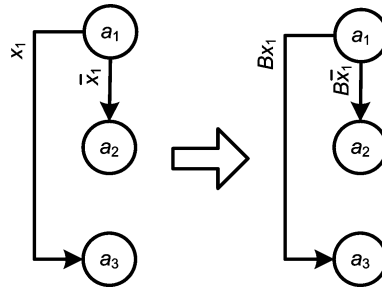


Рис. 4.6. Добавление запускающего сигнала B в ГСА микропрограммы

ходящим из a_1 , дополнительно приписывается запускающий сигнал B (см. рис. 4.7).

Рис. 4.7. Добавление запускающего сигнала B на графе автомата

Замечание 4.4. Необходимо учитывать, что автомат Мили интерпретирует микропрограмму корректно только в том случае, когда для всех путей перехода выполняется условие независимости логических условий x_α от результатов выполнения микроопераций y_β .

Условие независимости нарушается, если существует функциональная зависимость $x_\alpha = f(y_\beta)$. В этом случае выполнение микрооперации y_β может привести к изменению значения условия x_α . В свою очередь, автомат, реагируя на новый набор входных сигналов, может выдать ответ (набор выходных сигналов), не предусмотренный микропрограммой. Поэтому необходимо тщательно рассмотреть все пути перехода и выявить входные сигналы, зависящие от микроопераций. Далее, структурными средствами исключить влияние этих сигналов на процесс выполнения микроопераций. Для этих целей используются следующие способы:

1. Запоминание значений входных сигналов на один такт. Запоминание проводится с помощью триггеров, которые переключаются на критические значения входных сигналов в конце такта, одновременно с переходом автомата в новое состояние.

2. Введение в автомат дополнительных состояний. Пусть $x_\alpha = f(y_a, y_b)$. Как видно из рисунка 4.8, дополнительные состояния исключают возможность переключения сигналов y_a и

y_b при изменении значения x_α .

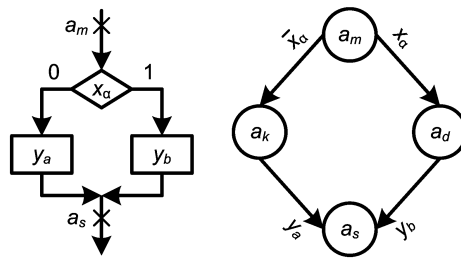


Рис. 4.8. Способ введения дополнительных состояний

3. Наконец, самое радикальное средство – синтез микропрограммного автомата по схеме Мура. В этом случае выходные сигналы зависят только от состояния автомата.

4.4. Синтез микропрограммного автомата Мура по ГСА

Пусть, как и прежде, дана ГСА микропрограммы. Построение микропрограммного автомата Мура, отвечающего данной граф-схеме, как и в случае модели Мили состоит из двух этапов:

1. Получение отмеченной ГСА.
2. Построение графа микропрограммного автомата Мура.

Построение отмеченной ГСА при синтезе микропрограммного автомата Мура состоит в том, что начальная, конечная и операторные вершины отмечаются символами a_1, a_2, \dots по правилам:

- 1) символом a_1 отмечается начальная и конечная вершины автомата;
- 2) каждая операторная вершина отмечается единственным символом a_i , то есть различные операторные вершины отмечаются различными символами;
- 3) все операторные вершины должны быть отмечены.

Заметим, что при синтезе автомата Мура, в отличие от автомата Мили, отмечаются не входы вершин, следующие за операторными, а сами операторные вершины. Число таких меток на единицу больше числа операторных вершин в ГСА.

Алгоритм построения графа микропрограммного автомата Мура S :

1. В качестве состояний автомата выбрать метки $\{a_1, \dots, a_M\}$, a_1 – начальное состояние.
2. Найти все пути перехода на отмеченной ГСА. Отметим, что при синтезе автомата Мура путь перехода имеет вид:

$$a_m x_{m_1}^{e_{m_1}} \wedge \dots \wedge x_{m_R}^{e_{m_R}} a_s = a_m X(a_m, a_s) a_s. \quad (4.11)$$

3. Каждому пути перехода $a_m X(a_m, a_s) a_s$ поставить в соответствие переход автомата S из состояния a_m в состояние a_s под действием входного сигнала $X(a_m, a_s)$. Если $R = 0$ (логическое условие отсутствует), то входной сигнал равен «1».

4. Каждому состоянию приписать выходной сигнал, соответствующий микрооперациям, записанным в операторной вершине, которая отмечена символом этого состояния.

Замечание 4.5. При построении графа микропрограммного автомата Мура не рассматриваются пути перехода вида $a_m X(a_m, a_m) a_m$, так как при таком переходе ни состояние, ни выходной сигнал не меняются (см. рис. 4.9).

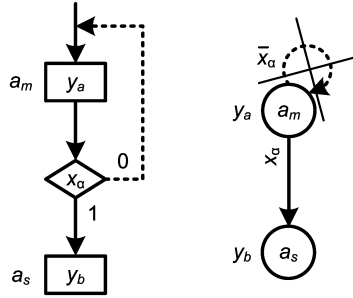


Рис. 4.9. Путь перехода $a_m X(a_m, a_m) a_m$

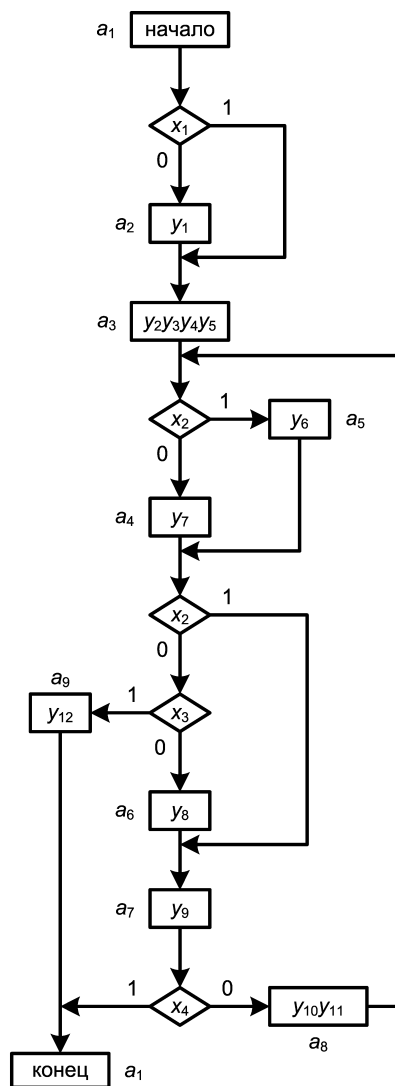


Рис. 4.10. Отмеченная ГСА при синтезе микропрограммного автомата Мура

Остаются в силе замечания 4.1, 4.2, 4.3.

Замечание 4.6. В общем случае автомат Мура имеет большее число состояний, чем автомат Мили, соответствующий той же ГСА. Следовательно, автомат Мили более экономичен, и для устройства управления предпочтительнее схема Мили, кроме тех случаев, которые отмечены в замечании 4.4.

Пример 4.4. Отмеченная ГСА и соответствующий ей граф микропрограммного автомата Мура представлены на рисунках 4.10, 4.11.

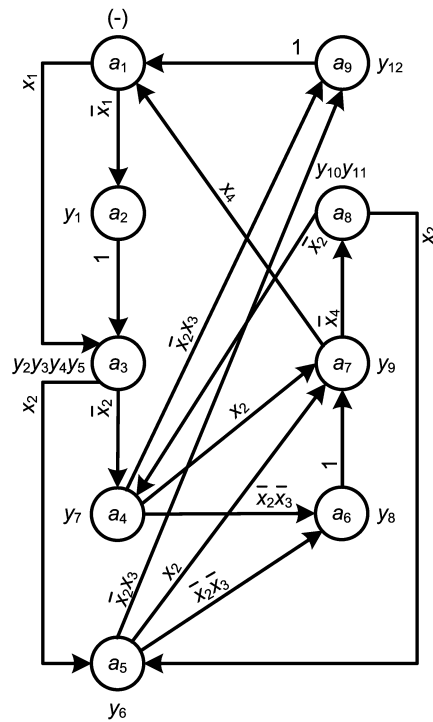


Рис. 4.11. Граф микропрограммного автомата Мура

4.5. Списки переходов микропрограммных автоматов

На практике микропрограммные автоматы, используемые в управляющих устройствах, имеют несколько десятков состояний. В связи с этим графы автоматов теряют свою наглядность. Вместо них используются *списки переходов микропрограммных автоматов*. Виды таких списков для моделей Мили и Мура рассмотрим на примерах.

Отметим, что списки переходов микропрограммных автоматов можно строить непосредственно по ГСА микропрограммы по тем же принципам, что и графы автоматов.

Пример 4.5. *Прямой список переходов микропрограммного автомата Мили* из примера 4.3 представлен в таблице 4.1.

Для каждого «исходного состояния» в колонке «состояние перехода» перечисляются все состояния, в которые возможен из него переход.

Таблица 4.1. Прямой список переходов микропрограммного автомата Мили

Исходное состояние	Состояние перехода	Входной сигнал	Выходной сигнал
a_1	a_2	\overline{x}_1	y_1
	a_3	x_1	$y_2y_3y_4y_5$
a_2	a_3	1	$y_2y_3y_4y_5$
a_3	a_4	x_2	y_6
	a_4	\overline{x}_2	y_7
a_4	a_1	\overline{x}_2x_3	y_{12}
	a_5	$\overline{x}_2\overline{x}_3$	y_8
	a_6	x_2	y_9
a_5	a_6	1	y_9
a_6	a_1	x_4	—
	a_3	\overline{x}_4	$y_{10}y_{11}$

Пример 4.6. Обратный список переходов микропрограммного автомата Мили из примера 4.3 представлен в таблице 4.2.

Для каждого «состояния перехода» в колонке «исходное состояние» перечисляются все состояния, из которых возможен в него переход.

Таблица 4.2. Обратный список переходов микропрограммного автомата Мили

Исходное состояние	Состояние перехода	Входной сигнал	Выходной сигнал
a_4	a_1	$\overline{x}_2 x_3$	y_{12}
a_6		x_4	—
...

Пример 4.7. Прямой список переходов микропрограммного автомата Мура из примера 4.4 представлен в таблице 4.3.

Таблица 4.3. Прямой список переходов микропрограммного автомата Мура

Исходное состояние	Состояние перехода	Входной сигнал
$a_1 (-)$	a_2	\overline{x}_1
	a_3	x_1
$a_2 (y_1)$	a_3	1
...

Пример 4.8. Обратный список переходов микропрограммного автомата Мура из примера 4.4 представлен в таблице 4.4.

Отличие прямого и обратного списков переходов микропрограммного автомата Мура от случая схемы Мили заключается в том, что выходной сигнал приписывается непосредственно либо исходному состоянию (прямой список), либо состоянию перехода (обратный список).

Таблица 4.4. Обратный список переходов микропрограммного автомата Мура

Исходное состояние	Состояние перехода	Входной сигнал
a_7 a_9	$a_1 (-)$	x_4 1
a_1	$a_2 (y_1)$	\bar{x}_1
...

Упражнение 4.1. Постройте обратный список переходов микропрограммного автомата Мили из примера 4.6.

Упражнение 4.2. Постройте прямой список переходов микропрограммного автомата Мура из примера 4.7.

Упражнение 4.3. Постройте обратный список переходов микропрограммного автомата Мура из примера 4.8.

Упражнение 4.4. Почему для микропрограммного автомата, построенного по ГСА, всегда выполнено условие детерминированности?

4.6. Синтез микропрограммного С-автомата

Рассмотрим устройство, моделью которого является С-автомат. Поведение такого устройства описывается ГСА, внутри операторных вершин которой наряду с короткими микрооперациями $\{y_1, \dots, y_N\}$ могут присутствовать длинные микрооперации $\{r_1, \dots, r_D\}$. Тогда каждой операторной вершине соответствует пара микрокоманд $Y_i R_j$ ($Y_i = \{y_{i_1}, \dots, y_{i_U}\}$, $R_j = \{r_{j_1}, \dots, r_{j_W}\}$). При этом микрокоманда Y_i выдается на переходе автомата из одного состояния в другое, микрокоманда R_j выдается в течение всего времени нахождения автомата в соответствующем состоянии.

Пример 4.9. Схематично построение графа С-автомата по ГСА микропрограммы представлено на рисунке 4.12. Пусть в

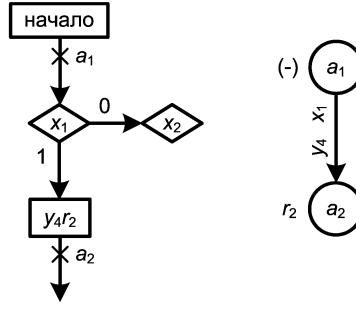


Рис. 4.12. Переход от ГСА к С-автомату

состоянии a_1 на вход автомата пришел сигнал $x_1 = 1$. Тогда (если запускающий сигнал $B = 1$) автомат перейдет в состояние a_2 и выдаст выходной сигнал y_4 , который по длительности совпадет с синхроимпульсом; в состоянии a_2 автомат будет выдавать сигнал r_2 .

В общем случае построение микропрограммного С-автомата состоит из двух этапов:

1. *Построение микропрограммного автомата Мили S'* . На этом этапе входы всех вершин, следующих за начальной и операторными, а также вход конечной вершины на ГСА отмечаются символами $\{a_1, \dots, a_M\}$. Эти отметки отождествляются с состояниями автомата Мили S' . Строится микропрограммный автомат Мили, в котором не делаются отличия между микрооперациями y_n и r_d . Такой автомат представляется обратным списком переходов.

2. *Построение микропрограммного С-автомата S* . Этот этап состоит из нескольких шагов.

- 2.1. Каждому состоянию перехода a_s из второго столбца списка ставится в соответствие множество $B_s = \{(a_s, R_j) | R_j - \text{длинная микрокоманда из списка выходных сигналов}\}$.

- 2.2. Определяется множество $B = \bigcup_{s=1}^M B_s$ – множество состояний С-автомата S .

- 2.3. Каждому состоянию $b_k = (a_s, R_j) \in B$ приписывается

длинный выходной сигнал R_j .

2.4. Каждой строке списка $a_m \ a_s \ X(a_m, a_s) \ Y_i \ R_j \ b_k$ ставятся в соответствие переходы автомата S из всех состояний $b_{m_k} \in B_m$ в состояние b_k под действием входного сигнала $X(a_m, a_s)$ с выдачей короткого выходного сигнала Y_i .

В итоге получаем обратный список переходов микропрограммного С-автомата S .

Пример 4.10. Построим по обратному списку переходов микропрограммного автомата Мили S' , содержащего длинные и короткие микрооперации, микропрограммный С-автомат S .

Вначале определим состояния С-автомата (см. табл. 4.5).

Таблица 4.5. Расширение обратного списка переходов автомата Мили S'

Исх. сост.	Сост. перех.	Входн. сигн.	Выход. сигн.	Состояния С-автомата
a_2	a_1	x_3x_2	y_1	$b_1 = (a_1, -)$
a_1		x_4x_2	y_1	$b_1 = (a_1, -)$
a_1	a_2	x_1	y_4r_2	$b_2 = (a_2, r_2)$
a_1		\overline{x}_1x_2	$y_2r_1r_3$	$b_3 = (a_2, r_1r_3)$
a_1		$\overline{x}_1\overline{x}_2$	y_5	$b_4 = (a_2, -)$
a_2		\overline{x}_3	$y_1y_2r_2$	$b_2 = (a_2, r_2)$

Здесь $B_1 = \{(a_1, -)\} = \{b_1\}$,

$B_2 = \{(a_2, r_2), (a_2, r_1r_3), (a_2, -)\} = \{b_2, b_3, b_4\}$.

Следуя вышеприведенному алгоритму, построим обратный список переходов микропрограммного С-автомата (см. табл. 4.6).

В заключении параграфа сформулируем ряд замечаний, касающихся вопросов минимизации и кодирования состояний микропрограммных автоматов.

Замечание 4.7. Минимизация полностью определенных микропрограммных автоматов проводится *методом последовательных разбиений*. При этом автомат удобно представлять прямым списком переходов.

Таблица 4.6. Обратный список переходов С-автомата S

Исх. сост.	Сост. перех.	Входн. сигн.	Выход. сигн.
b_2	$b_1(-)$	x_3x_2	y_1
b_3		x_3x_2	y_1
b_4		x_3x_2	y_1
b_1		x_4x_2	y_1
b_1	$b_2(r_2)$	x_1	y_4
b_2		\overline{x}_3	y_1y_2
b_3		\overline{x}_3	y_1y_2
b_4		\overline{x}_3	y_1y_2
b_1	$b_3(r_1r_3)$	\overline{x}_1x_2	y_2
b_1	$b_4(-)$	$\overline{x}_1\overline{x}_2$	y_5

Замечание 4.8. К микропрограммным автоматам применима модификация *алгоритма развязывания пар переходов* (метод противогоночного кодирования состояний). Пусть в автомате есть пара переходов (a_m, a_s) и (a_k, a_l) , и эти переходы происходят под действием входных сигналов $X(a_m, a_s)$ и $X(a_k, a_l)$. Эта пара переходов должна быть развязана, если $X(a_m, a_s) \wedge X(a_k, a_l) \neq \emptyset$ (то есть, существует хотя бы один набор входных переменных, на котором $X(a_m, a_s)$ и $X(a_k, a_l)$ равны единице) и $\{a_m, a_s\} \cap \{a_k, a_l\} = \emptyset$.

Замечание 4.9. К микропрограммным автоматам применим *эвристический алгоритм кодирования состояний*, уменьшающий сложность логической схемы.

4.7. Построение логической схемы микропрограммного автомата

Синтез логической схемы микропрограммного автомата разобьем на два этапа.

1. Построение *структурного списка микропрограммного автомата*. Структурный список представляет собой расшире-

ние прямого или обратного списка переходов микропрограммного автомата (см. табл. 4.7). Для каждого пути перехода (да-

Таблица 4.7. Структурный список микропрограммного автомата

Исх. сост.	Код исх.сост.	Сост. перех.	Код сост.перех.
a_m	$K(a_m)$	a_s	$K(a_s)$

Входн. сигн.	Выход. сигн.	Ф-ции возб.
$X(a_m, a_s)$	$Y(a_m, a_s)$	$F(a_m, a_s)$

же если выходы совпадают) отводится одна строка в структурном списке. Таким образом, столбец «Входной сигнал» в каждой строке содержит одну конъюнкцию. Структурный список может быть как прямым, так и обратным. Для модели Мили он состоит из 7-ми столбцов. Для автомата Мура структурный список содержит на один столбец меньше (отсутствует столбец «Выходной сигнал»).

Пример 4.11. Структурный список, построенный по прямому списку переходов микропрограммного автомата Мили из примера 4.5 при синтезе на триггерах с отдельными входами, представлен в таблице 4.8.

Таблица 4.8. Прямой структурный список микропрограммного автомата Мили

Исх. сост.	Код	Сост. перех.	Код	Входн. сигн.	Выход. сигн.	Ф-ции возб.
a_1	001	a_2	010	\bar{x}_1	y_1	$\varphi_2\psi_3$
		a_3	011	x_1	$y_2y_3y_4y_5$	φ_2
a_2	010	a_3	011	1	$y_2y_3y_4y_5$	φ_3
...
a_6	110	a_1	001	x_4	—	$\psi_1\psi_2\varphi_3$
		a_3	011	\bar{x}_4	$y_{10}y_{11}$	$\psi_1\varphi_3$

Заметим, что правила заполнения столбца, соответствующего функциям возбуждения, совпадают с аналогичными правилами подписи дуг графа при графической интерпретации канонического метода (см. § 2.6).

2. *Построение логической схемы по структурному списку.* Напомним, что логическая схема автомата строится по *канонической системе уравнений*. После составления структурного списка, построение функций выходов и функций возбуждения памяти, составляющих эту систему, аналогично графическому методу синтеза структурных автоматов (см. § 2.6).

Пример 4.12. Используя структурный список примера 4.11, нетрудно выписать каноническую систему уравнений. В частности:

$$\varphi_3 = \bar{\tau}_1 \tau_2 \bar{\tau}_3 \vee \tau_1 \tau_2 \bar{\tau}_3 x_4 \vee \tau_1 \tau_2 \bar{\tau}_3 \bar{x}_4 \vee \dots = \tau_2 \bar{\tau}_3 \vee \dots$$

$$y_2 = \bar{\tau}_1 \bar{\tau}_2 \tau_3 x_1 \vee \bar{\tau}_1 \tau_2 \bar{\tau}_3 \vee \dots$$

К сожалению, для встречающихся на практике микропрограммных автоматов уравнения канонической системы (системы ДНФ) очень сложны как по числу уравнений, так и по количеству переменных. А следовательно, неприемлемы известные методы минимизации булевых функций в классе нормальных форм. Практика показала, что основное сокращение объема логической схемы происходит не за счет минимизации в нормальных формах, а в результате выделения функций и подфункций, допускающих совместную минимизацию; представления системы функций в виде декомпозиции и т. д.

Метод синтеза структуры автомата, основанный на непосредственной интерпретации структурного списка микропрограммного автомата элементами логической схемы, называется *интерпретационным методом*. Как правило, используются обратные списки.

Упражнение 4.5. Для микропрограммного автомата Мили (из примера 4.3) и микропрограммного автомата Мура (из

примера 4.4) постройте прямые и обратные структурные списки. Выбор элементов памяти – произволен. Для каждого списка выпишите каноническую систему уравнений.

4.8. Интерпретационный метод структурного синтеза

Интерпретационный метод структурного синтеза микропрограммного автомата представляет собой совокупность различных способов построения и минимизации логической схемы. При этом **критерием минимальности схемы** будем считать минимум суммарного числа входов на всех логических элементах.

4.8.1. Объединенное построение компонент функций возбуждения памяти и функций выходов

Первым шагом при минимизации логической схемы микропрограммного автомата является объединенное построение компонент функций возбуждения и функций выходов, записанных в одной строке структурного списка.

Пример 4.13. Рассмотрим одну строку структурного списка микропрограммного автомата (табл. 4.9).

Таблица 4.9. Строка структурного списка

	$\tau_1 \tau_2 \tau_3$					
a_2	001	a_1	000	$\bar{x}_1 x_4 \bar{x}_6$	$y_4 y_8$	ψ_3

Если объединить построение компонент для функций выходов и функций возбуждения памяти, то этой строке будет соответствовать фрагмент логической схемы, представленный на рисунке 4.13.

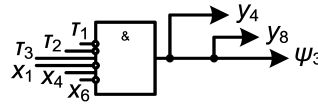


Рис. 4.13. Общая компонента функций выходов и функций возбуждения памяти

Учет только этого обстоятельства позволяет сократить логическую схему по сравнению с канонической примерно в ξ раз, где ξ равно среднему числу выходных сигналов и функций возбуждения в одной строке структурного списка.

4.8.2. Раздельный синтез для переходов в каждое состояние

Будем рассматривать обратный структурный список микропрограммного автомата. Синтез схемы можно проводить раздельно для переходов в каждое состояние.

В пределах переходов в одно состояние строится схема для каждой микрокоманды.

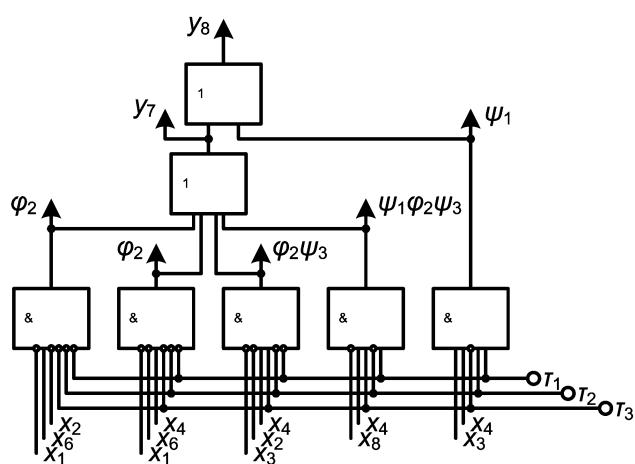
Конъюнкции, соответствующие одной микрокоманде, заводятся на элемент «ИЛИ». С этого элемента снимаются соответствующие микрокоманде выходные сигналы. Если пересечение микрокоманд не пусто, можно поставить еще один элемент «ИЛИ».

Пример 4.14. Рассмотрим фрагмент обратного структурного списка микропрограммного автомата, соответствующий переходу в одно состояние (табл. 4.10).

Следуя вышеприведенными правилами, построим фрагмент логической схемы для двух микрокоманд: $Y_1 = \{y_7, y_8\}$ и $Y_2 = \{y_8\}$. Соответствующая логическая подсхема изображена на рисунке 4.14.

Таблица 4.10. Фрагмент обратного структурного списка, соответствующий переходу в состояние a_5

a_1	000	a_5	010	$\bar{x}_1 x_6 \bar{x}_2$	$y_7 y_8$	φ_2
a_1	000			$\bar{x}_1 \bar{x}_6 x_4$	$y_7 y_8$	φ_2
a_2	001			$\bar{x}_3 \bar{x}_2 x_4$	$y_7 y_8$	$\varphi_2 \psi_3$
a_4	101			$\bar{x}_8 x_4$	$y_7 y_8$	$\psi_1 \varphi_2 \psi_3$
a_7	110			$x_3 x_4$	y_8	ψ_1

Рис. 4.14. Логическая подхема, соответствующая переходу в состояние a_5

4.8.3. Декомпозиция

Рассмотрим теперь *декомпозицию конъюнкций* (или *термов*). Эту методику разберем на примере.

Пример 4.15. Рассмотрим фрагмент обратного структурного списка микропрограммного автомата, представленный в таблице 4.11. Подсхема, интерпретирующая этот список и от-

Таблица 4.11. Фрагмент обратного структурного списка

a_1	0011	a_3	0101	$x_1\bar{x}_3\bar{x}_4$	y_1	$\varphi_2\psi_3$
a_1	0011			$x_1\bar{x}_3x_4$	y_1y_2	$\varphi_2\psi_3$
a_2	0100			$x_2\bar{x}_3\bar{x}_4$	y_1	φ_4
a_2	0100			$x_2\bar{x}_3x_4$	y_1y_2	φ_4
a_1	0011	a_4	1101	$x_1x_3\bar{x}_4$	y_1y_3	$\varphi_1\varphi_2\psi_3$
a_1	0011			$x_1x_3x_4$	y_2y_3	$\varphi_1\varphi_2\psi_3$
a_2	0100			$x_2x_3\bar{x}_4$	y_1y_3	$\varphi_1\varphi_4$
a_2	0100			$x_2x_3x_4$	y_2y_3	$\varphi_1\varphi_4$

вечающая за формирование конъюнкций (термов), из которых в дальнейшем строятся функции выходов и функции возбуждения, представлена на рисунке 4.15 (1).

Процесс декомпозиции заключается в следующем. Обозначим каждый терм символом T_i :

$$\begin{aligned}
 T_1 &= a_1x_1\bar{x}_3\bar{x}_4 = t_1t_3, & T_5 &= a_1x_1x_3\bar{x}_4 = t_1t_5, \\
 T_2 &= a_1x_1\bar{x}_3x_4 = t_1t_4, & T_6 &= a_1x_1x_3x_4 = t_1t_6, \\
 T_3 &= a_2x_2\bar{x}_3\bar{x}_4 = t_2t_3, & T_7 &= a_2x_2x_3\bar{x}_4 = t_2t_5, \\
 T_4 &= a_2x_2\bar{x}_3x_4 = t_2t_4, & T_8 &= a_2x_2x_3x_4 = t_2t_6.
 \end{aligned}$$

Заметим, что термы могут содержать одинаковые наборы. В нашем случае:

$$\begin{aligned}
 t_1 &= a_1x_1, & t_3 &= \bar{x}_3\bar{x}_4, & t_5 &= x_3\bar{x}_4, \\
 t_2 &= a_2x_2, & t_4 &= \bar{x}_3x_4, & t_6 &= x_3x_4.
 \end{aligned}$$

Утверждение 4.1. Пусть набор t_i содержит n элементов и используется в m термах. Если $m > \frac{n}{n-1}$, то вычисление набора t_i на отдельном элементе «И» уменьшит цену

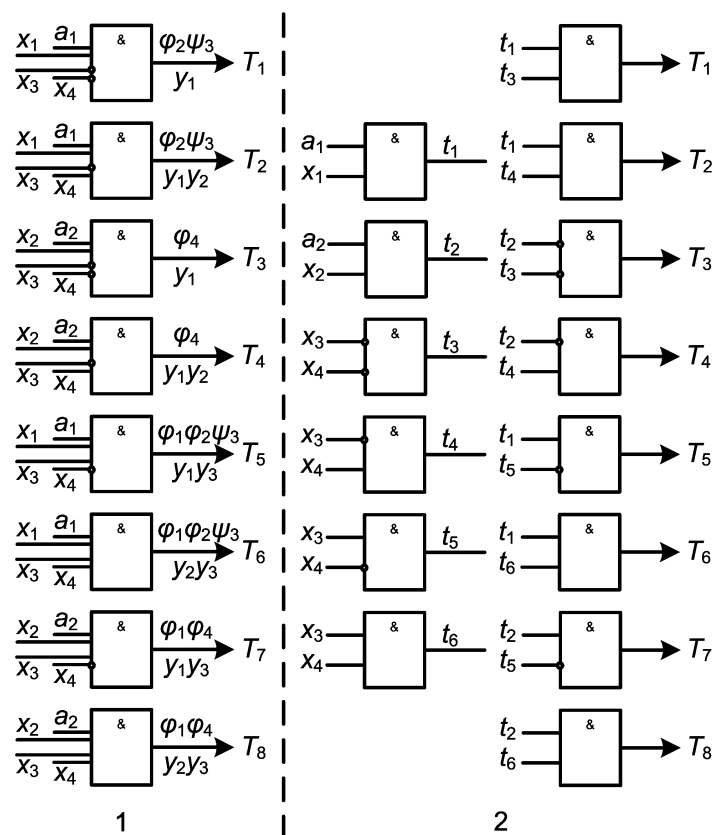


Рис. 4.15. Логические подсхемы, соответствующие формированию термов: 1 – подсхема, полученная непосредственно по структурному списку; 2 – подсхема, полученная методом декомпозиции

набора (в смысле суммарного числа входов на логических элементах).

Доказательство. Из данного неравенства следует, что $(m \cdot n) > (m + n)$. Но $(m \cdot n)$ – это общее число входов при каноническом синтезе (m элементов «И» по n входов на каждом), в свою очередь, $(m + n)$ – суммарное число входов при декомпозиции (на каждый из m элементов «И» по одному входу от нового элемента + n входов на этот новый элемент).

Пример 4.15 (продолжение). Фрагмент логической схемы, полученный методом декомпозиции, представлен на рисунке 4.15 (2). Цена такой подсхемы равна 28. В то время как цена подсхемы 4.15 (1) равна 32.

4.8.4. Доопределение функций возбуждения

Метод доопределения функций возбуждения также разберем на примере. Отметим, что данный подход применим при синтезе автомата на *триггерах с раздельными входами*.

Пример 4.16. Рассмотрим фрагмент автомата, синтезируемого на триггерах с раздельными входами (рис. 4.16). Пусть $F(a_m, a_s)$ – множество функций возбуждения на пере-

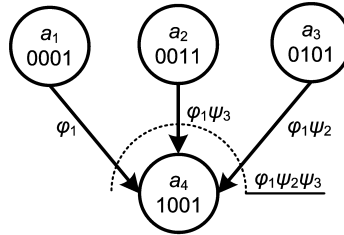


Рис. 4.16. Фрагмент графа автомата при синтезе на триггерах с раздельными входами

ходе (a_m, a_s) . В нашем примере $F(a_1, a_4) = \{\varphi_1\}$, $F(a_2, a_4) = \{\varphi_1, \psi_3\}$, $F(a_3, a_4) = \{\varphi_1, \psi_2\}$.

Функции возбуждения поступают на входы элементов памяти через элементы «ИЛИ». Число входов элементов «ИЛИ», соответствующих переходам в состояние a_s , равно числу функций возбуждения, перечисленных во множествах $F(a_{m_1}, a_s), \dots, F(a_{m_k}, a_s)$. С целью уменьшения цены логической подсистемы, обрабатывающей функции возбуждения, на всех переходах в каждое состояние a_s можно формировать единственный набор функций возбуждения:

$$F(a_s) = \bigcup_{i=1}^k F(a_{m_i}, a_s). \quad (4.12)$$

В частности, $F(a_4) = \{\varphi_1, \psi_2, \psi_3\}$.

Полученное множество называется **доопределением функций возбуждения** автомата. Оно избыточно для каждого отдельного перехода, но минимально по отношению ко всему множеству переходов.

Очевидно, что в результате доопределения функций возбуждения функционирование автомата не нарушается. Например, если при переходе из состояния «0001» в состояние «1001» вместо φ_1 формировать функции возбуждения $\{\varphi_1, \psi_2, \psi_3\}$, то состояния второго и третьего триггеров по-прежнему не изменятся. То есть, на каждом переходе кроме обязательных, можно вырабатывать функции возбуждения, подтверждающие состояния элементов памяти, не меняющиеся на данном переходе.

Пример 4.17. Вернемся к подавтомату, представленному в таблице 4.11. Построим подсистему, обрабатывающую функции возбуждения.

При стандартном подходе получаем подсистему, изображенную на рисунке 4.17 (1). Ее цена равна 16. Так как входы функций φ_2 и ψ_3 совпадают, то очевидным образом цену можно понизить до 12.

Теперь применим метод доопределения функций возбуждения к таблице 4.11:

$$F(a_3) = \varphi_2 \psi_3 \varphi_4, F(a_4) = \varphi_1 \varphi_2 \psi_3 \varphi_4,$$

$$F(a_3) = T_1 \vee T_2 \vee T_3 \vee T_4, F(a_4) = T_5 \vee T_6 \vee T_7 \vee T_8.$$

В итоге получаем под схему, представленную на рисунке 4.17 (2). Цена этой под схемы равна 10.

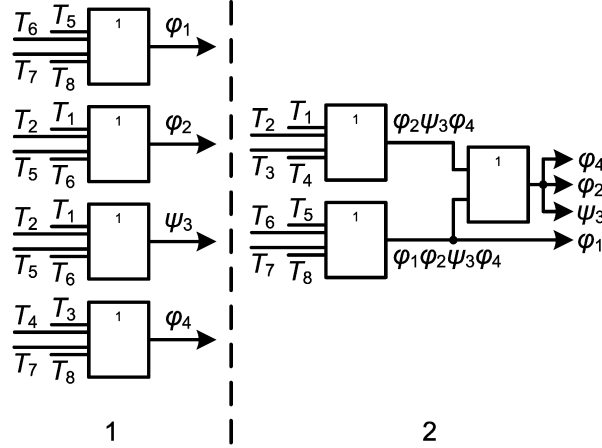


Рис. 4.17. Логические под схемы, обрабатывающие функции возбуждения: 1 – под схема, непосредственно интерпретирующая структурный список; 2 – под схема, полученная методом доопределения функций возбуждения

Замечание 4.10. Существуют и другие способы минимизации логической схемы (например, преддешефратор обратной связи [1], учет узлов ГСА микропрограммы и сокращение структурного списка [1, 5], факторизация [1]), которые вместе с вышеперечисленными составляют интерпретационный метод.

4.9. Устройство управления с программируемой логикой

Возвращаясь к структуре дискретного устройства, напомним, что оно состоит из двух частей: устройства управления (УУ) и операционного устройства (ОУ).

В структурном отношении ОУ состоит из *операционных элементов*, каждый из которых представляет собой последовательный или комбинационный автомат.

В свою очередь, со структурной точки зрения, УУ можно разделить на два типа:

1. УУ с *жесткой логикой* – для формирования последовательности микрокоманд, которая соответствует последовательности логических условий (то есть для задания микропрограммы), используется последовательный автомат (см. §§ 4.3–4.8).

2. УУ с *программируемой логикой* построено на основе принципа программного управления: упорядоченная последовательность микрокоманд хранится в некотором массиве (ПЗУ).

Определим простейшую структуру *управляющих слов* (управляющих сигналов или микрокоманд). Пусть, как и прежде, $Y = \{y_1, \dots, y_N\}$ – множество выходных сигналов, которые возбуждают микрооперации, реализуемые затем операционным устройством. Также предположим, что на каждом такте совместно выполняется не более, чем H микроопераций. $X = \{x_1, \dots, x_L\}$ – множество осведомительных сигналов (логических условий).

Тогда микрооперации или, точнее, возбуждающие их сигналы, можно закодировать двоичными числами, количество разрядов которых определяется числом $n = \lceil \log_2 N \rceil$.

Разобьем управляющее слово на две части: *операционную часть* (ОЧ), где будут представлены одновременно выполняемые микрооперации, и *адресную часть* (АЧ), где задан адрес слова, описывающего следующую микрокоманду – так называемая *принудительная адресация* (см. рис. 4.18).

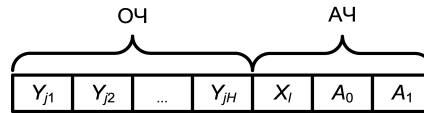


Рис. 4.18. Структура микрокоманды

ОЧ состоит из H полей по n разрядов каждое. В каждом поле стоит значение Y_{jh} , равное *номеру микрооперации*. Если же $Y_{jh} = 0$ то никакая микрооперация данным полем не возбуждается.

Адрес следующей микрокоманды может задаваться без-условно, а может зависеть от текущих значений осведомительных сигналов x_i . Примем, что в каждой микрокоманде можно задавать значение только одного условия из X . Тогда АЧ начинается с поля X_l – *номер оповещающего сигнала* (это поле содержит $\lceil \log_2 L \rceil$ разрядов). Если все разряды данного поля чистые, то следующей выполняется микрокоманда с адресом, который стоит в поле A_0 . В противном случае проверяется значение указанного оповещающего сигнала x_i . Если $x_i = 0$, то следующей выполняется микрокоманда с адресом из поля A_0 . Если же $x_i = 1$, то следующей будет выполняться микрокоманда, адрес которой прописан в поле A_1 . Длины полей A_0 , A_1 определяются числом микрокоманд, составляющих микропрограмму, то есть логарифмом от числа микрокоманд P .

Для хранения микрокоманд используется *постоянное запоминающее устройство* (ПЗУ). Подробно структура ПЗУ и особенности хранения микрокоманд описаны в [5].

Таким образом, *устройство управления с программируемой логикой* реализует алгоритм управления на основе хранимой в ПЗУ микропрограммы.

В заключении параграфа отметим, что устройства управления с жесткой логикой быстрее устройств управления с программируемой логикой, но наряду с этим они менее гибки и более затратны по оборудованию (особенно при реализации сложных микропрограмм).

Глава 5

Эксперименты с автоматами

5.1. Классификация экспериментов

Реакция последовательного автомата S на определенное воздействие непредсказуема, если не известно состояние автомата. В связи с этим возникает следующая **задача анализа автомата**: распознать состояние автомата

- а) начальное;
- б) конечное.

Для решения этой проблемы с автоматом проводится **эксперимент** – процесс приложения входных последовательностей к автомату, наблюдения получаемых выходных последовательностей и вывода заключений. Заключения об автомате можно сделать только на основе

- а) приложенных воздействий;
- б) наблюдаемых реакций;
- в) таблиц переходов и выходов (графа, матрицы) автомата.

Эксперименты классифицируются по следующим признакам:

1. *Безусловные и условные эксперименты.* В **безусловных** экспериментах входная последовательность полностью определена заранее (такие эксперименты легче осуществить, но в некоторых случаях они не позволяют решить задачу анализа). В **условных** экспериментах входная последовательность состоит из нескольких подпоследовательностей; все подпоследовательности, кроме первой, определяются на основе реакции на предыдущие подпоследовательности.

2. *Простые и кратные эксперименты.* Для **простых** экспериментов требуется единственный экземпляр автомата (такие эксперименты предпочтительнее). Для **кратных** экспериментов необходимо более одного экземпляра автомата.

3. *Диагностический и установочный эксперименты.* **Диагностический** эксперимент решает задачу определения начального состояния (*диагностическую задачу*): известна объединенная таблица переходов автомата S (таблица переходов и выходов); известно, что автомат находится в одном из состояний a_{i_1}, \dots, a_{i_m} ; найти это состояние. **Установочный** эксперимент решает задачу определения конечного состояния (*установочную задачу*): в отличие от предыдущего случая требуется найти состояние, в котором оказался автомат после проведения эксперимента, или установить автомат в заданное состояние.

Исходя из классификации, выделяют ряд *критериев стоимости эксперимента*:

1. Длина эксперимента – общее число входных символов.
2. Порядок эксперимента – число входных подпоследовательностей, из которых состоит эксперимент.
3. Кратность эксперимента – число требуемых экземпляров автомата.

Далее нам понадобятся следующие обозначения. Пусть $A(S) = \{a_{i_1}, \dots, a_{i_m}\}$ – некоторое подмножество множества состояний автомата S такое, что начальное состояние $a_0 \in A(S)$. $A(S)$ назовем **множеством допустимых начальных состояний**. Будем считать, что $m \geq 2$.

5.2. Простой безусловный диагностический эксперимент

В этом параграфе рассмотрим метод поиска диагностической последовательности для проведения простого безусловного диагностического эксперимента. Прежде всего дадим ряд определений.

σ -множество автомата S – это любое конечное множество состояний автомата (причем состояния не обязательно различны, то есть в общем случае σ -множество является некоторой совокупностью состояний). Если мощность σ -множества равна единице, то оно называется **простым**; если σ -множество содержит несколько одинаковых элементов, то оно называется **кратным**.

A -группа – это множество σ -множеств автомата S такое, что число всех элементов из σ -множеств A -группы равно m , где $m = |A(S)|$ – мощность множества допустимых начальных состояний. A -группа называется **простой**, если все входящие в нее σ -множества простые.

Пусть $\{z_1, \dots, z_F\}$ – входной алфавит; $z_{i_1} z_{i_2} \dots z_{i_f} = \xi$ – некоторая входная последовательность; G – A -группа, которая содержит σ -множества $\sigma_1, \dots, \sigma_r$.

ξ -преемник G – это другая A -группа, построенная по следующим правилам:

1. Каждое σ -множество σ_j ($j = 1, \dots, r$) разбивается на такие подмножества, что два состояния попадают в одно подмножество, если они *вырабатывают* одинаковые реакции на входную последовательность ξ (используется таблица выходов автомата S). Из полученных подмножеств получается A -группа G' .

2. В σ -множествах из G' каждое состояние заменяется его *преемником* относительно последовательности ξ (используется таблица переходов автомата S). Полученная A -группа и будет ξ -преемником G .

Дерево преемников – структура, определенная для данного автомата S из заданного множества допустимых началь-

ных состояний $A(S)$. Дерево преемников состоит из узлов, распределенных по уровням. Высший уровень – нулевой, он содержит единственный начальный узел.

В дереве преемников каждый узел $(k - 1)$ -го уровня расщепляется на F узлов (по числу входных сигналов) k -го уровня. Узел, соответствующий входному сигналу z_f на уровне k , называется узлом « z_f^k ».

Каждый узел в дереве преемников связан с A -группой. Начальному узлу приписывается множество $A(S)$. A -группы, связанные с узлами k -го уровня, определяются из A -групп $(k - 1)$ -го уровня как преемники: если узел « z_i^{k-1} » связан с A -группой G , то узел « z_j^k », который порождается из « z_i^{k-1} », связывается с z_j -преемником G .

Очевидно, что дерево преемников по своему определению бесконечно. В связи с этим будем рассматривать **усеченное дерево**, используя **правила завершения**. Эти правила определяют, когда можно остановить построение очередной ветви дерева. Узел « z_i^k » k -го уровня становится листом (последним узлом), если выполнено одно из условий:

1. A -группа, связанная с узлом « z_i^k », содержит кратное σ -множество.
2. A -группа, связанная с узлом « z_i^k », совпадает с A -группой некоторого узла, расположенного на одном из предыдущих уровнях.
3. Имеется узел k -го уровня (возможно сам « z_i^k »), связанный с простой A -группой.

Усеченное дерево, построенное с использованием данных правил завершения, называется **диагностическим деревом**.

Диагностический путь – это любой путь в диагностическом дереве, соединяющий начальный узел и конечный узел, связанный с простой A -группой.

Несложно показать, что по диагностическому пути можно выписать **диагностическую последовательность** (перечисляя узлы, вошедшие в этот путь), которая характеризуется следующим свойством: *если на вход автомата S подать ди-*

агностическую последовательность, выбирая в качестве начальных состояний состояния a_{i_1}, \dots, a_{i_m} из $A(S)$, то на выходе автомата получим m различных выходных последовательностей.

Отсюда следует вывод: построение диагностического дерева и выбор диагностической последовательности дает возможность провести простой безусловный диагностический эксперимент (при условии, что такая последовательность существует).

Пример 5.1. Рассмотрим автомат S , заданный таблицами переходов и выходов (табл. 5.1, 5.2).

Таблица 5.1. Таблица переходов автомата S

	1	2	3	4	5	6	7	8	9
a	1	1	5	3	2	7	8	5	2
b	4	5	1	4	6	3	4	9	8

Таблица 5.2. Таблица выходов автомата S

	1	2	3	4	5	6	7	8	9
a	0	0	0	1	1	1	0	1	0
b	1	1	1	1	1	0	1	1	0

1. Пусть $A(S) = \{4, 5, 6, 7, 9\}$. Диагностическое дерево T_1 изобразим в виде таблицы (см. табл. 5.3). В частности, чтобы получить первого a -преемника множества $A(S)$, по таблице выходов строим множество $G' = \{\{4, 5, 6\}, \{7, 9\}\}$, а затем по таблице переходов получаем новую A -группу, приписанную узлу « a^1 »: $\{\{3, 2, 7\}, \{8, 2\}\}$.

A -множества, содержащие σ -группы, выделенные в таблице жирным шрифтом, удовлетворяют правилам завершения: σ -группы $\{4, 6, 4\}$ и $\{4, 4, 6\}$ – кратные, A -множество $\{\{1\}, \{2\}, \{1\}, \{1\}, \{1\}\}$ – простое. Наличие простого A -множества в диагностическом дереве позволяет построить диагностическую последовательность: « $aaaa$ ».

Таблица 5.3. Диагностическое дерево T_1

0	{4, 5, 6, 7, 9}					
	a				b	
1	{3, 2, 7}{8, 2}					{4, 6, 4}{3, 8}
	a			b		
2	{1, 5, 8}{5}{1}			{5, 1, 4}{5, 9}		
	a		b		a	b
3	{2, 5}{1}		{4, 6, 9}		{1}{3, 2}	
	{2}{1}		{6}{4}		{2}{2}	
	a	b	a	b	a	b
4	{1}	{5, 6}	{3, 7}	{3, 8}	{1}	{4}
	{2}	{4}	{2}	{4}	{1, 5}	{1, 5}
	{1}	{5}	{7}	{3}	{1}	{5}
	{1}	{4}	{3}	{4}	{1}	{5}
	{1}					

Таблица 5.4. Определение начального состояния автомата S

Начальное состояние	Реакция на «aaaa»
4	1 0 1 0
5	1 0 0 0
6	1 0 1 1
7	0 1 1 0
9	0 0 0 0

Анализируя реакцию автомата на диагностическую последовательность, можно определить в каком именно состоянии находился автомат на момент начала работы (см. табл. 5.4).

2. Пусть теперь $A(S) = \{1, 2, 3, 4, 5\}$. Диагностическое дерево T_2 представлено таблицей 5.5. Оба A -множества первого уровня являются кратными. Очевидно, что в этом случае дерево не содержит диагностическую последовательность, тем самым, для заданного $A(S)$ не существует решения диагностической задачи с помощью простого безусловного эксперимента.

Таблица 5.5. Диагностическое дерево T_2

0	{1, 2, 3, 4, 5}	
	a	b
1	{1,1,5}{3, 2}	{1,5,1,4,6}

Предложенный в этом параграфе метод поиска диагностической последовательности для проведения простого безусловного диагностического эксперимента называется **методом дерева приемников**.

5.3. Неисправности

Неисправность – это изменение закона функционирования автомата, а именно, функции выходов.

Пример 5.2. Пусть автомат S задан таблицами 5.6 и 5.7.

Таблица 5.6. Таблица переходов автомата S

	1	2	3
1	2	3	1
2	3	1	2

Таблица 5.7. Таблица выходов автомата S

	1	2	3
1	1	2	2
2	2	1	1

Возможные неисправности порождают новые автоматы, например S_1 и S_2 (см. табл. 5.8, 5.9). В таблицах выходов жирным шрифтом обозначены неисправности.

Для решения **задачи анализа неисправностей** надо установить, является ли испытуемый автомат автоматом S , S_1 или S_2 .

Рассмотрим объединение автоматов S , S_1 , S_2 : $\tilde{S} = \Delta(S, S_1, S_2)$ (см. табл. 5.10, 5.11). Пусть $A(\tilde{S}) = \{1, 2, 3, 1_1, 2_1, 3_1, 1_2, 2_2, 3_2\}$ – множество допустимых начальных состояний. Проведем *установочный эксперимент* с

Таблица 5.8. Таблица выходов автомата S_1

	1	2	3
1	1	1	2
2	2	1	1

Таблица 5.9. Таблица выходов автомата S_2

	1	2	3
1	1	2	1
2	2	1	1

Таблица 5.10. Таблица переходов автомата \tilde{S}

	1	2	3	1_1	2_1	3_1	1_2	2_2	3_2
1	2	3	1	2_1	3_1	1_1	2_2	3_2	1_2
2	3	1	2	3_1	1_1	2_1	3_2	1_2	2_2

Таблица 5.11. Таблица выходов автомата \tilde{S}

	1	2	3	1_1	2_1	3_1	1_2	2_2	3_2
1	1	2	2	1	1	2	1	2	1
2	2	1	1	2	1	1	2	1	1

автоматом \tilde{S} : по выходной последовательности необходимо определить состояние, в котором оказался автомат.

Замечание 5.1. Очевидно, что диагностический эксперимент можно довести до установочного, определив конечное состояние после того, как найдены диагностическая последовательность, реакция автомата на нее и начальное состояние. Таким образом, для проведения простого безусловного установочного эксперимента применим метод дерева преемников, разобранный в предыдущем параграфе.

Рассмотрим входную последовательность «11122». Результаты применения этой последовательности к автомату \tilde{S} представлены в таблице 5.12. Так как различны все выходные последовательности, то последовательность «11122» является **установочной**: по реакции автомата на эту последовательность можно однозначно установить начальное и конечное состояния.

Таблица 5.12. Определение начального и конечного состояний автомата \tilde{S}

Начальное состояние	Реакция на «11122»	Конечное состояние
1	1 2 2 2 1	2
2	2 2 1 1 2	3
3	2 1 2 1 1	1
1 ₁	1 1 2 2 1	2 ₁
2 ₁	1 2 1 1 2	3 ₁
3 ₁	2 1 1 1 1	1 ₁
1 ₂	1 2 1 2 1	2 ₂
2 ₂	2 1 1 1 2	3 ₂
3 ₂	1 1 2 1 1	1 ₂

Кроме того, если конечное состояние принадлежит множеству $\{1, 2, 3\}$, то автомат исправен, в противном случае – имеет место неисправность.

Вывод: для решения задачи анализа неисправностей необходимо провести установочный эксперимент с автоматом, полученным из исходного и всех неисправных, путем их объединения.

Литература

- [1] Баранов С.И. *Синтез микропрограммных автоматов*. Ленинград: Энергия, 1974. 216 с.
- [2] Глушков В.М. *Синтез цифровых автоматов*. М.: Государственное издательство физико-математической литературы, 1962. 476 с.
- [3] Данилова О.Т., Файзуллин Р.Т. *Проектирование комбинационных схем*. Учебно-методическое пособие. Омск: Издательство Наследие, 2004. 178 с.
- [4] Закревский А.Д. *Алгоритмы синтеза дискретных автоматов*. М.: Наука, 1971. 512 с.
- [5] Майоров С.А., Новиков Г.И. *Принципы организации цифровых машин*. Ленинград: Машиностроение, 1974. 432 с.
- [6] Савельев А.Я. *Основы информатики*. М.: Издательство МГТУ им. Н.Э. Баумана, 2001.

Оглавление

1	Абстрактный синтез	3
1.1.	Дискретный автомат	3
1.2.	Определение абстрактного автомата	7
1.3.	Автоматы Мура и Мили	8
1.4.	Начальные языки	10
1.4.1.	Язык регулярных выражений алгебры событий	10
1.4.2.	Язык логических схем алгоритмов	11
1.4.3.	Язык граф-схем алгоритмов	12
1.5.	Автоматные (стандартные) языки	15
1.5.1.	Табличный метод	15
1.5.2.	Графовый метод	16
1.5.3.	Матричный метод	17
1.6.	Полностью и частично определенные автоматы .	19
1.7.	Синхронные и асинхронные автоматы	20
1.8.	Совмещенная модель автомата (С-автомат) . . .	22
1.9.	Связь между автоматами Мили и Мура	24
1.9.1.	Эквивалентные автоматы	24
1.9.2.	Преобразование автомата Мура в эквива- лентный автомат Мили	26
1.9.3.	Преобразование автомата Мили в эквива- лентный автомат Мура	27
1.10.	Минимизация полностью определенных автоматов	31
1.10.1.	Метод последовательных разбиений	31
1.10.2.	Таблица пар состояний	38

1.11. Минимизация частично определенных автоматов	41
1.12. Операции над автоматами	49
1.12.1. Разбиение автомата (декомпозиция) . . .	49
1.12.2. Соединение автоматов (композиция) . . .	51
1.13. Классификация	54
2 Структурный синтез	56
2.1. Элементы и базис	56
2.2. Схемы	61
2.3. Канонический метод структурного синтеза . . .	62
2.4. Функция входов элемента памяти	67
2.5. Пример канонического метода структурного синтеза	70
2.6. Выбор элементов памяти	76
2.6.1. Синтез автомата на задержках	76
2.6.2. Синтез автомата на триггерах со счетным входом	77
2.6.3. Синтез автомата на триггерах с раздель- ными входами	78
2.7. Графическая интерпретация канонического ме- тода	80
2.7.1. Синтез на задержках	81
2.7.2. Синтез на триггерах со счетным входом .	82
2.7.3. Синтез на триггерах с раздельными вхо- дами	83
3 Кодирование состояний автомата	84
3.1. Гонки в автомате	84
3.2. Противогоночное кодирование состояний	86
3.2.1. Развязывание пар переходов	86
3.2.2. Соседнее кодирование	91
3.3. Кодирование состояний и сложность логической схемы	92
3.3.1. Оптимальное кодирование при синтезе на задержках	93

3.3.2. Эвристический алгоритм кодирования, уменьшающий сложность логической схемы	95
4 Синтез микропрограммных автоматов	101
4.1. Структура дискретного устройства	101
4.2. ГСА микропрограммы	105
4.3. Синтез микропрограммного автомата Мили по ГСА	109
4.4. Синтез микропрограммного автомата Мура по ГСА	115
4.5. Списки переходов микропрограммных автоматов	119
4.6. Синтез микропрограммного С-автомата	121
4.7. Построение логической схемы микропрограмм- ного автомата	124
4.8. Интерпретационный метод структурного синтеза	127
4.8.1. Объединенное построение компонент функций возбуждения памяти и функ- ций выходов	127
4.8.2. Раздельный синтез для переходов в каж- дое состояние	128
4.8.3. Декомпозиция	130
4.8.4. Доопределение функций возбуждения	132
4.9. Устройство управления с программируемой ло- гикой	135
5 Эксперименты с автоматами	137
5.1. Классификация экспериментов	137
5.2. Простой безусловный диагностический экспери- мент	139
5.3. Неисправности	143
Литература	146

Н.Ф. Богаченко, Р.Т. Файзуллин

Синтез дискретных автоматов

Учебное пособие

Авторское редактирование

Подготовлено к печати
ООО «Издательство Наследие. Диалог-Сибирь»
Лицензия ЛР N 071680 от 04.06.98.
Подписано в печать 17.02.2006.
Формат 60 × 84 1/16. Усл.печ.л. 9,38. Уч.-изд.л. 6,25.
Тираж 100 экз.

Полиграфический центр КАН
644050, Омск-50, пр. Мира, 32, к.11
тел. (3812) 65-47-31
Лицензия ПЛД N 58-47 от 21.04.97.